GeoLink Core Ontology Design Patterns

GeoLink Technical Report 2014.12
Version: 0.1
(Supersedes OceanLink Technical Report 2014.2)

Contributors:
ADILA KRISNADHI — Wright State University
YINGJIE HU — University of California, Santa Barbara
ROBERT ARKO — Lamont-Doherty Earth Observatory, Columbia University
SUZANNE CARBOTTE — Lamont-Doherty Earth Observatory, Columbia University
CYNTHIA CHANDLER — Woods Hole Oceanographic Institution
MICHELLE CHEATHAM — Wright State University
DOUGLAS FILS — Ocean Leadership Consortium
TIMOTHY FININ — University of Maryland, Baltimore County
PASCAL HITZLER — Wright State University
KRZYSZTOF JANOWICZ — University of California, Santa Barbara
MATTHEW JONES — National Center for Ecological Analysis and Synthesis, University of California, Santa Barbara
AUDREY MICKLE — Woods Hole Oceanographic Institution
THOMAS NAROCK — Marymount University
MARGARET O'BRIEN — University of California, Santa Barbara
LISA RAYMOND — Woods Hole Oceanographic Institution
MARK SCHILDHAUER — National Center for Ecological Analysis and Synthesis, University of California, Santa Barbara
ADAM SHEPHERD — Woods Hole Oceanographic Institution
PETER WIEBE — Woods Hole Oceanographic Institution

Document Date: May 22, 2015

This work was supported by the National Science Foundation under award 1354778 *EAGER: Collaborative Research: EarthCube Building Blocks, Leveraging Semantics and Linked Data for Geoscience Data Sharing and Discovery*. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

0 Overview

0.1 Version Notes

Version 0.92

- Reverted URI scheme to using 'hash namespace'. Controlled vocabulary terms and named individuals reside in a separate namespace.
- Localized all class and properties. Relationships between two different patterns are facilitated through alignment patterns.
- Alignment between patterns are explicitly visualized in diagrams of all patterns.
- Graphical notation is modified to distinguish alignment axioms, shortcuts, etc.
- Drop the use of hasXXXType properties from the patterns and use subclassing instead.

Version 0.91

- Base IRI for all entities was changed from `http://schema.oceanlink.org/` into `http://schema.geolink.org/dev/` where `dev` will be replaced with a version number later.
- IRIs for all entities as well as IRI scheme in 0.3 was modified to use 'slash namespace', instead of 'hash namespace'.
- Relationship between Event pattern and Agent Role pattern was simplified. Property `hasActor` connecting Event and Agent is now a shortcut obtained from Actor.
- Property hasBirthEvent is redefined as a shortcut. Axiomatization is modified to account for changes in the Event pattern. Non-simple properties no longer appear in cardinality restrictions.
- Class OrganizationInformationObject is instroduced as a subclass of InformationObject specific for organization. Axiomatization is adjusted.
- Class Person from the Person pattern is explicitly added as a subclass of Agent to clarify the use of pattern for representing affiliation. Likewise, subclass relationship between Organization and Agent is added for the same purpose. Class PersonAffiliation and OrganizationAffiliation were added as a subclass of AgentAffiliation. Named individual `affiliate` was renamed into `affiliate_role`.
- Axiomatization was modified to prevent an information object to be described by yet another information object.
- hasFundingAgent relationship was redefined using ⟨AgentRole⟩ pattern.
- Added properties for including dates to programs.

0.2 About This Document

This is a deliverable of the GeoLink project supported by the the National Science Foundation under award ...

This document is derived from a deliverable of the OceanLink project (NSF award 1354778 *EAGER: Collaborative Research: EarthCube Building Blocks, Leveraging Semantics and Linked Data for Geoscience Data Sharing and Discovery*. It describes all core (ontology design) patterns that will serve two major roles in the project. First, they will act as the basis of the integration layer to which various data repositories within the GeoLink project will be mapped. Second, they are used to guide the navigational and retrieval components of the integrated user interface. The content of this document is mainly derived from the results of the OceanLink project meeting at University of Maryland, Baltimore County on November 5-8, 2013 as well as subsequent online communications.

In general, each core pattern in this document is presented as a separate chapter. Each chapter is then organized according to the following sequence.

1. An informal description is given, possibly with some visual depiction of the pattern.

2. An axiomatization of the pattern is presented in the form of a set of axioms. Each axiom can either be written using Description Logic (DL) notation (translatable to the Web Ontology Language (OWL)) or Datalog notation. The axiomatization formalizes the semantic relationships that are implicit in the informal description.
3. A set of alignment axioms, which express relationships of classes/properties in this pattern with some class/property in some other patterns in the pattern collection.
4. A collection of *views* for the pattern is given, if any. A view is simply a DL axiom or rule whose sole purpose is to ease the user task of expressing certain important queries. Although it is expressed as axioms, It does not constraint the meaning of a pattern, i.e., it makes no ontological commitment. In a sense, a view is simply a shortcut for queries analogous to the notion of view in relational databases.
5. Miscellaneous remarks if necessary.

0.3 Internationalized Resource Identifier (IRI) Scheme

We describe the convention we use for IRIs occurring in all of the patterns. We use recommendation from `http://protege.cim3.net/cgi-bin/wiki.pl?URIsURLsAndVersioning` and `http://www.w3.org/TR/swbp-vocab-pub/` to define the IRI scheme and configuration.

1. The *`version-info`* part in the URI scheme below stands for a version number, e.g., `1.0`, `2.0`, etc., or the string `dev`. Each version number signifies a major release of the pattern/vocabulary. If a version is under development and not yet stable for a release, we use `dev` in the version information part of each URI. There will only be one development version at any given time, however, this development version is expected to be changed/updated often until it is deemed suitable for a major release, in which the URIs will be frozen and a version number will be designated.
2. Each core pattern is provided in its own OWL file identified with an ontology IRI of the following scheme:
   - http://schema.geolink.org/*version-info*/*patternname*

   A pattern name always uses small letters. If the name consists of more than one words, it will be obtained by concatenating them all together. Versioning for patterns is done for the whole collection – we do not maintain separate versioning for each pattern.
3. Each alignment pattern contains axioms that align a core pattern to another core pattern or an external ontology. We consider such an alignment to be uni-directional. If a bidirectional alignment is necessary, it will be realized through two separate alignment patterns. The definition of an alignment pattern is defined in its own OWL file with an ontology IRI of the following scheme:
   - http://schema.geolink.org/*version-info*/*xxxx-to-yyyy*

   where *xxxx* and *yyyy* are core pattern names, and the direction of alignment is from the pattern *xxxx* to the pattern *yyyy*.
4. Class names and property names follows a camel case and their URIs are obtained by adding a hashed fragment to the pattern URI to which they were defined. Class names are always begun with a capital letter, while property names are begun with a small letter. That is, they follow the following scheme:
   - http://schema.geolink.org/*version-info*/*patternname#ClassName*
   - http://schema.geolink.org/*version-info*/*patternname#propertyName*
5. Named individuals that may appear in a pattern are maintained in a different namespace using the following scheme, which will be called the voc namespace henceforth:
   - http://schema.geolink.org/voc/*version-info*/iname#*individual_name*

   Named individuals are written using small letters whereby compound phrases are separated by an underscore ('_').

   External vocabulary terms, e.g., from NERC, can be adopted into GeoLink using this separate namespace as named individuals. The separate namespace provides us with a more flexible maintenance of controlled vocabulary terms, and furthermore, the versioning will be independent from the versioning of the pattern.
6. In pattern descriptions and axiomatizations, we usually omit the full URIs of class/property/named individuals, and simply use their name fragment. For example, we use `startsAtTime` instead of `http://schema.geolink.org/dev/time-tag#startsAtTime`.

7. Some external namespaces are used for some of the class/property/individual names. In that case, we use a prefixed format for the URI for those names. In this document, the external namespaces (and its prefix) are as follows:
   PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
   PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
   PREFIX owl: <http://www.w3.org/2002/07/owl#>
   PREFIX time: <http://www.w3.org/2006/time#>
   PREFIX dcterms: <http://purl.org/dc/terms/>
8. Different typefaces do not indicate different names/IRIs, e.g., startsAtTime and `startsAt` are the same property name within the same pattern.

Note that there exist some terms in our vocabulary that are similar to terms in existing (external) vocabulary. For example, our vocabulary introduces a class Person which is similar to the class `foaf:Person` (i.e., `http://xmlns.com/foaf/0.1/Person`). In such a situation, one can argue that it is better to reuse the term from an existing, external vocabulary directly. However, we think that this would imply a strong ontological commitment to that external vocabulary which is something that we do not necessarily want. By using our own namespace, we avoid this issue and only when necessary will we make a link/mapping/alignment to an existing vocabulary.

0.4 Notations

0.4.1 Graphical Notation

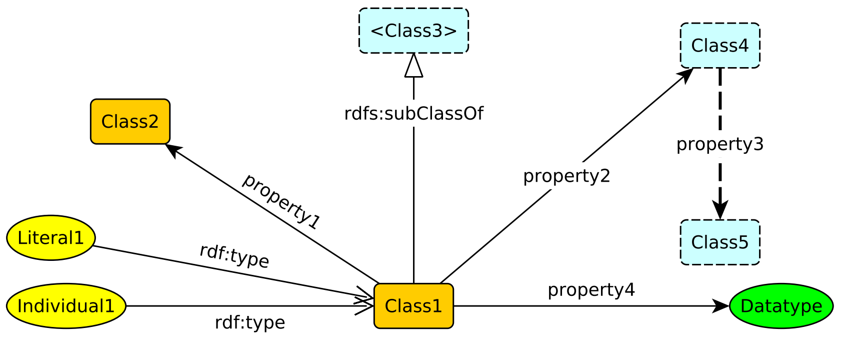

Figure 0.1: Graphical Notation for a pattern

Each pattern is visualized as a graph structure as depicted in Fig. 0.1. The label of a node or an edge represents the name of the class/property/datatype/individual it represents (see also Section 0.4.2 for more formal notation of class, property, datatype and individual). Let P be the pattern whose graph currently under consideration. An orange-colored node (e.g., `Class1` and `Class2`) is a class defined in P. A light-blue-colored node with dotted-line border (e.g., `Class3`, `Class4`, and `Class5`) is a class defined in a different, external pattern. A yellow-colored node (e.g., `Individual1` and `Literal1`) is an individual or literal defined in P. A green-colored node (e.g., `Datatype`)is an RDF datatype. A class name enclosed with an angled bracket means that the class name is also the name of the external pattern that defines the class.

(Directed) edges in the graph represent properties. An edge with an open arrowhead represents the `rdfs:subClassOf` property whose direction is from the subclass to the superclass. An edge with a bone-like arrowhead represents the `rdf:type` property whose direction is from the individual to the class it belongs to. Other edges with a standard arrowhead are either object properties or data properties (e.g., `property1`, `property2`, `property3`, and `property4`. The origin of the edge is the domain of the property, whereas the destination is the range of the property. If an edge is drawn with a dashed line (e.g., `property3`), then it indicates that it is defined in an external pattern.

Note that if a class is defined in an external pattern (i.e., light-blue-colored nodes) then there are almost always additional details (other related names, axioms, etc.) that are not explicitly spelled out in the current pattern. Typically, we only care about the class name and other details are not important for the current pattern. However, there are occasions in which some of the external details are important in the current pattern, e.g., if we want to assert some axiom involving both names defined in the current pattern and some class names in the inner part of the external pattern. In such a situation, we may visualize parts of the external pattern into the current pattern, hence edges with dotted line are necessary.

0.4.2 Notation for Axiomatization

Aximaotization of a pattern is a set of logical axiom, i.e., an ontology in the formal sense. An *axiom* is a logical assertion that is expressed either as a description logic (DL) axiom or a Datalog rule, possibly with equality. In general, DL notation is preferred over rule notation since the former immediately corresponds to OWL syntax. We use the latter here if it is not possible to express the assertion under consideration in DL. We describe both notations below.

0.4.2.1 DL Notation

An axiom is syntactically formed of *atomic symbols*, *literals* and a number of DL construct written in some special symbols. The atomic symbols are IRIs that, unless stated otherwise, are not reserved by OWL 2 and are disjointly divided into *class* IRIs, *object property* IRIs, *data property* IRIs, *named individual* IRIs, and *datatype* IRIs. Note that the same IRI occurring in different ontologies always represents the same entity (the same individual if the IRI is an individual IRI, the same class if it is a class IRI, etc.). Literals are data values appearing in the form of "abc"^^datatypeIRI where datatypeIRI is a datatype IRI. We list the kinds of axiom occurring in this document below together with their intended reading.

$C \sqsubseteq D$	C is a subclass of D, i.e., every instance of C is also an instance of D.
$C \equiv D$	C is an equivalent class of D, i.e., $C \sqsubseteq D$ and $D \sqsubseteq C$ both hold.
$C(a)$	a is an instance of C, i.e., a belongs to C or a $\texttt{rdf:type}$ C
$R(a, b)$	R connects a to b.
$R_1 \circ \cdots \circ R_m \sqsubseteq S$	where $m \geq 1$; if R_1 connects x_1 to $x_2, \ldots, R_m$ connects x_m to x_{m+1}, then S connects x_1 to x_{m+1}.
$\mathsf{alldifferent}(a_1, \ldots, a_n)$	a_i and a_j are different individuals for all i, j with $i \neq j, 1 \leq i, j \leq n$
$\mathsf{alldisjoint}(C_1, \ldots, C_n)$	C_i and C_j are pairwise disjoint classes for all i, j with $i \neq j, 1 \leq i, j \leq n$

where
- C, $C_1, \ldots, C_n$, and D are any kind of *class expressions*;
- $a, a_1, \ldots, a_n$ are always a *named individual* IRI;
- b is either a named individual IRI or a literal;
- R is an *property expression* whereby if b is a named individual IRI, then R is an *object property expression*, otherwise, (i.e., when b is a literal), R is a data property IRI;
- $R_1, \ldots, R_m$ and S are object property expressions; and
- $P_1, \ldots, P_m$ are data property IRIs.

Individuals represent actual objects from the domain of discourse. Here, named individuals are individuals that are given explicit name, i.e., a particular IRI. Individuals that are not named anywhere are called *anonymous individuals*. Anonymous individuals are analogous to blank nodes in Resource Description Framework (RDF).

Each datatype IRI represents a datatype, i.e., 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.

Both data property IRIs and object property expressions are binary relations. Each data property IRI is a binary relation connecting individuals in the domain of discourse to some literal. There are two predefined data properties: `owl:topDataProperty`, connecting all possible individuals with all literals, and `owl:bottomDataProperty` that does not connect any individual with any literal. An object property expression is an expression of the form R where R is an object property IRI, or S^- where S itself is some object property expression. Each object property expression is semantically interpreted as a binary relation between individuals (named or anonymous — the ones without an IRI) and S^- is the inverse binary relation of S, i.e., S connects x to y iff S^- connects y to x. There are two predefined object property IRI: `owl:topObjectProperty`, which connects all possible pairs of individuals in the domain of discourse and is denoted by U using DL notation, and `owl:bottomObjectProperty`, which does not connect any pair of individuals.

A *class expression* is an expression of one of the following form, each of which is interpreted as a some set of individuals in the domain of discourse;

- A where A is some class IRI, including two predefined class IRIs: `owl:Thing`, which is interpreted as the set of all individuals in the domain of discourse and denoted by $\top$ using DL notation, and `owl:Nothing`, which is interpreted as the empty set and denoted by $\bot$ using DL notation;
- $C_1 \sqcap \cdots \sqcap C_m$ with $m \geq 2$ and all $C_1, \ldots, C_m$ themselves class expressions; this class expression is interpreted as the set intersection of all sets represented by $C_1, \ldots, C_m$; the axiom $\mathsf{alldisjoint}(C_1, \ldots, C_m)$ is equivalent to a set of axioms of the form $C_i \sqcap C_j \sqsubseteq \bot$ for $1 \leq i < j \leq m$;
- $C_1 \sqcup \cdots \sqcup C_m$ with $m \geq 2$ and all $C_1, \ldots, C_m$ themselves class expressions; this class expression is interpreted as the set union of all sets given by $C_1, \ldots, C_m$;
- $\{a\}$ with a a named individual IRI, representing the singleton set containing exactly the individual given by a; the enumerated set $\{a_1, \ldots, a_m\}$ with $m \geq 1$ is equivalent to $\{a_1\} \sqcup \cdots \sqcup \{a_m\}$; also, the axiom $\mathsf{alldifferent}(a_1, \ldots, a_m)$ is equivalent to the set of axioms of the form $\{a_i\} \sqcap \{a_j\} \sqsubseteq \bot$ for $1 \leq i < j \leq m$;
- $\exists R.\mathsf{Self}$ where R is an object property expression; this represents the set of all individuals that are connected by R to itself.
- $\exists R.C$ where R is either an object property expression or a data property IRI and C is a class expression if R is an object property expression, otherwise, C is a datatype; this class expression represents the set of all individuals that are connected by R to individuals/literals that belong to C;
- $\forall R.C$ where R is either an object property expression or a data property IRI and C is a class expression if R is an object property expression, otherwise, C is a datatype; this class expression represents the set of all individuals that are connected by R only, if any, to individuals/literals that belong to C;
- $(\geqslant n\ R.C)$ where n is a natural number, R is either an object property expression or a data property IRI and C is a class expression if R is an object property expression, otherwise, C is a datatype; this class expression represents the set of all individuals that are connected by R to at least n individuals/literals that belong to C; note that $\exists R.C$ is equivalent to $(\geqslant 1\ R.C)$;
- $(\leqslant n\ R.C)$ where n is a natural number, R is either an object property expression or a data property IRI and C is a class expression if R is an object property expression, otherwise, C is a datatype; this class expression represents the set of all individuals that are connected by R to at most n individuals/literals that belong to C; note that $\forall R.C$ is equivalent to $(\leqslant 0\ R.\neg C)$;
- $(=n\ R.C)$ where n is a natural number; this is equivalent to $(\geqslant n\ R.C) \sqcap (\leqslant n\ R.C)$.

Datalog/Rule notation

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).

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. For example, let $\langle PAT \rangle$ be a pattern in which R is a property and we assert that the class A cover its domain and the class B covers its range. One obvious way to do it then is by asserting the rule $R(x,y) \rightarrow A(x) \wedge B(y)$ in $\langle PAT \rangle$ which is essentially an abbreviation of two rules: the domain restriction $R(x,y) \rightarrow A(x)$ (or $\exists R.\top \sqsubseteq A$ in DL notation) and the range restriction $R(x,y) \rightarrow B(y)$ (or $\exists R^{-}.\top \sqsubseteq B$ in DL notation). These two rules assert that for all individuals x and y, whenever x is connected to y through R, then it must be the case that both x belongs to A and y belongs to B.

For the purpose of reusability of patterns, however, the above rules force a very strong ontological commitment since they may prevent the reuse of R in other patterns, especially those which does not (wish to) use A or B in it. 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.

1 Agent

1.1 Description

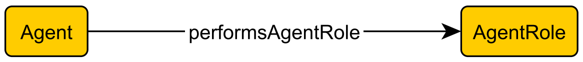

Figure 1.1: The Agent pattern

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 (Chapter 2) depicted in Figure 1.2.

1.2 Axiomatization

1.2.1 IRI Declarations

Prefix: : <http://schema.geolink.org/dev/agent#>
Ontology: <http://schema.geolink.org/dev/agent>

ObjectProperty: performsAgentRole
Class: AgentRole
Class: Agent

1.2.2 Core Axioms

Guarded domain and range restrictions of performsAgentRole.

$$\exists \mathsf{performsAgentRole}.\mathsf{AgentRole} \sqsubseteq \mathsf{Agent} \tag{1.1}$$

$$\mathsf{Agent} \sqsubseteq \forall \mathsf{performsAgentRole}.\mathsf{AgentRole} \tag{1.2}$$

Also, we assert the following pairwise-disjointness axiom.

$$\mathsf{Agent} \sqcap \mathsf{AgentRole} \sqsubseteq \bot \tag{1.3}$$

1.3 Alignment

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

1.3.1 Alignment with Agent Role pattern

We align Agent and AgentRole classes in the Agent pattern to the Agent and AgentRole class in the Agent Role pattern (Chapter 2) as depicted in Figure 1.2. 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#>

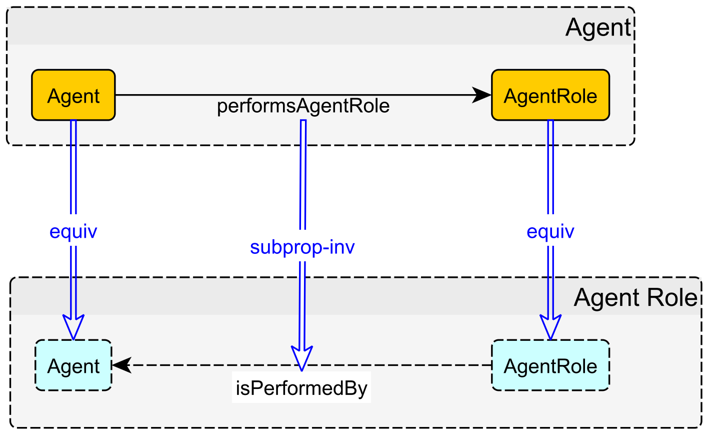

Figure 1.2: Alignment of Agent to Agent Role

Ontology: <http://schema.geolink.org/dev/agent-to-agentrole>

ObjectProperty: ecglag:performsAgentRole
ObjectProperty: ecglar:isPerformedBy

Class: ecglag:Agent
Class: ecglag:AgentRole

Class: ecglar:Agent
Class: ecglar:AgentRole

$$\text{ecglag:Agent} \equiv \text{ecglar:Agent} \tag{1.4}$$
$$\text{ecglag:AgentRole} \equiv \text{ecglar:AgentRole} \tag{1.5}$$
$$\text{ecglag:performsAgentRole} \sqsubseteq \text{ecglar:isPerformedBy}^{-} \tag{1.6}$$

2 Agent Role

2.1 Description

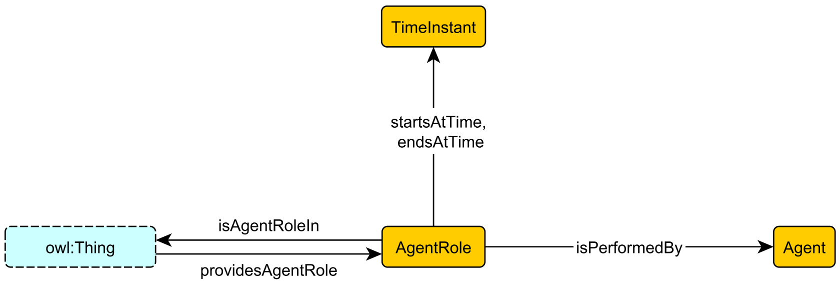

Figure 2.1: The Agent Role pattern

The Agent Role pattern describes a role that may be performed by an agent within a particular context, i.e., in an organization, a cruise, a project, etc. Such a role is temporally restricted, i.e., an AgentRole starts at one TimeInstant and ends at one TimeInstant.

An AgentRole is performed by exactly one Agent. An AgentRole is an agent-role in exactly one context (we simply use owl:Thing to cover such a context). For the inverse relationship, a thing (e.g., orgnaization, cruise, etc.) may provide an agent-role, although in this case, it may provide more than one agent-role. Alignment of the Agent Role pattern with Agent pattern and OWL Time ontology [Hobbs and Pan, 2006] are respectively depcited in 2.3 Figure 2.3.

2.2 Axiomatization

2.2.1 IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/agentrole#>
Ontology: <http://schema.geolink.org/dev/agentrole>

ObjectProperty: isAgentRoleIn
ObjectProperty: providesAgentRole
ObjectProperty: isPerformedBy
ObjectProperty: startsAtTime
ObjectProperty: endsAtTime

Class: AgentRole
Class: Agent
Class: TimeInstant
```

2.2.2 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.

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

$$\sqcap\,(=1\ \mathsf{startsAtTime.TimeInstant})\sqcap(=1\ \mathsf{endsAtTime.TimeInstant}) \tag{2.1}$$

$$\mathsf{providesAgentRole} \equiv \mathsf{isAgentRoleIn}^- \tag{2.2}$$

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.

$$\exists\mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{2.3}$$
$$\mathsf{AgentRole} \sqsubseteq \forall\mathsf{isPerformedBy.Agent} \tag{2.4}$$
$$\exists\mathsf{startsAtTime.TimeInstant} \sqsubseteq \mathsf{AgentRole} \tag{2.5}$$
$$\mathsf{AgentRole} \sqsubseteq \forall\mathsf{startsAtTime.TimeInstant} \tag{2.6}$$
$$\exists\mathsf{endsAtTime.TimeInstant} \sqsubseteq \mathsf{AgentRole} \tag{2.7}$$
$$\mathsf{AgentRole} \sqsubseteq \forall\mathsf{endsAtTime.TimeInstant} \tag{2.8}$$
$$\exists\mathsf{isAgentRoleIn}.\top \sqsubseteq \mathsf{AgentRole} \tag{2.9}$$
$$\top \sqsubseteq \forall\mathsf{providesAgentRole.AgentRole} \tag{2.10}$$

We express axiom (2.9) and (2.10) in the corresponding OWL file through the use of range and domain restrictions on the properties. Finally, we assert the following class disjointness axioms.

$$\mathsf{alldisjoint}(\mathsf{AgentRole}, \mathsf{Agent}, \mathsf{TimeInstant}) \tag{2.11}$$

2.3 Alignment

The Agent Role pattern specifies alignments with the Agent pattern and OWL Time ontology [Hobbs and Pan, 2006].

2.3.1 Alignment with Agent pattern

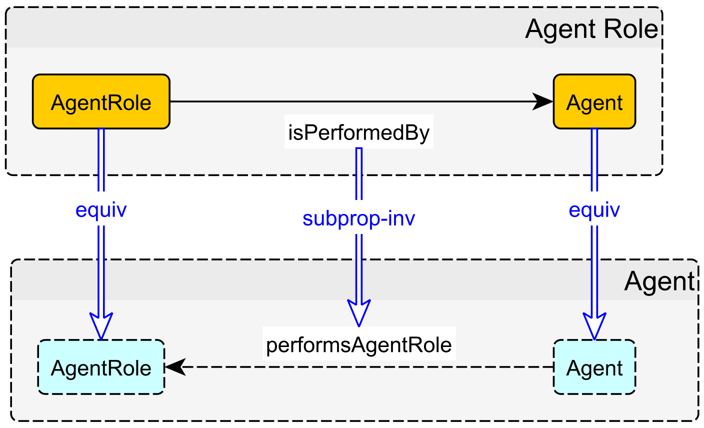

Figure 2.2: Agent Role aligned to Agent

We align Agent and AgentRole classes in the Agent Role pattern to the Agent and AgentRole class in the Agent pattern (Chapter 1) as depicted in Figure 2.2. 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/agent-to-agentrole>

ObjectProperty: ecglag:performsAgentRole
ObjectProperty: ecglar:isPerformedBy

Class: ecglar:AgentRole
Class: ecglar:Agent

Class: ecglag:AgentRole
Class: ecglag:Agent
```

$$\text{ecglar:AgentRole} \equiv \text{ecglag:AgentRole} \tag{2.12}$$

$$\text{ecglar:Agent} \equiv \text{ecglag:Agent} \tag{2.13}$$

$$\text{ecglar:isPerformedBy} \sqsubseteq \text{ecglar:performsAgentRole}^{-} \tag{2.14}$$

2.3.2 Alignment with OWL Time Ontology

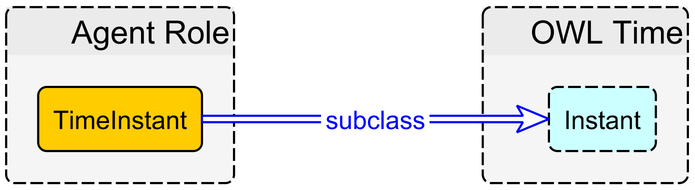

Figure 2.3: Agent Role aligned to OWL Time

We align TimeInstant with the time:Instant class from the OWL Time ontology as depicted in Figure 2.3.

```
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
Class: time:Instant
```

$$\text{ecglar:TimeInstant} \sqsubseteq \text{time:Instant} \tag{2.15}$$

2.4 Specializing AgentRole

When one uses the Agent Role pattern in some other pattern, (s)he usually wants to specialize it by introducing specific agent-role types. For instance, in a cruise, there can be a person who acts as the captain of the cruise. So, a cruise pattern can realize this by introducing a "captain role". Two modeling styles (Figure 2.4) are accomodated here:

(1) by subclassing the AgentRole class into, say the CaptainAgentRole class; or
(2) by using a controlled vocabulary term modeled as a named individual with a URI from the voc namespace, e.g., http://schema.geolink.org/voc/*version-info*/iname#cruise_captain_role), and have this connected from an instance of AgentRole by the hasAgentRoleType property.

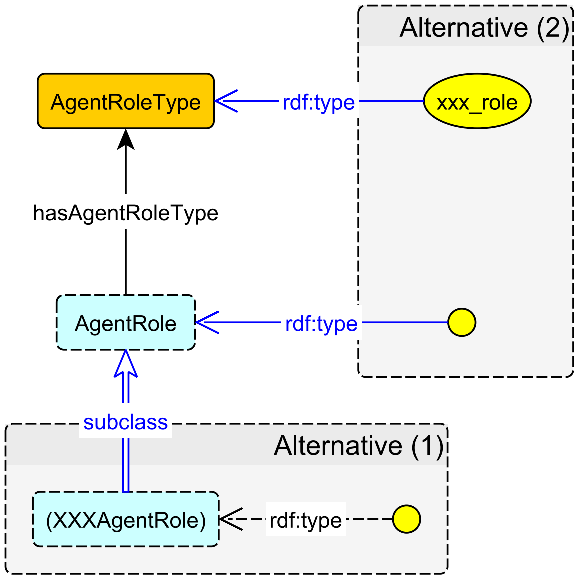

Figure 2.4: Typing AgentRole with subclassing and controlled vocabulary terms

Alternative (1) is considered preferable from an ontology modeling perspective because it allows one to make more general statements about the particular role as it is seen as a class. On the other hand, people with background in databases or frame systems sometimes prefer alternative (2) since they often find it more intuitive.

Semantically, OWL 2 expressive power means that (1) and (2) are equivalent. That is, we can express (1) in terms of (2) and vice versa. The following axioms express this equivalence.

$$\mathsf{XXXAgentRole} \equiv \mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{xxx\_role}\} \tag{2.16}$$

Axiom (2.16), however, will not be provided by the Agent Role pattern itself because the Agent Role pattern is not responsible for introducing XXXAgentRole class and the xxx_role individual name. Specializing Agent Role pattern can then be done in either of the following two alternatives.

(i) If style (1) is followed, the specialization should be aligned to the pattern given by the ontology IRI:
http://schema.geolink.org/*version-info*/agentrole
Axiom (2.16) is considered optional. The class name XXXAgentRole should be replaced with an actual class name in the specialization.

(ii) If style (2) is followed, the specialization should be aligned to the pattern given by the ontology IRI:
http://schema.geolink.org/*version-info*/agentroletype
Axiom (2.16) is considered mandatory with xxx_role replaced with some concrete individual name or controlled vocabulary term in the specialization.

Within the GeoLink pattern collection, any individual name introduced through either styles above will reside in the voc namespace.

The OWL implementation of http://schema.geolink.org/*version-info*/agentroletype imports the pattern in http://schema.geolink.org/*version-info*/agentrole. It also contains definitions of AgentRoleType class, as well as hasAgentRoleType and providesAgentRoleType property axiomatized according to the following axiomatization.

```
Prefix: : <http://schema.geolink.org/dev/agentroletype#>
Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>

Ontology: <http://schema.geolink.org/dev/agentroletype>
```

```
Import: <http://schema.geolink.org/dev/agentrole>

ObjectProperty: hasAgentRoleType

Class: ecglar:AgentRole
Class: AgentRoleType
```

$$\exists \mathsf{hasAgentRoleType}.\mathsf{AgentRoleType} \sqsubseteq \mathsf{ecglar{:}AgentRole} \tag{2.17}$$
$$\mathsf{ecglar{:}AgentRole} \sqsubseteq \forall \mathsf{hasAgentRoleType}.\mathsf{AgentRoleType} \tag{2.18}$$
$$\mathsf{ecglar{:}AgentRole} \sqsubseteq (=1\ \mathsf{hasAgentRoleType}.\mathsf{AgentRoleType}) \tag{2.19}$$

Note that the axioms above asserted that an AgentRole can only have exactly one AgentRoleType. Class disjointness is also asserted.

$$\mathsf{alldisjoint}(\mathsf{ecglar{:}AgentRole}, \mathsf{AgentRoleType}, \mathsf{ecglar{:}Agent}, \mathsf{ecglar{:}TimeInstant}) \tag{2.20}$$

2.5 Restricting Types of AgentRole

In some specialization of Agent Role pattern, one may want to restrict the possible types of agent-role. That is, in a particular context, one may want to assert that the only possible agent-roles allowed are role type X, role type Y, and role type Z. This can be accommodated in the specialization by including closure axioms, depending on whether style (1) or (2) from the previous section is being used.

- For style (1), suppose that AThing is the class representing objects (e.g., organizations, cruises, etc.) that provide some agent-roles. Also, suppose that there are only three kinds of roles AThing may provide: role type X, role type Y, and role type Z, resp. represented by the classes: XAgentRole, YAgentRole, and ZAgentRole. Then, this can be modeled using the following axioms, some of which may need to be asserted in the corresponding alignment pattern with the appropriate namespace:

$$\mathsf{XAgentRole} \sqsubseteq \mathsf{AgentRole} \tag{2.21}$$
$$\mathsf{YAgentRole} \sqsubseteq \mathsf{AgentRole} \tag{2.22}$$
$$\mathsf{ZAgentRole} \sqsubseteq \mathsf{AgentRole} \tag{2.23}$$
$$\mathsf{AThing} \sqsubseteq \forall \mathsf{providesAgentRole}.(\mathsf{XAgentRole} \sqcup \mathsf{YAgentRole} \sqcup \mathsf{ZAgentRole}) \tag{2.24}$$
$$\mathsf{alldisjoint}(\mathsf{XAgentRole}, \mathsf{YAgentRole}, \mathsf{ZAgentRole}) \tag{2.25}$$

- For style (2), we use the following individual names: x_roletype, y_roletype, and z_roletype. We then assert the following axioms, some of which may need to be asserted in the corresponding aligment pattern with the appropriate namespace:

$$\begin{aligned}\mathsf{AThing} \sqsubseteq \forall \mathsf{providesAgentRole}.((&\mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{x\_roletype}\})\\ &\sqcup (\mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{y\_roletype}\})\\ &\sqcup (\mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{z\_roletype}\}))\end{aligned} \tag{2.26}$$
$$\mathsf{alldifferent}(\mathsf{x\_roletype}, \mathsf{y\_roletype}, \mathsf{z\_roletype}) \tag{2.27}$$

If so desired, one may also add the typecasting axioms:

$$\mathsf{XAgentRole} \equiv \mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{x\_roletype}\} \tag{2.28}$$
$$\mathsf{YAgentRole} \equiv \mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{y\_roletype}\} \tag{2.29}$$
$$\mathsf{ZAgentRole} \equiv \mathsf{AgentRole} \sqcap \exists \mathsf{hasAgentRoleType}.\{\mathsf{z\_roletype}\} \tag{2.30}$$

3 Event

3.1 Description

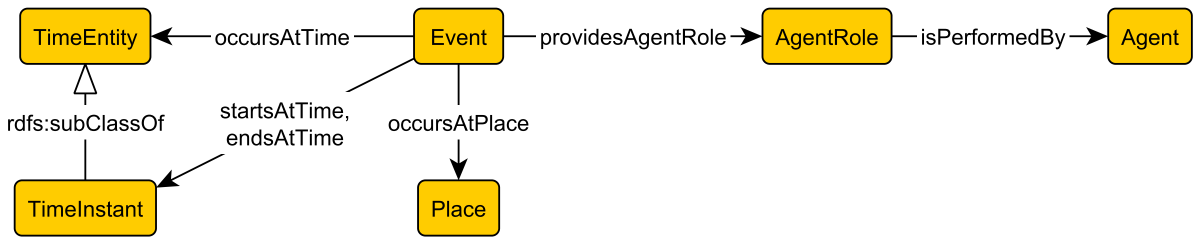

Figure 3.1: Event pattern

This is a simple pattern describing events. The modeling approach is based on the Simple Event Model (SEM) [van Hage et al., 2011]. 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 (Chapter 2) to which the Event pattern is aligned. When reusing this pattern, one may also restrict tbe set of all possible agent-role type an event may have, so that all agent-roles the event provides only come from that set, as described in Section 2.5, although this is not at all enforeced 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 [Hobbs and Pan, 2006]. 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 (Chapter 6) to model birth event. The Cruise pattern (Chapter 13) also uses this pattern.

3.2 Axiomatization

3.2.1 IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/event#>
Ontology: <http://schema.geolink.org/dev/event>

ObjectProperty: occursAtPlace
ObjectProperty: occursAtTime
ObjectProperty: startsAtTime
ObjectProperty: endsAtTime
ObjectProperty: providesAgentRole
ObjectProperty: isPerformedBy

Class: Event
Class: Place
Class: TimeInstant
Class: AgentRole
Class: Agent
```

3.2.2 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.

$$\mathsf{Event} \sqsubseteq \exists \mathsf{occursAtPlace}.\mathsf{Place} \sqcap \exists \mathsf{occursAtTime}.\mathsf{TimeEntity} \tag{3.1}$$

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.

$$\mathsf{startsAtTime} \sqsubseteq \mathsf{occursAtTime} \tag{3.2}$$
$$\mathsf{endsAtTime} \sqsubseteq \mathsf{occursAtTime} \tag{3.3}$$
$$\mathsf{TimeInstant} \sqsubseteq \mathsf{TimeEntity} \tag{3.4}$$

The Event pattern contains a specialization of the Agent Role pattern. As described in Fig. 3.1, 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.,

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy}.\mathsf{Agent}) \tag{3.5}$$

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. Mechanism to close the set of agent-role types was discussed in Section 2.5.

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

$$\exists \mathsf{occursAtPlace}.\mathsf{Place} \sqsubseteq \mathsf{Event} \tag{3.6}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{occursAtPlace}.\mathsf{Place} \tag{3.7}$$
$$\exists \mathsf{occursAtTime}.\mathsf{TimeEntity} \sqsubseteq \mathsf{Event} \tag{3.8}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{occursAtTime}.\mathsf{TimeEntity} \tag{3.9}$$
$$\exists \mathsf{startsAtTime}.\mathsf{TimeInstant} \sqsubseteq \mathsf{Event} \tag{3.10}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{startsAtTime}.\mathsf{TimeInstant} \tag{3.11}$$
$$\exists \mathsf{endsAtTime}.\mathsf{TimeInstant} \sqsubseteq \mathsf{Event} \tag{3.12}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{endsAtTime}.\mathsf{TimeInstant} \tag{3.13}$$
$$\exists \mathsf{providesAgentRole}.\mathsf{AgentRole} \sqsubseteq \mathsf{Event} \tag{3.14}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{providesAgentRole}.\mathsf{AgentRole} \tag{3.15}$$
$$\exists \mathsf{isPerformedBy}.\mathsf{Agent} \sqsubseteq \mathsf{AgentRole} \tag{3.16}$$
$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy}.\mathsf{Agent} \tag{3.17}$$

Finally, we assert class disjointness axioms as follows.

$$\mathsf{alldisjoint}(\mathsf{Event}, \mathsf{Place}, \mathsf{TimeEntity}, \mathsf{AgentRole}, \mathsf{Agent}) \tag{3.18}$$

3.3 Alignment

3.3.1 Alignment with Agent pattern

We align Event pattern with Agent pattern on the Agent class as depicted in Figure 3.2.

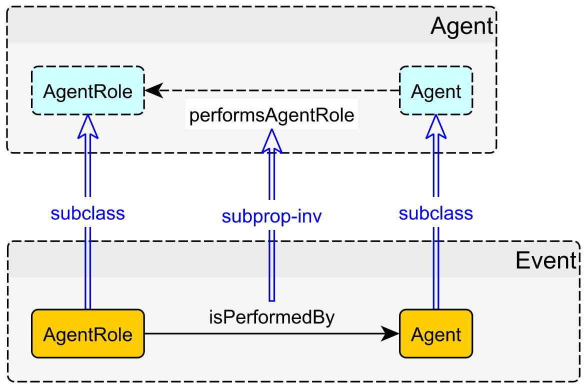

Figure 3.2: Event aligned to Agent

```
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
ObjectProperty: ecglag:performsAgentRole
Class: ecglev:Agent
Class: ecglev:AgentRole
Class: ecglag:Agent
Class: ecgleg:AgentRole
```

$$\text{ecglev:Agent} \sqsubseteq \text{ecglag:Agent} \tag{3.19}$$

$$\text{ecglev:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{3.20}$$

$$\text{ecglev:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{3.21}$$

3.3.2 Alignment with Agent Role pattern

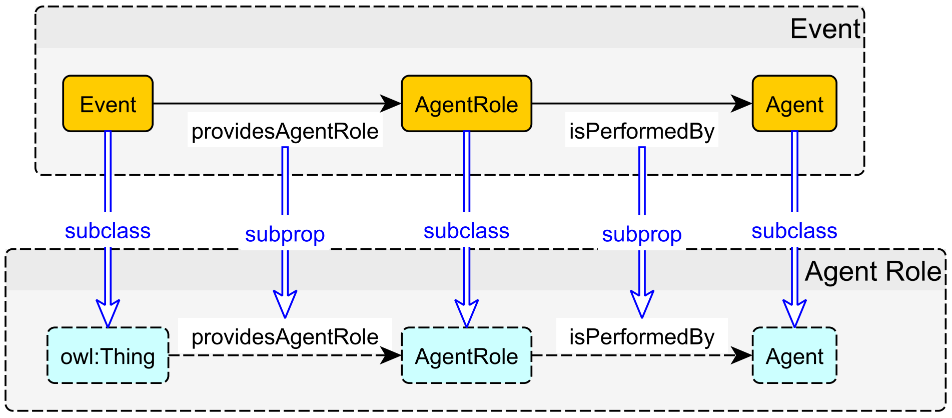

Figure 3.3: Event aligned to Agent Role

We align Event pattern with Agent Role pattern on AgentRole class, as well as providesAgentRole property, as depicted in Figure 3.3.

```
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
ObjectProperty: ecglev:isPerformedBy

ObjectProperty: ecglar:providesAgentRole
ObjectProperty: ecglar:isPerformedBy

Class: ecglev:AgentRole
Class: ecglev:Agent

Class: ecglar:AgentRole
Class: ecglar:Agent
```

$$\text{ecglev:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{3.22}$$
$$\text{ecglev:Agent} \sqsubseteq \text{ecglar:Agent} \tag{3.23}$$
$$\text{ecglev:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{3.24}$$
$$\text{ecglev:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{3.25}$$

3.3.3 Alignment with Place pattern

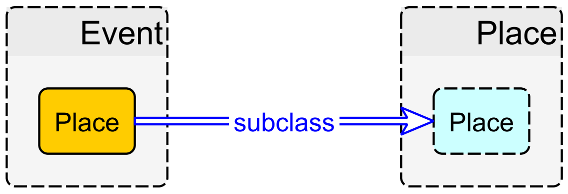

Figure 3.4: Event aligned to Place

We align Event pattern with Place pattern on the Place class as depicted in Figure 3.4.

```
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
Class: ecglpl:Place
```

$$\text{ecglev:Place} \sqsubseteq \text{ecglpl:Place} \tag{3.26}$$

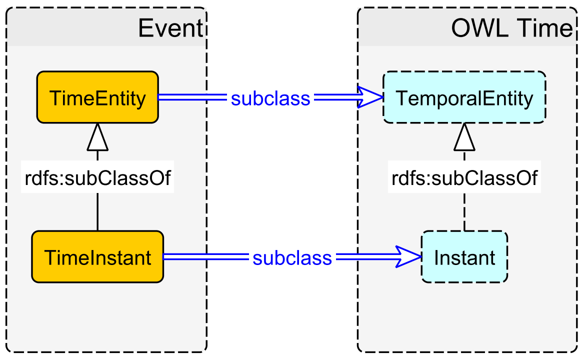

Figure 3.5: Event aligned to OWL Time

3.3.4 Alignment with OWL Time ontology

We align Event pattern with OWL Time ontology [Hobbs and Pan, 2006] as depicted in Figure 3.5.

```
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
Class: ecglev:TimeEntity
Class: time:Instant
Class: time:TemporalEntity
```

$$\mathsf{ecglev{:}TimeInstant} \sqsubseteq \mathsf{time{:}Instant} \tag{3.27}$$

$$\mathsf{ecglev{:}TimeEntity} \sqsubseteq \mathsf{time{:}TemporalEntity} \tag{3.28}$$

4 Information Object

4.1 Description

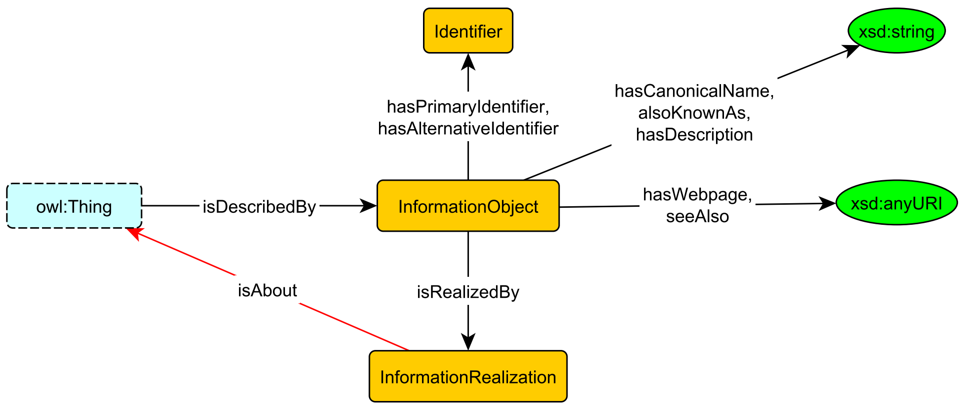

Figure 4.1: Information Object pattern

The Information Object pattern in Figure 4.1 models the additional information assigned to an object. Essentially, any thing 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 [Oberle et al., 2007]. 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 described it) can also be included via hasPrimaryIdentifier and hasAlternativeIdentifier property. The detail of identifier is modeled by a separate Identifier pattern in Chapter 5.

4.2 Axiomatization

4.2.1 IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/informationobject#>

Ontology: <http://schema.geolink.org/dev/informationobject>

ObjectProperty: isDescribedBy
ObjectProperty: isRealizedBy
```

ObjectProperty: hasPrimaryIdentifier
ObjectProperty: hasAlternativeIdentifier

DataProperty: hasWebpage
DataProperty: alsoKnownAs
DataProperty: hasDescription
DataProperty: seeAlso
DataProperty: hasCanonicalName

Class: InformationObject
Class: Identifier
Class: InformationRealization

4.2.2 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.

$$\top \sqsubseteq (\leqslant 1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{4.1}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\top) \tag{4.2}$$

$$\mathsf{InformationObject} \sqsubseteq \neg\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{4.3}$$

$$\mathsf{InformationRealization} \sqsubseteq \neg\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{4.4}$$

$$\mathsf{Identifier} \sqsubseteq \neg\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{4.5}$$

$$\mathsf{isRealizedBy}^{-} \circ \mathsf{isDescribedBy}^{-} \sqsubseteq \mathsf{isAbout} \tag{4.6}$$

Domain and range restrictions

$$\mathsf{range}(\mathsf{isDescribedBy}) \sqsubseteq \mathsf{InformationObject} \tag{4.7}$$

$$\exists \mathsf{isRealizedBy}.\mathsf{InformationRealization} \sqsubseteq \mathsf{InformationObject} \tag{4.8}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{isRealizedBy}.\mathsf{InformationRealization} \tag{4.9}$$

$$\mathsf{dom}(\mathsf{isAbout}) \sqsubseteq \mathsf{InformationRealization} \tag{4.10}$$

$$\exists \mathsf{hasCanonicalName}.\mathsf{xsd{:}string} \sqsubseteq \mathsf{InformationObject} \tag{4.11}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasCanonicalName}.\mathsf{xsd{:}string} \tag{4.12}$$

$$\exists \mathsf{alsoKnownAs}.\mathsf{xsd{:}string} \sqsubseteq \mathsf{InformationObject} \tag{4.13}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{alsoKnownAs}.\mathsf{xsd{:}string} \tag{4.14}$$

$$\exists \mathsf{hasDescription}.\mathsf{xsd{:}string} \sqsubseteq \mathsf{InformationObject} \tag{4.15}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasDescription}.\mathsf{xsd{:}string} \tag{4.16}$$

$$\exists \mathsf{hasWebpage}.\mathsf{xsd{:}anyURI} \sqsubseteq \mathsf{InformationObject} \tag{4.17}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasWebpage}.\mathsf{xsd{:}anyURI} \tag{4.18}$$

$$\exists \mathsf{seeAlso}.\mathsf{xsd{:}anyURI} \sqsubseteq \mathsf{InformationObject} \tag{4.19}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{seeAlso}.\mathsf{xsd{:}anyURI} \tag{4.20}$$

$$\exists \mathsf{hasAlternativeIdentifier}.\mathsf{Identifier} \sqsubseteq \mathsf{InformationObject} \tag{4.21}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasAlternativeIdentifier}.\mathsf{Identifier} \tag{4.22}$$

$$\exists \mathsf{hasPrimaryIdentifier}.\mathsf{Identifier} \sqsubseteq \mathsf{InformationObject} \tag{4.23}$$

$$\mathsf{InformationObject} \sqsubseteq \forall \mathsf{hasPrimaryIdentifier}.\mathsf{Identifier} \tag{4.24}$$

Disjointness axioms:

$$\mathsf{alldisjoint}(\mathsf{InformationObject}, \mathsf{InformationRealization}, \mathsf{Identifier}) \tag{4.25}$$

4.3 Alignment

4.3.1 Alignment with Identifier pattern

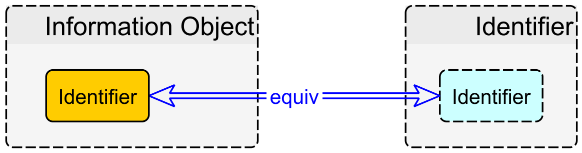

Figure 4.2: Information Object pattern aligned with Identifier pattern

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
Class: ecglid:Identifier

$$\text{ecglio:Identifier} \equiv \text{egclid:Identifier} \tag{4.26}$$

5 Identifier

5.1 Description

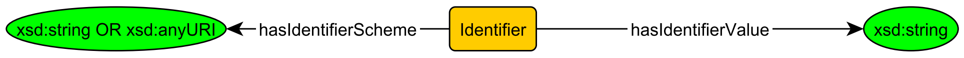

Figure 5.1: Identifier pattern

The Identifier pattern, as depicted in Figure 5.1, captures arbitrary identifiers that a data provider may use on their data. This is especially useful for non-URI identifiers such as local catalogue numbers, etc. We expect identifier information to be used in conjunction with the Information Object pattern (Chapter 5, 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. The scheme can either be a string literal or a URI.

5.2 Axiomatization

5.2.1 IRI Declaration

5.2.2 Core Axioms

$$\mathsf{Identifier} \sqsubseteq (=1\ \mathsf{hasIdentifierValue.xsd{:}string}) \tag{5.1}$$

$$\mathsf{Identifier} \sqsubseteq (\leqslant 1\ \mathsf{hasIdentifierScheme.(xsd{:}string} \sqcup \mathsf{xsd{:}anyURI)}) \tag{5.2}$$

$$\exists \mathsf{hasIdentifierValue.xsd{:}string} \sqsubseteq \mathsf{Identifier} \tag{5.3}$$

$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasIdentifierValue.xsd{:}string} \tag{5.4}$$

$$\exists \mathsf{hasIdentifierScheme.(xsd{:}string} \sqcup \mathsf{xsd{:}anyURI)} \sqsubseteq \mathsf{Identifier} \tag{5.5}$$

$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasIdentifierScheme.(xsd{:}string} \sqcup \mathsf{xsd{:}anyURI)} \tag{5.6}$$

6 Person

6.1 Description

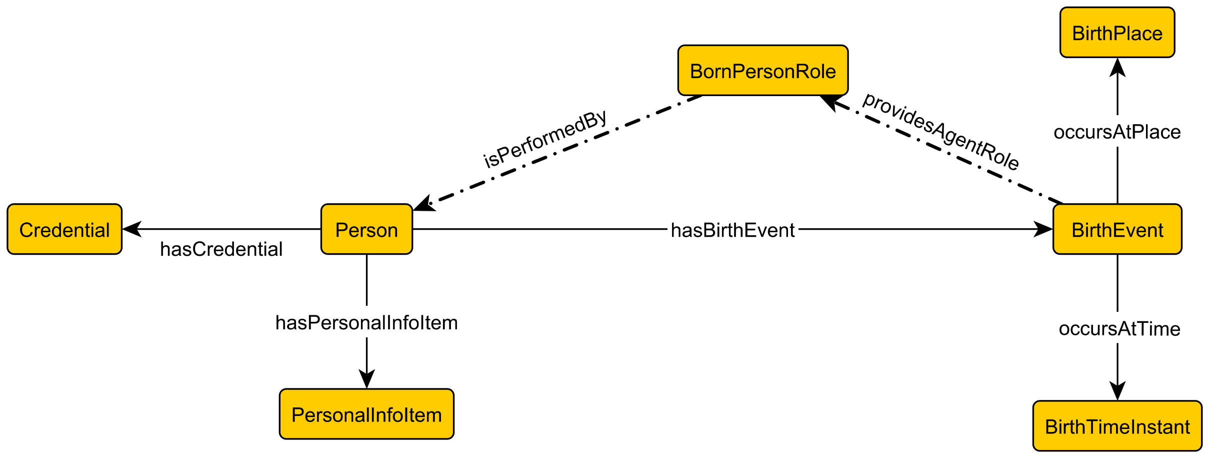

Figure 6.1: The Person pattern with birth event

The Person pattern, depicted in Figure 6.1, describes a person. In our modeling here, the person entity is represented via the Person class. Every person is an agent. Thus, we align Person with Agent from the Agent and Agent Role patterns (Chapter 1 and 2), as well as the Person pattern's AgentRole with that of Agent and Agent Role patterns.

We noticed that many attributes of Person are not permanent: even a person's name may change during his lifetime. The only thing that arguably never changes is birthday and birthplace. Note that the day/time a person dies also does not change, but for a living person, this information is always unknown. To incorporate the non-permanent/time-dependent attributes, a person may have some credentials, represented by the Credential class, and some personal information items, represented by the PersonalInfoItem class. The latter is used to model attributes such as name, address, etc. We also define an alignment to the Personal Info Item pattern (Chapter 7).

The birth of a person is modeled through BirthEvent. We assert that a person must have a birthday and a birthplace. The birth event itself provides a role performed by the person, represented by BornPersonRole, which is a particular kind of agent-role. As birth-event is a type of event, we define an alignment with the Event pattern (Chapter 3). Here, BirthEvent, BornPersonRole, and Person in the Person pattern are resepectively aligned to Event, AgentRole, and Agent in the Event pattern. In addition, the latter two are also aligned to AgentRole and Agent of the Agent Role pattern. Alignments to the Place pattern and the OWL Time ontology are also specified.

The BirthEvent itself is defined such that it provides exactly one agent-role that is an instance of type BornPersonRole and this agent-role is performed by the person.

In Figure 6.1, a few properties are depicted using a dotted-dashed line (providesAgentRole, isPerformedBy) to indicate that their domain and range restrictions are not fully imposed.

6.2 Axiomatization

6.2.1 IRI Declarations

```
Prefix: : <http://schema.geolink.org/dev/person#>
Ontology: <http://schema.geolink.org/dev/person>

ObjectProperty: hasPersonalInfoItem
ObjectProperty: hasCredential
ObjectProperty: hasBirthEvent
ObjectProperty: occursAtPlace
ObjectProperty: occursAtTime
ObjectProperty: providesAgentRole
ObjectProperty: isPerformedBy
ObjectProperty: performsAgentRole

Class: Person
Class: PersonalInfoItem
Class: Credential
Class: BirthEvent
Class: Place
Class: TimeInstant
Class: AgentRole
Class: BornPersonRole
```

6.2.2 Core Axioms

Every person is an agent. This is, however, axiomatized in the alignment with the Agent pattern (Section 6.3.1. A person may also have a credential and personal information item. There is no axiomatization except the domain and range restrictions of the hasCredential and hasPersonalInfoItem property defined later. So, we start with modeling birthplace and birthday of a person, which is done through the BirthEvent class. There is a one-to-one correspondencebetween Person and BirthEvent. This correspondence is given by the hasBirthEvent property.

$$\mathsf{Person} \sqsubseteq (=1\ \mathsf{hasBirthEvent.BirthEvent}) \tag{6.1}$$

$$\mathsf{BirthEvent} \sqsubseteq (=1\ \mathsf{hasBirthEvent}^{-}.\mathsf{Person}) \tag{6.2}$$

Every birth-event is an event that occurs exactly at one place and at one time point. The subclass relationship between BirthEvent and Event is specified in the alignment with the Event pattern. Also, there is a one-to-one correspondence between BirthEvent and BornPersonRole, which is given by the providesAgentRole property. A BornPersonRole is a type of AgentRole and it is performed by exactly one person.

$$\mathsf{BirthEvent} \sqsubseteq (=1\ \mathsf{occursAtPlace.BirthPlace}) \sqcap (=1\ \mathsf{occursAtTime.BirthTimeInstant}) \tag{6.3}$$

$$\mathsf{BirthEvent} \sqsubseteq (=1\ \mathsf{providesAgentRole.BornPersonRole}) \tag{6.4}$$

$$\mathsf{BornPersonRole} \sqsubseteq (=1\ \mathsf{providesAgentRole}^{-}.\mathsf{BirthEvent}) \sqcap (=1\ \mathsf{isPerformedBy.Person}) \tag{6.5}$$

Furthermore, this person has to be the person being born according to the birth-event. This will be entailed by all the above axioms, the guarded domain and range restrictions of providesAgentRole and hasBirthEvent (asserted later), and the following rule:

$$\mathsf{BornPersonRole}(x), \mathsf{providesAgentRole}(y,x), \mathsf{hasBirthEvent}(z,y) \to \mathsf{isPerformedBy}(x,z) \tag{6.6}$$

The above rule can actually be expressed in description logic as follows where $R_{BornPersonRole}$ is a fresh object property specifically defined for the BornPersonRole class:

$$\mathsf{BornPersonRole} \equiv \exists R_{BornPersonRole}.\mathsf{Self}$$

$$R_{BornPersonRole} \circ \mathsf{providesAgentRole}^{-} \circ \mathsf{hasBirthEvent}^{-} \sqsubseteq \mathsf{isPerformedBy}$$

Unfortunately, the above property chain leads to a violation on the regularity restriction for object properties in OWL 2. In the OWL implementation of the pattern, we thus include the (DL-safe) rule version instead.

Next, we axiomatize the domain and range restrictions for the properties in this pattern.

$$\exists \mathsf{hasBirthEvent.BirthEvent} \sqsubseteq \mathsf{Person} \tag{6.7}$$
$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasBirthEvent.BirthEvent} \tag{6.8}$$
$$\exists \mathsf{hasPersonalInfoItem.PersonalInfoItem} \sqsubseteq \mathsf{Person} \tag{6.9}$$
$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasPersonalInfoItem.PersonalInfoItem} \tag{6.10}$$
$$\exists \mathsf{hasCredential.Credential} \sqsubseteq \mathsf{Person} \tag{6.11}$$
$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasCredential.Credential} \tag{6.12}$$
$$\exists \mathsf{occursAtPlace.BirthPlace} \sqsubseteq \mathsf{BirthEvent} \tag{6.13}$$
$$\mathsf{BirthEvent} \sqsubseteq \forall \mathsf{occursAtPlace.BirthPlace} \tag{6.14}$$
$$\exists \mathsf{occursAtTime.BirthTimeInstant} \sqsubseteq \mathsf{BirthEvent} \tag{6.15}$$
$$\mathsf{BirthEvent} \sqsubseteq \forall \mathsf{occursAtTime.BirthTimeInstant} \tag{6.16}$$
$$\mathsf{BornPersonRole} \sqsubseteq \forall \mathsf{isPerformedBy.Person} \tag{6.17}$$
$$\exists \mathsf{providesAgentRole.BornPersonRole} \sqsubseteq \mathsf{BirthEvent} \tag{6.18}$$

Finally, class disjointness is asserted.

$$\mathsf{alldisjoint(Person, Credential, PersonalInfoItem, BirthEvent, BornPersonRole, BirthTimeInstant, BirthPlace)} \tag{6.19}$$

6.3 Alignment

6.3.1 Alignment with Agent pattern

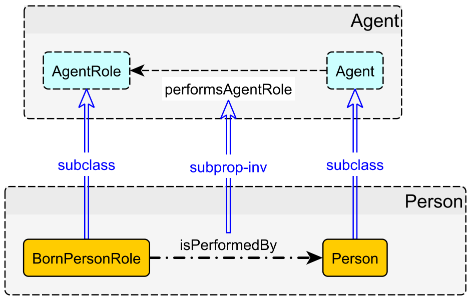

Figure 6.2: Person aligned with Agent

The alignment is depicted in Figure 6.2

```
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>

ObjectProperty: ecglpr:isPerformedBy
ObjectProperty: ecglag:performsAgentRole
Class: ecglpr:Person
Class: ecglpr:BornPersonRole
```

Class: ecglag:Agent
Class: ecglag:AgentRole

$$\text{ecglpr:Person} \sqsubseteq \text{ecglag:Agent} \tag{6.20}$$

$$\text{ecglpr:BornPersonRole} \sqsubseteq \text{ecglag:AgentRole} \tag{6.21}$$

$$\text{ecglpr:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{6.22}$$

6.3.2 Alignment with Agent Role pattern

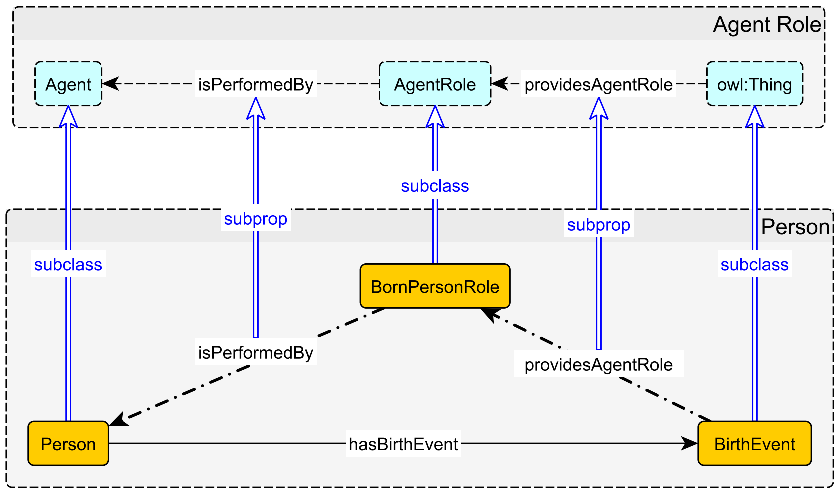

Figure 6.3: Person aligned with Agent Role

The alignment is depicted in Figure 6.3. Note that BirthEvent is by definition a subclass of owl:Thing.

Prefix: ecglar: <http://schema.geolink.org/dev/agentrole#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>

Ontology: <http://schema.geolink.org/dev/person-to-agentrole>

ObjectProperty: ecglpr:isPerformedBy
ObjectProperty: ecglpr:providesAgentRole

ObjectProperty: ecglar:isPerformedBy
ObjectProperty: ecglar:providesAgentRole

Class: ecglpr:BornPersonRole
Class: ecglpr:Person

Class: ecglar:AgentRole
Class: ecglar:Agent

$$\text{ecglpr:Person} \sqsubseteq \text{ecglar:Agent} \tag{6.23}$$

$$\text{ecglpr:BornPersonRole} \sqsubseteq \text{ecglar:AgentRole} \tag{6.24}$$

$$\text{ecglpr:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{6.25}$$

$$\text{ecglpr:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{6.26}$$

6.3.3 Alignment with Event pattern

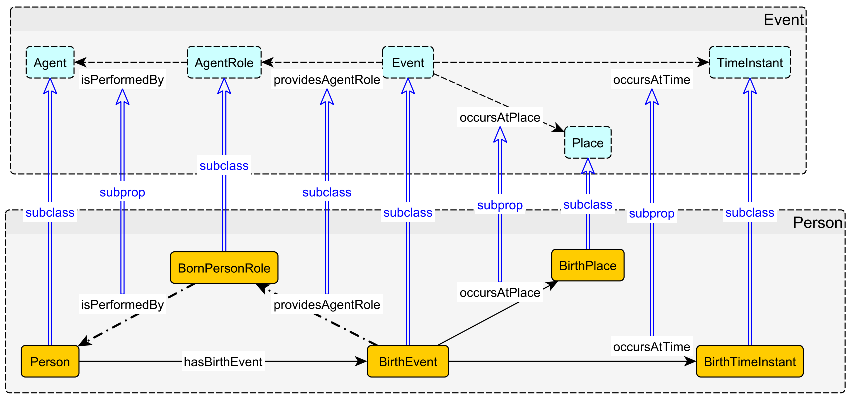

Figure 6.4: Person aligned with Event

The alignment is depicted in Figure 6.4.

```
Prefix: ecglev: <http://schema.geolink.org/dev/event#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>

Ontology: <http://schema.geolink.org/dev/person-to-event>

ObjectProperty: ecglev:occursAtTime
ObjectProperty: ecglev:occursAtPlace
ObjectProperty: ecglev:providesAgentRole
ObjectProperty: ecglev:isPerformedBy

ObjectProperty: ecglpr:occursAtTime
ObjectProperty: ecglpr:occursAtPlace
ObjectProperty: ecglpr:providesAgentRole
ObjectProperty: ecglpr:isPerformedBy

Class: ecglev:Event
Class: ecglev:AgentRole
Class: ecglev:Place
Class: ecglev:TimeInstant
Class: ecglev:Agent

Class: ecglpr:BirthEvent
Class: ecglpr:BornPersonRole
Class: ecglpr:BirthPlace
Class: ecglpr:BirthTimeInstant
Class: ecglpr:Person
```

$$\text{ecglpr:BirthEvent} \sqsubseteq \text{ecglev:Event} \tag{6.27}$$
$$\text{ecglpr:BornPersonRole} \sqsubseteq \text{ecglev:AgentRole} \tag{6.28}$$
$$\text{ecglpr:BirthPlace} \sqsubseteq \text{ecglev:Place} \tag{6.29}$$
$$\text{ecglpr:BirthTimeInstant} \sqsubseteq \text{ecglev:TimeInstant} \tag{6.30}$$

$$\text{ecglpr:Person} \sqsubseteq \text{ecglev:Agent} \tag{6.31}$$
$$\text{ecglpr:providesAgentRole} \sqsubseteq \text{ecglev:providesAgentRole} \tag{6.32}$$
$$\text{ecglpr:isPerformedBy} \sqsubseteq \text{ecglev:isPerformedBy} \tag{6.33}$$
$$\text{ecglpr:occursAtPlace} \sqsubseteq \text{ecglev:occursAtPlace} \tag{6.34}$$
$$\text{ecglpr:occursAtTime} \sqsubseteq \text{ecglev:occursAtTime} \tag{6.35}$$

6.3.4 Alignment with Personal Info Item pattern

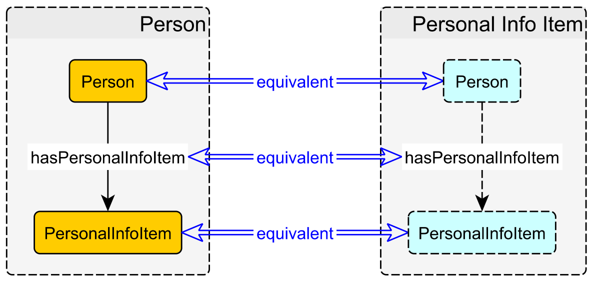

Figure 6.5: Person aligned with Personal Info Item

The alignment is depicted in Figure 6.5.

```
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
ObjectProperty: ecglpr:hasPersonalInfoItem

Class: ecglpi:Person
Class: ecglpi:PersonalInfoItem

Class: ecglpr:Person
Class: ecglpr:PersonalInfoItem
```

$$\text{ecglpr:Person} \equiv \text{ecglpi:Person} \tag{6.36}$$
$$\text{ecglpr:PersonalInfoItem} \equiv \text{ecglpi:PersonalInfoItem} \tag{6.37}$$
$$\text{ecglpr:hasPersonalInfoItem} \equiv \text{ecglpi:hasPersonalInfoItem} \tag{6.38}$$

6.3.5 Alignment with Place pattern

The alignment is depicted in Figure 6.6.

```
Prefix: ecglpl: <http://schema.geolink.org/dev/place#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>

Ontology: <http://schema.geolink.org/dev/person-to-place>

Class: ecglpl:Place
Class: ecglpr:BirthPlace
```

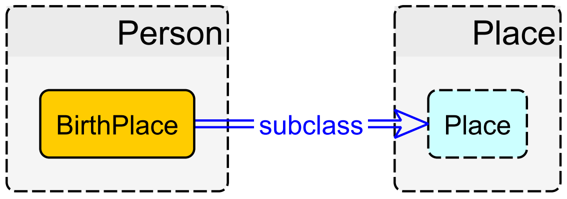

Figure 6.6: Person aligned with Place

$$\text{ecglpr:BirthPlace} \sqsubseteq \text{ecglpl:Place} \tag{6.39}$$

6.3.6 Alignment with OWL Time ontology

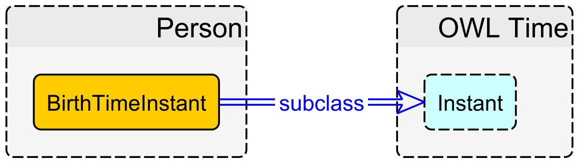

Figure 6.7: Person aligned with OWL Time

The alignment is depicted in Figure 6.7.

```
Prefix: time: <http://www.w3.org/2006/time#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>

Ontology: <http://schema.geolink.org/dev/person-to-owltime>

Class: time:Instant
Class: ecglpr:BirthTimeInstant
```

$$\text{ecglpr:BirthTimeInstant} \sqsubseteq \text{time:Instant} \tag{6.40}$$

7 Personal Info Item pattern

7.1 Description

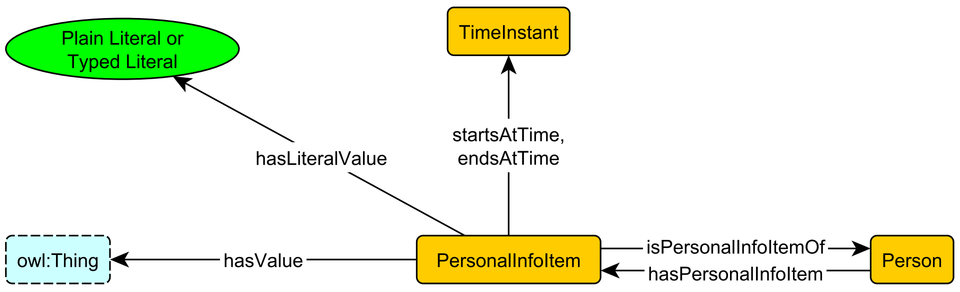

Figure 7.1: The Personal Info Item pattern

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 Chapter 8 for a concrete specialization of this pattern stub.

7.2 Axiomatization

7.2.1 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
ObjectProperty: hasPersonalInfoItem
ObjectProperty: startsAtTime
ObjectProperty: endsAtTime
```

```
ObjectProperty: hasValue

DataProperty: hasLiteralValue

Class: PersonalInfoItem
Class: TimeInstant
Class: Person
```

7.2.2 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.

$$\mathsf{PersonalInfoItem} \sqsubseteq (=1\ \mathsf{isPersonalInfoItemOf}.\mathsf{Person}) \tag{7.1}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq (=1\ \mathsf{startsAtTime}.\mathsf{time{:}Instant}) \sqcap (\leqslant 1\ \mathsf{endsAtTime}.\mathsf{time{:}Instant}) \tag{7.2}$$

$$\mathsf{hasPersonalInfoItem} \equiv \mathsf{isPersonalInfoItemOf}^{-} \tag{7.3}$$

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.

$$\exists \mathsf{isPersonalInfoItemOf}.\mathsf{Person} \sqsubseteq \mathsf{PersonalInfoItem} \tag{7.4}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq \forall \mathsf{isPersonalInfoItemOf}.\mathsf{Person} \tag{7.5}$$

$$\exists \mathsf{hasPersonalInfoItem}.\mathsf{PersonalInfoItem} \sqsubseteq \mathsf{Person} \tag{7.6}$$

$$\mathsf{Person} \sqsubseteq \forall \mathsf{hasPersonalInfoItem}.\mathsf{PersonalInfoItem} \tag{7.7}$$

$$\exists \mathsf{startsAtTime}.\mathsf{time{:}Instant} \sqsubseteq \mathsf{PersonalInfoItem} \tag{7.8}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq \forall \mathsf{startsAtTime}.\mathsf{time{:}Instant} \tag{7.9}$$

$$\exists \mathsf{endsAtTime}.\mathsf{time{:}Instant} \sqsubseteq \mathsf{PersonalInfoItem} \tag{7.10}$$

$$\mathsf{PersonalInfoItem} \sqsubseteq \forall \mathsf{endsAtTime}.\mathsf{time{:}Instant} \tag{7.11}$$

$$\mathsf{dom}(\mathsf{hasValue}) \sqsubseteq \mathsf{PersonalInfoItem} \tag{7.12}$$

$$\mathsf{dom}(\mathsf{hasLiteralValue}) \sqsubseteq \mathsf{PersonalInfoItem} \tag{7.13}$$

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

$$\mathsf{alldisjoint}(\mathsf{PersonalInfoItem}, \mathsf{Person}, \mathsf{TimeInstant}) \tag{7.14}$$

7.3 Alignment

7.3.1 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
ObjectProperty: ecglpr:hasPersonalInfoItem

Class: ecglpi:Person
```

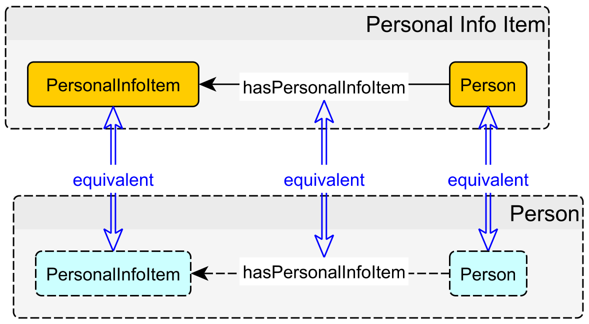

Figure 7.2: Personal Info Item pattern aligned to Person pattern

Class: ecglpi:PersonalInfoItem

Class: ecglpr:Person
Class: ecglpr:PersonalInfoItem

$$\text{ecglpi:Person} \equiv \text{ecglpr:Person} \tag{7.15}$$
$$\text{ecglpi:PersonalInfoItem} \equiv \text{ecglpr:PersonalInfoItem} \tag{7.16}$$
$$\text{ecglpi:hasPersonalInfoItem} \equiv \text{ecglpr:hasPersonalInfoItem} \tag{7.17}$$

7.3.2 Alignment with OWL Time ontology

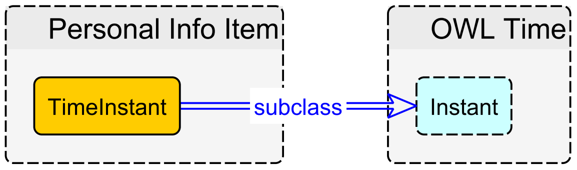

Figure 7.3: Personal Info Item pattern aligned to OWL Time ontology

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
Class: ecglpi:TimeInstant

$$\text{ecglpi:TimeInstant} \sqsubseteq \text{time:Instant} \tag{7.18}$$

8 Person Name

8.1 Description

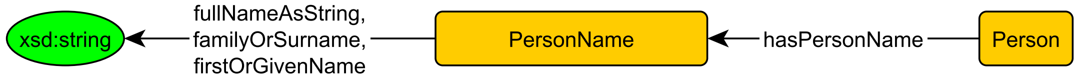

Figure 8.1: The Person Name pattern

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.

8.2 Axiomatization

8.2.1 IRI Declaration

```
Prefix: : <http://schema.geolink.org/dev/personname#>

Ontology: <http://schema.geolink.org/dev/personname>

ObjectProperty: hasPersonName

DataProperty: fullNameAsString
DataProperty: firstOrGivenName
DataProperty: familyOrSurname

Class: Person
Class: PersonName
```

8.2.2 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.

$$\begin{aligned} \mathsf{PersonName} &\sqsubseteq (=1\ \mathsf{fullNameAsString.xsd{:}string}) \sqcap (\leqslant 1\ \mathsf{firstOrGivenName.xsd{:}string}) \\ &\sqcap\ (\leqslant 1\ \mathsf{familyOrSurname.xsd{:}string}) \quad (8.1) \\ \mathsf{Person} &\sqsubseteq \exists \mathsf{hasPersonName.PersonName} \quad (8.2) \end{aligned}$$

Next, we assert domain and range restrictions for fullNameAsString, firstOrGivenName, and familyOrSurname properties. Note that these are data properties whose range is rdf:PlainLiteral

$$\begin{aligned} \exists \mathsf{hasPersonName.PersonName} &\sqsubseteq \mathsf{Person} \quad (8.3) \\ \mathsf{Person} &\sqsubseteq \forall \mathsf{hasPersonName.PersonName} \quad (8.4) \\ \exists \mathsf{fullNameAsString.xsd{:}string} &\sqsubseteq \mathsf{PersonName} \quad (8.5) \\ \mathsf{PersonName} &\sqsubseteq \forall \mathsf{fullNameAsString.xsd{:}string} \quad (8.6) \\ \exists \mathsf{firstOrGivenName.xsd{:}string} &\sqsubseteq \mathsf{PersonName} \quad (8.7) \\ \mathsf{PersonName} &\sqsubseteq \forall \mathsf{firstOrGivenName.xsd{:}string} \quad (8.8) \\ \exists \mathsf{familyOrSurname.xsd{:}string} &\sqsubseteq \mathsf{PersonName} \quad (8.9) \\ \mathsf{PersonName} &\sqsubseteq \forall \mathsf{familyOrSurname.xsd{:}string} \quad (8.10) \end{aligned}$$

Finally, the class disjointness is asserted.

$$\mathsf{PersonName} \sqcap \mathsf{Person} \sqsubseteq \bot \quad (8.11)$$

8.3 Alignment

8.3.1 Alignment with Personal Info Item pattern

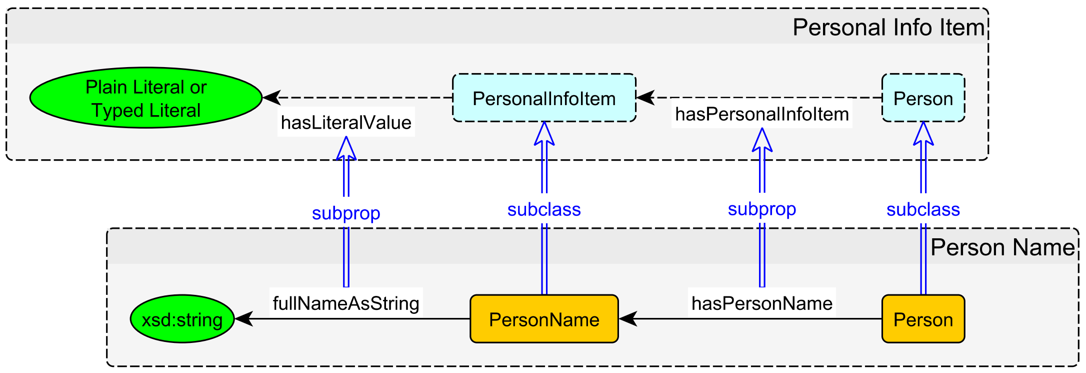

Figure 8.2: Person Name pattern aligned to Personal Info Item pattern

```
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
DataProperty: ecglpn:fullNameAsString

ObjectProperty: ecglpi:hasPersonalInfoItem
```

```
DataProperty: ecglpi:hasLiteralValue

Class: ecglpn:Person
Class: ecglpn:PersonName

Class: ecglpi:Person
Class: ecglpi:PersonalInfoItem
```

$$\text{ecglpn:Person} \sqsubseteq \text{ecglpi:Person} \tag{8.12}$$
$$\text{ecglpn:PersonName} \sqsubseteq \text{ecglpi:PersonalInfoItem} \tag{8.13}$$
$$\text{ecglpn:hasPersonName} \sqsubseteq \text{ecglpi:hasPersonalInfoItem} \tag{8.14}$$
$$\text{ecglpn:fullNameAsString} \sqsubseteq \text{ecglpi:hasLiteralValue} \tag{8.15}$$
$$\tag{8.16}$$

8.3.2 Alignment with Person pattern

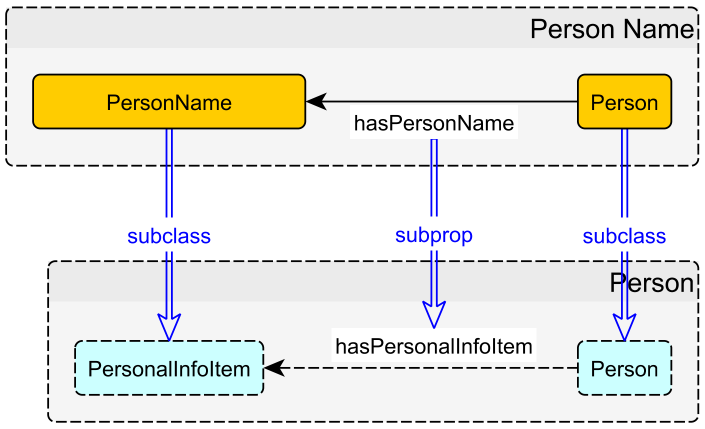

Figure 8.3: Person Name pattern aligned to Personal Info Item pattern

```
Prefix: ecglpn: <http://schema.geolink.org/dev/personname#>
Prefix: ecglpr: <http://schema.geolink.org/dev/person#>

Ontology: <http://schema.geolink.org/dev/personname-to-person>

ObjectProperty: ecglpn:hasPersonName
ObjectProperty: ecglpr:hasPersonalInfoItem

Class: ecglpn:Person
Class: ecglpn:PersonName

Class: ecglpr:Person
Class: ecglpr:PersonalInfoItem
```

$$\mathsf{ecglpn{:}Person} \sqsubseteq \mathsf{ecglpr{:}Person} \tag{8.17}$$
$$\mathsf{ecglpn{:}PersonName} \sqsubseteq \mathsf{ecglpr{:}PersonalInfoItem} \tag{8.18}$$
$$\mathsf{ecglpn{:}hasPersonName} \sqsubseteq \mathsf{ecglpr{:}hasPersonalInfoItem} \tag{8.19}$$

9 Organization

9.1 Description

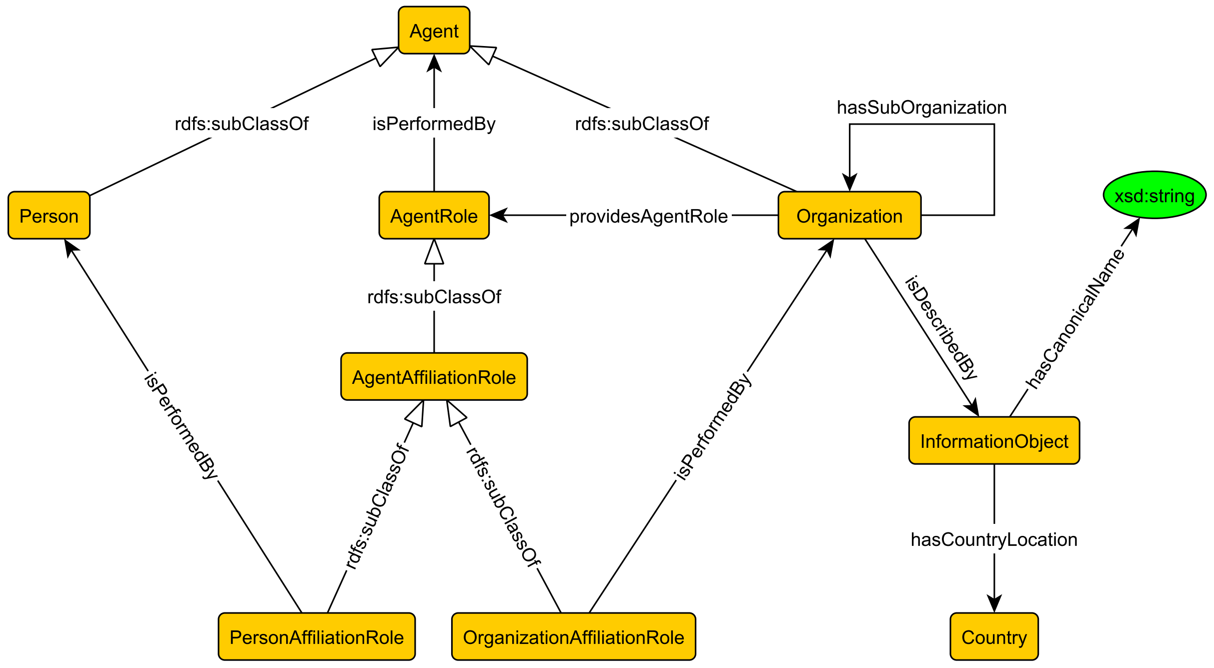

Figure 9.1: The Organization pattern

This pattern describes organizations, as depicted in Figure 9.1. 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 align 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.

9.2 Axiomatization

9.2.1 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
ObjectProperty: isDescribedBy
ObjectProperty: isPerformedBy
ObjectProperty: providesAgentRole
ObjectProperty: hasCountryLocation
DataProperty: hasCanonicalName

Class: Organization
Class: Person
Class: InformationObject
Class: Country
Class: AgentRole
Class: Agent
Class: AgentAffiliationRole
Class: OrganizationAffiliationRole
Class: PersonAffiliationRole
```

9.2.2 Core Axioms

Each organization is an agent and is described by exactly one instance of InformationObject. Further, every InformationObject describes exactly one organization.

$$\mathsf{Organization} \sqsubseteq \mathsf{Agent} \tag{9.1}$$
$$\mathsf{Person} \sqsubseteq \mathsf{Agent} \tag{9.2}$$
$$\mathsf{Organization} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{9.3}$$
$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\mathsf{Organization}) \tag{9.4}$$

The hasSubOrganization property is transitive.

$$\mathsf{hasSubOrganization} \circ \mathsf{hasSubOrganization} \sqsubseteq \mathsf{hasSubOrganization} \tag{9.5}$$

Information objects that describe organizations have exactly one canonical name.

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{hasCanonicalName}.\mathsf{xsd{:}string}) \tag{9.6}$$

Hierarchy of agent-roles:

$$\mathsf{AgentAffiliationRole} \sqsubseteq \mathsf{AgentRole} \tag{9.7}$$
$$\mathsf{PersonAffiliationRole} \sqsubseteq \mathsf{AgentAffiliationRole} \tag{9.8}$$
$$\mathsf{PersonAffiliationRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Person}) \tag{9.9}$$
$$\mathsf{OrganizationAffiliationRole} \sqsubseteq \mathsf{AgentAffiliationRole} \tag{9.10}$$
$$\mathsf{OrganizationAffiliationRole} \sqsubseteq (=1\ \mathsf{isPerformedBy.Organization}) \tag{9.11}$$

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.

$$\exists\mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{9.12}$$
$$\mathsf{AgentRole} \sqsubseteq \forall\mathsf{isPerformedBy.Agent} \tag{9.13}$$
$$\mathsf{PersonAffiliationRole} \sqsubseteq \forall\mathsf{isPerformedBy.Person} \tag{9.14}$$
$$\mathsf{OrganizationAffiliationRole} \sqsubseteq \forall\mathsf{isPerformedBy.Organization} \tag{9.15}$$
$$\exists\mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{Organization} \tag{9.16}$$
$$\mathsf{Organization} \sqsubseteq \forall\mathsf{providesAgentRole.AgentRole} \tag{9.17}$$
$$\exists\mathsf{hasSubOrganization.Organization} \sqsubseteq \mathsf{Organization} \tag{9.18}$$
$$\mathsf{Organization} \sqsubseteq \forall\mathsf{hasSubOrganization.Organization} \tag{9.19}$$
$$\exists\mathsf{isDescribedBy.InformationObject} \sqsubseteq \mathsf{Organization} \tag{9.20}$$
$$\mathsf{Organization} \sqsubseteq \forall\mathsf{isDescribedBy.InformationObject} \tag{9.21}$$
$$\exists\mathsf{hasCountryLocation.Country} \sqsubseteq \mathsf{InformationObject} \tag{9.22}$$
$$\mathsf{InformationObject} \sqsubseteq \forall\mathsf{hasCountryLocation.Country} \tag{9.23}$$
$$\exists\mathsf{hasCanonicalName.xsd{:}string} \sqsubseteq \mathsf{InformationObject} \tag{9.24}$$
$$\mathsf{InformationObject} \sqsubseteq \forall\mathsf{hasCanonicalName.xsd{:}string} \tag{9.25}$$

Finally, we assert the following disjointness axioms.

$$\mathsf{alldisjoint}(\mathsf{Agent}, \mathsf{AgentRole}, \mathsf{OrganizationInformationObject}{,}\mathsf{Country}) \tag{9.26}$$
$$\mathsf{Organization} \sqcap \mathsf{Person} \sqsubseteq \bot \tag{9.27}$$
$$\mathsf{PersonAffiliationRole} \sqcap \mathsf{OrganizationAffiliationRole} \sqsubseteq \bot \tag{9.28}$$

9.3 Alignment

9.3.1 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
ObjectProperty: ecglag:performsAgentRole
Class: ecglor:Agent
Class: ecglor:AgentRole
Class: ecglag:Agent
Class: ecglag:AgentRole
```

$$\mathsf{ecglor{:}Agent} \sqsubseteq \mathsf{ecglag{:}Agent} \tag{9.29}$$
$$\mathsf{ecglor{:}AgentRole} \sqsubseteq \mathsf{ecglag{:}AgentRole} \tag{9.30}$$
$$\mathsf{ecglor{:}isPerformedBy} \sqsubseteq \mathsf{ecglag{:}performsAgentRole}^{-} \tag{9.31}$$

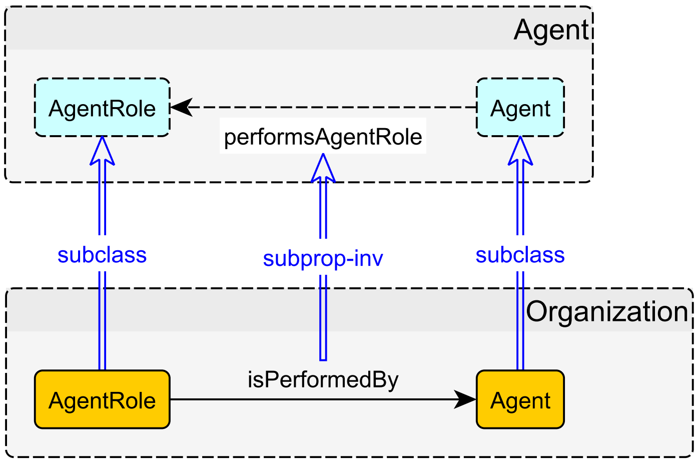

Figure 9.2: The Organization pattern aligned to Agent pattern

9.3.2 Alignment with Agent Role pattern

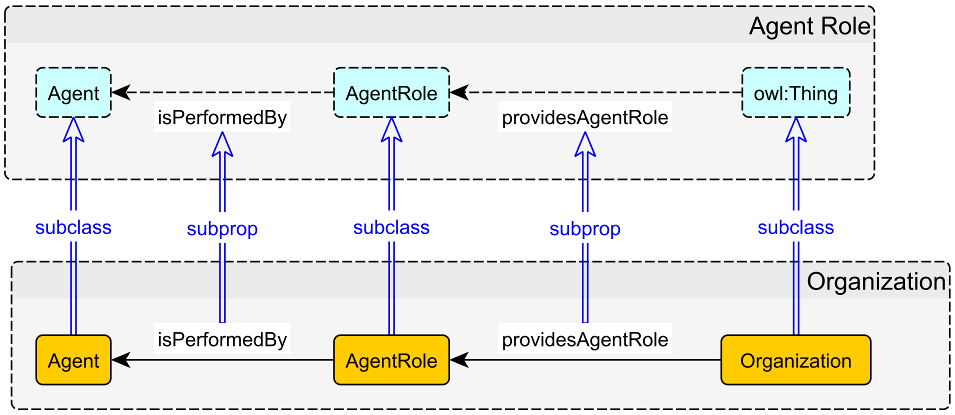

Figure 9.3: The Organization pattern aligned to Agent Role pattern

Figure 9.3 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
ObjectProperty: ecglor:providesAgentRole
ObjectProperty: ecglar:isPerformedBy
ObjectProperty: ecglar:providesAgentRole
Class: ecglor:Agent
Class: ecglor:AgentRole
Class: ecglar:Agent
Class: ecglar:AgentRole
```

$$\text{ecglor:Agent} \sqsubseteq \text{ecglar:Agent} \tag{9.32}$$
$$\text{ecglor:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{9.33}$$
$$\text{ecglor:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{9.34}$$
$$\text{ecglor:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{9.35}$$

9.3.3 Alignment with Person pattern

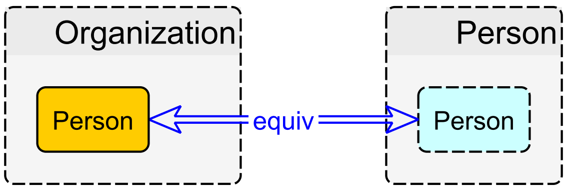

Figure 9.4: The Organization pattern aligned to Person pattern

```
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
Class: ecglpr:Person
```

$$\text{ecglor:Person} \equiv \text{ecglpr:Person} \tag{9.36}$$

9.3.4 Alignment with Information Object pattern

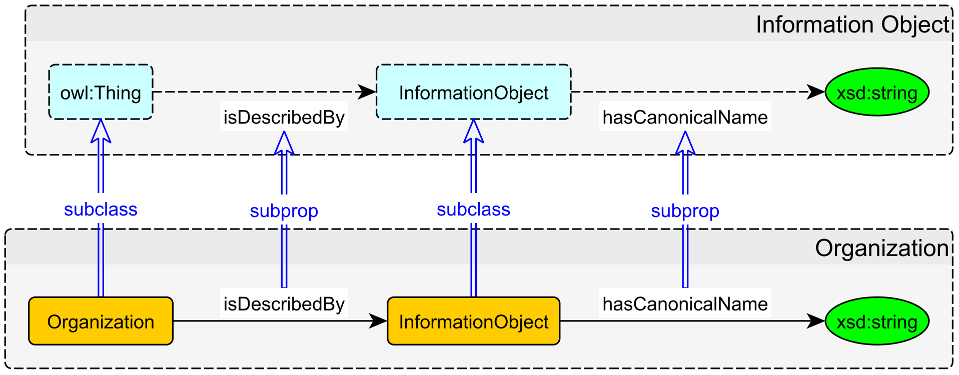

Figure 9.5: The Organization pattern aligned to Information Object pattern

Figure 9.5 depicts the alignment. Organization is by definition a subclass of owl:Thing, hence an axiom is not needed.

```
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
ObjectProperty: ecglio:isDescribedBy

DataProperty: ecglor:hasCanonicalName
DataProperty: ecglio:hasCanonicalName

Class: ecglor:InformationObject
Class: ecglio:InformationObject
```

$$\text{ecglor:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{9.37}$$

$$\text{ecglor:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{9.38}$$

$$\text{ecglor:hasCanonicalName} \sqsubseteq \text{ecglio:hasCanonicalName} \tag{9.39}$$

10 Funding Award

10.1 Description

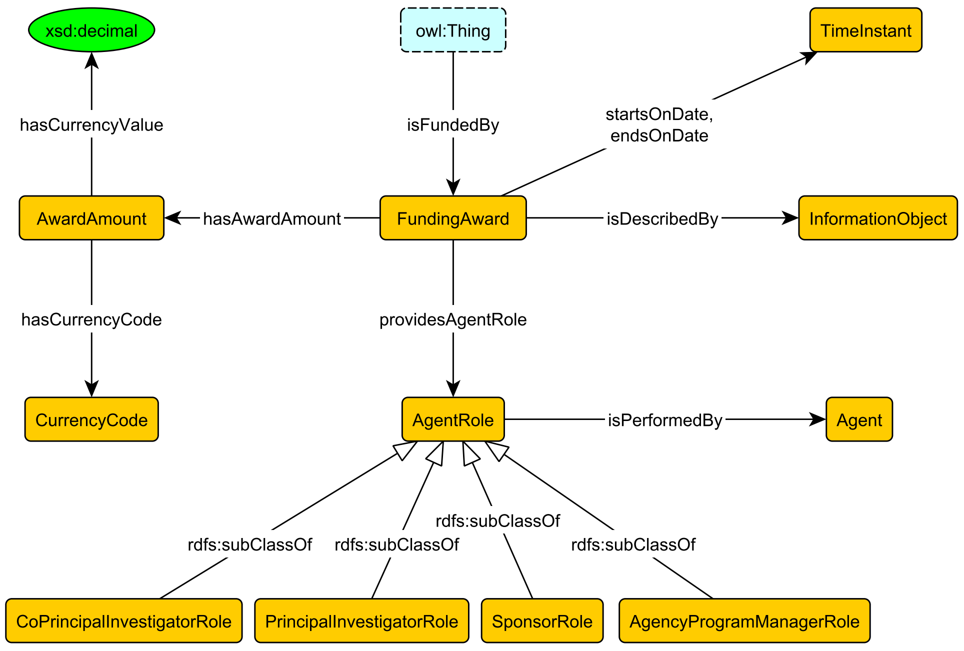

Figure 10.1: The Funding Award pattern

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.

10.2 Axiomatization

10.2.1 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
ObjectProperty: startsOnDate
ObjectProperty: endsOnDate
ObjectProperty: isDescribedBy
ObjectProperty: providesAgentRole
ObjectProperty: isPerformedBy
ObjectProperty: hasAwardAmount
ObjectProperty: hasCurrencyCode
DataProperty: hasCurrencyValue

Class: FundingAward
Class: Agent
Class: TimeInstant
Class: AgentRole
Class: SponsorRole
Class: PrincipalInvestigatorRole
Class: CoPrincipalInvestigatorRole
Class: AgencyProgramManagerRole
Class: InformationObject
Class: AwardAmount
Class: CurrencyCode
```

10.2.2 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.

$$\mathsf{FundingAward} \sqsubseteq (=1\ \mathsf{isDescribedBy.InformationObject}) \tag{10.1}$$

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.

$$\mathsf{FundingAward} \sqsubseteq (=1\ \mathsf{startsOnDate.TimeInstant}) \sqcap (=1\ \mathsf{endsOnDate.TimeInstant}) \tag{10.2}$$
$$\mathsf{FundingAward} \sqsubseteq \exists \mathsf{providesAgentRole.SponsorRole} \tag{10.3}$$
$$\mathsf{SponsorRole} \sqsubseteq \mathsf{AgentRole} \tag{10.4}$$
$$\mathsf{PrincipalInvestigatorRole} \sqsubseteq \mathsf{AgentRole} \tag{10.5}$$
$$\mathsf{CoPrincipalInvestigatorRole} \sqsubseteq \mathsf{AgentRole} \tag{10.6}$$
$$\mathsf{AgencyProgramManagerRole} \sqsubseteq \mathsf{AgentRole} \tag{10.7}$$
$$\mathsf{FundingAward} \sqsubseteq (\leqslant 1\ \mathsf{hasAwardAmount.AwardAmount}) \tag{10.8}$$
$$\mathsf{AwardAmount} \sqsubseteq (=1\ \mathsf{hasCurrencyValue.xsd{:}decimal}) \tag{10.9}$$
$$\mathsf{AwardAmount} \sqsubseteq (=1\ \mathsf{hasCurrencyCode.CurrencyCode}) \tag{10.10}$$

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.

$$\mathsf{range(isFundedBy)} \sqsubseteq \mathsf{FundingAward} \tag{10.11}$$
$$\exists \mathsf{startsOnDate.time{:}Instant} \sqsubseteq \mathsf{FundingAward} \tag{10.12}$$
$$\mathsf{FundingAward} \sqsubseteq \forall \mathsf{startsOnDate.time{:}Instant} \tag{10.13}$$
$$\exists \mathsf{endsOnDate.time{:}Instant} \sqsubseteq \mathsf{FundingAward} \tag{10.14}$$
$$\mathsf{FundingAward} \sqsubseteq \forall \mathsf{endsOnDate.time{:}Instant} \tag{10.15}$$
$$\exists \mathsf{providesAgentRole.AgentRole} \sqsubseteq \mathsf{FundingAward} \tag{10.16}$$
$$\mathsf{FundingAward} \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{10.17}$$
$$\exists \mathsf{isPerformedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{10.18}$$
$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy.Agent} \tag{10.19}$$
$$\exists \mathsf{isDescribedBy.InformationObject} \sqsubseteq \mathsf{FundingAward} \tag{10.20}$$
$$\mathsf{FundingAward} \sqsubseteq \forall \mathsf{isDescribedBy.InformationObject} \tag{10.21}$$
$$\exists \mathsf{hasAwardAmount.AwardAmount} \sqsubseteq \mathsf{FundingAward} \tag{10.22}$$
$$\mathsf{FundingAward} \sqsubseteq \forall \mathsf{hasAwardAmount.AwardAmount} \tag{10.23}$$
$$\exists \mathsf{hasCurrencyValue.xsd{:}decimal} \sqsubseteq \mathsf{AwardAmount} \tag{10.24}$$
$$\mathsf{AwardAmount} \sqsubseteq \forall \mathsf{hasCurrencyValue.xsd{:}decimal} \tag{10.25}$$
$$\exists \mathsf{hasCurrencyCode.CurrencyCode} \sqsubseteq \mathsf{AwardAmount} \tag{10.26}$$
$$\mathsf{AwardAmount} \sqsubseteq \forall \mathsf{hasCurrencyCode.CurrencyCode} \tag{10.27}$$

Disjointness axioms:

$$\mathsf{alldisjoint(FundingAward, InformationObject, AwardAmount, CurrencyCode, AgentRole, Agent, TimeInstant)} \tag{10.28}$$

10.3 Alignment

10.3.1 Alignment with Agent pattern

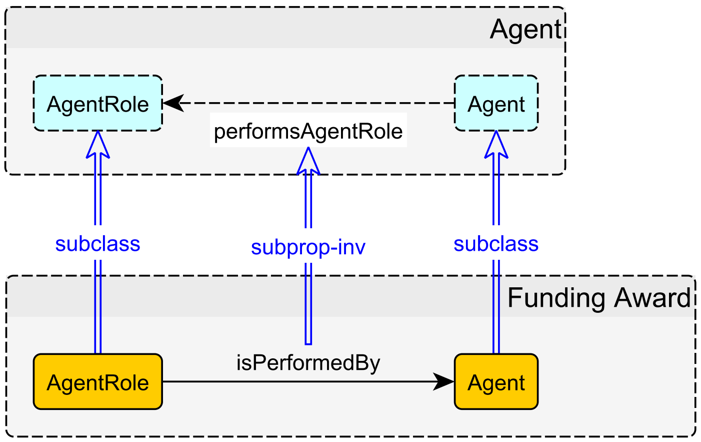

Figure 10.2: The Funding Award pattern aligned to Agent pattern

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
ObjectProperty: ecglag:performsAgentRole
Class: ecglfa:Agent
Class: ecglfa:AgentRole
Class: ecglag:Agent
Class: ecglag:AgentRole
```

$$\text{ecglfa:Agent} \sqsubseteq \text{ecglag:Agent} \tag{10.29}$$

$$\text{ecglfa:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{10.30}$$

$$\text{ecglfa:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{10.31}$$

10.3.2 Alignment with Agent Role pattern

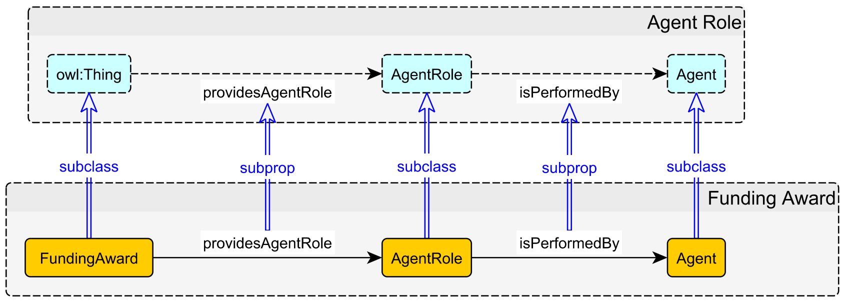

Figure 10.3: The Funding Award pattern aligned with Agent Role pattern

```
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
ObjectProperty: ecglfa:providesAgentRole
ObjectProperty: ecglar:isPerformedBy
ObjectProperty: ecglar:providesAgentRole
Class: ecglfa:Agent
Class: ecglfa:AgentRole
Class: ecglar:Agent
Class: ecglar:AgentRole
```

$$\text{ecglfa:Agent} \sqsubseteq \text{ecglar:Agent} \tag{10.32}$$

$$\text{ecglfa:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{10.33}$$

$$\text{ecglfa:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{10.34}$$

$$\text{ecglfa:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{10.35}$$

10.3.3 Alignment with Information Object pattern

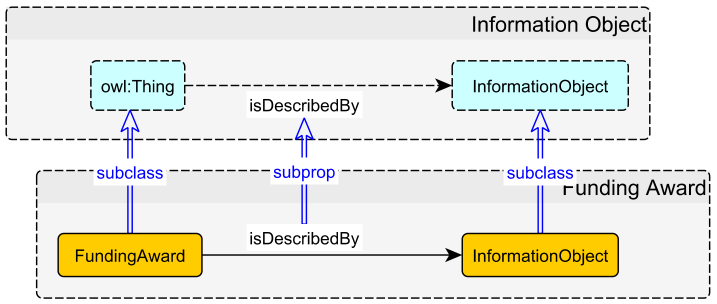

Figure 10.4: The Funding Award pattern aligned with Information Object pattern

```
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
ObjectProperty: ecglio:isDescribedBy
Class: ecglfa:InformationObject
Class: ecglio:InformationObject
```

$$\text{ecglfa:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{10.36}$$

$$\text{ecglfa:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{10.37}$$

10.3.4 Alignment with OWL Time

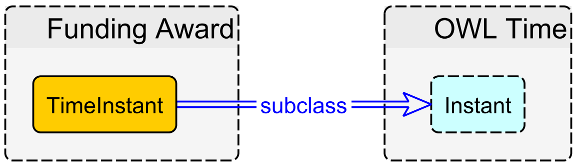

Figure 10.5: The Funding Award pattern aligned with OWL Time

```
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
Class: time:Instant
```

$$\mathsf{ecglfa{:}TimeInstant} \sqsubseteq \mathsf{time{:}Instant} \tag{10.38}$$

11 Program

11.1 Description

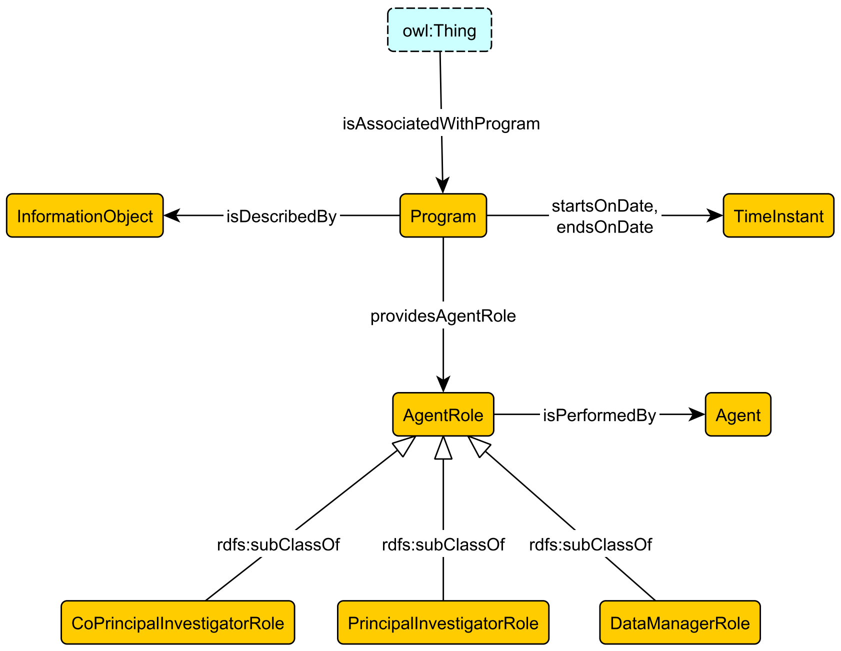

Figure 11.1: The Program pattern

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 Figure 11.1. 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 thorugh the use of InformationObject pattern.

11.2 Axiomatization

11.2.1 IRI Declarations

11.2.2 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.

$$\mathsf{Program} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{11.1}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^-.\mathsf{Program}) \tag{11.2}$$
$$\mathsf{Program} \sqsubseteq (\leqslant 1\ \mathsf{startsOnDate}.\mathsf{TimeInstant}) \sqcap (\leqslant 1\ \mathsf{endsOnDate}.\mathsf{TimeInstant}) \tag{11.3}$$

PrincipalInvestigatorRole, CoPrincipalInvestigatorRole, and DataManagerRole are types of AgentRole.

$$\mathsf{PrincipalInvestigatorRole} \sqsubseteq \mathsf{AgentRole} \tag{11.4}$$
$$\mathsf{CoPrincipalInvestigatorRole} \sqsubseteq \mathsf{AgentRole} \tag{11.5}$$
$$\mathsf{DataManagerRole} \sqsubseteq \mathsf{AgentRole} \tag{11.6}$$

We assert guarded domain and range restrictions below.

$$\mathsf{range}(\mathsf{isAssociatedWithProgram}) \sqsubseteq \mathsf{Program} \tag{11.7}$$
$$\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Program} \tag{11.8}$$
$$\mathsf{Program} \sqsubseteq \forall \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{11.9}$$
$$\exists \mathsf{startsOnDate}.\mathsf{TimeInstant} \sqsubseteq \mathsf{Program} \tag{11.10}$$
$$\mathsf{Program} \sqsubseteq \forall \mathsf{startsOnDate}.\mathsf{TimeInstant} \tag{11.11}$$
$$\exists \mathsf{endsOnDate}.\mathsf{TimeInstant} \sqsubseteq \mathsf{Program} \tag{11.12}$$
$$\mathsf{Program} \sqsubseteq \forall \mathsf{endsOnDate}.\mathsf{TimeInstant} \tag{11.13}$$
$$\exists \mathsf{providesAgentRole}.\mathsf{AgentRole} \sqsubseteq \mathsf{Program} \tag{11.14}$$
$$\mathsf{Program} \sqsubseteq \forall \mathsf{providesAgentRole}.\mathsf{AgentRole} \tag{11.15}$$
$$\exists \mathsf{isPerformedBy}.\mathsf{Agent} \sqsubseteq \mathsf{AgentRole} \tag{11.16}$$
$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy}.\mathsf{Agent} \tag{11.17}$$

Disjointness axioms

$$\mathsf{alldisjoint}(\mathsf{Program}, \mathsf{InformationObject}, \mathsf{TimeInstant}, \mathsf{AgentRole}, \mathsf{Agent}) \tag{11.18}$$
$$\mathsf{alldisjoint}(\mathsf{PrincipalInvestigatorRole}, \mathsf{CoPrincipalInvestigatorRole}, \mathsf{DataManagerRole}) \tag{11.19}$$

11.3 Alignment

11.3.1 Alignment with Agent pattern

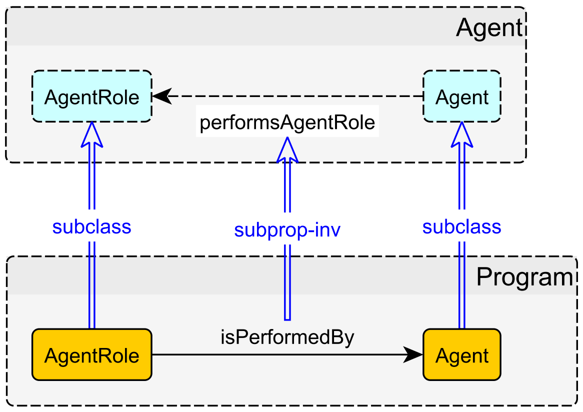

Figure 11.2: The Program pattern aligned to Agent pattern

```
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
ObjectProperty: ecglag:performsAgentRole
Class: ecglpg:Agent
Class: ecglpg:AgentRole
Class: ecglag:Agent
Class: ecglag:AgentRole
```

$$\text{ecglpg:Agent} \sqsubseteq \text{ecglag:Agent} \tag{11.20}$$

$$\text{ecglpg:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{11.21}$$

$$\text{ecglpg:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{11.22}$$

11.3.2 Alignment with Agent Role pattern

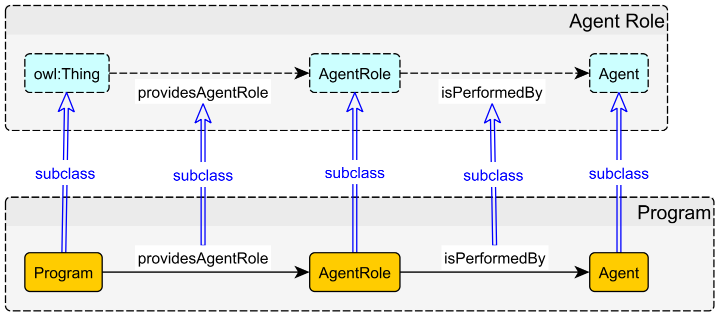

Figure 11.3: The Program pattern aligned to Agent Role pattern

```
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
ObjectProperty: ecglpg:providesAgentRole
ObjectProperty: ecglar:isPerformedBy
ObjectProperty: ecglar:providesAgentRole
Class: ecglpg:Agent
Class: ecglpg:AgentRole
Class: ecglar:Agent
Class: ecglar:AgentRole
```

$$\text{ecglpg:Agent} \sqsubseteq \text{ecglar:Agent} \tag{11.23}$$
$$\text{ecglpg:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{11.24}$$
$$\text{ecglpg:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{11.25}$$
$$\text{ecglpg:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{11.26}$$

11.3.3 Alignment with Information Object pattern

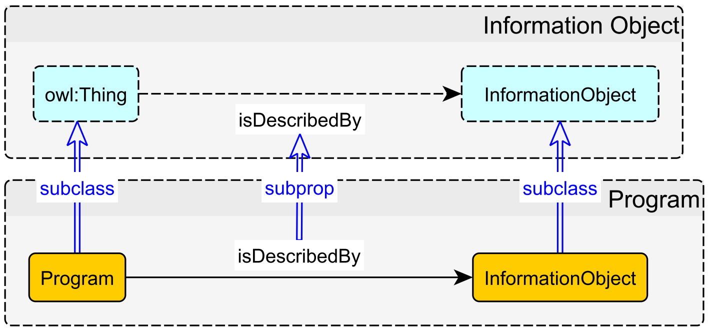

Figure 11.4: The Program pattern aligned to Information Object pattern

```
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
ObjectProperty: ecglio:isDescribedBy
Class: ecglpg:InformationObject
Class: ecglio:InformationObject
```

$$\text{ecglpg:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{11.27}$$
$$\text{ecglpg:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{11.28}$$

11.3.4 Alignment with OWL Time

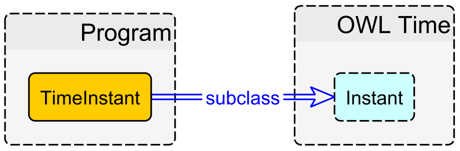

Figure 11.5: The Program pattern aligned to OWL Time

```
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
Class: time:Instant
```

$$\mathsf{ecglpg{:}TimeInstant} \sqsubseteq \mathsf{time{:}Instant} \tag{11.29}$$

12 Place

12.1 Description

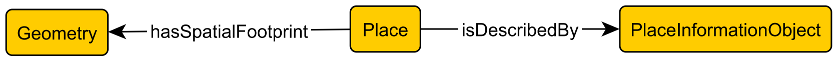

Figure 12.1: The Place pattern stub

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].

12.2 Axiomatization

12.2.1 IRI Declaration

12.2.2 Core Axioms

$$\mathsf{Place} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{12.1}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\mathsf{Place}) \tag{12.2}$$

$$\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Place} \tag{12.3}$$

$$\mathsf{Place} \sqsubseteq \forall \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{12.4}$$

$$\exists \mathsf{hasSpatialFootprint}.\mathsf{Geometry} \sqsubseteq \mathsf{Place} \tag{12.5}$$

$$\mathsf{Place} \sqsubseteq \forall \mathsf{hasSpatialFootprint}.\mathsf{Geometry} \tag{12.6}$$

12.3 Alignment

12.3.1 Alignment 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
ObjectProperty: ecglio:isDescribedBy
Class: ecglpl:InformationObject
Class: ecglio:InformationObject
```

[1]http://www.opengeospatial.org/standards/geosparql

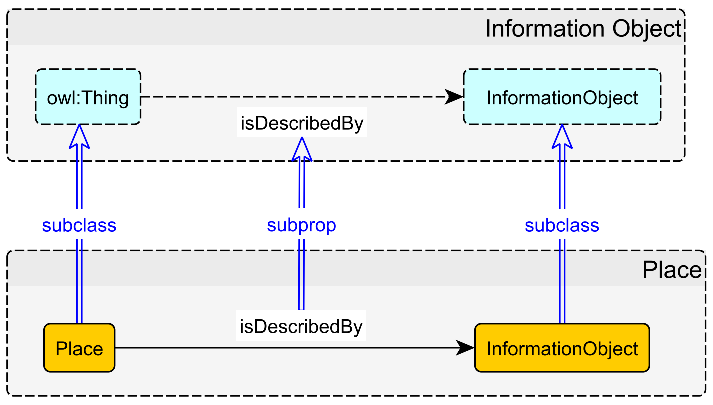

Figure 12.2: The Place pattern stub aligned with Information Object pattern

$$\text{ecglpl:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{12.7}$$
$$\text{ecglpl:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{12.8}$$

12.3.2 Alignment with GeoSPARQL Ontology

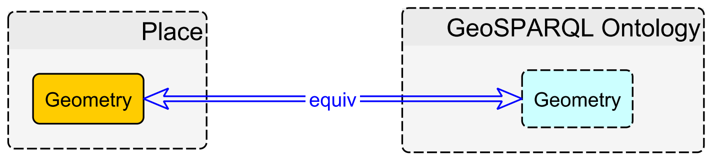

Figure 12.3: The Place pattern stub aligned with GeoSPARQL ontology

Note: We are not entirely sure with using class equivalence for aligning Geometry with GeoSPARQL's Geometry class.

```
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
Class: geosparql:Geometry
```

$$\text{ecglpl:Geometry} \equiv \text{geosparql:Geometry} \tag{12.9}$$

13 Cruise

13.1 Description

13.1.1 Overview

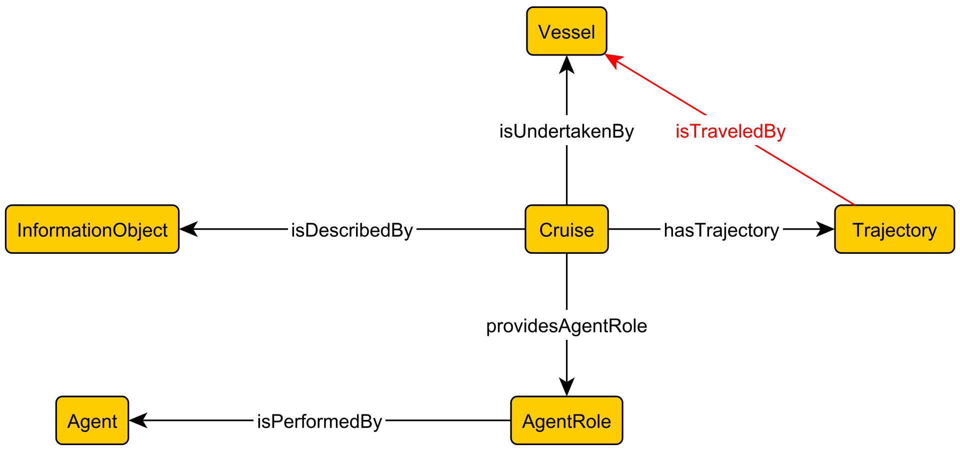

Figure 13.1: The Cruise micro-ontology overview

The Cruise micro-ontology describes ocean science cruises. Intuitively, the notion of ocean science 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, rather than developing such a pattern, we opt to consider Cruise as a micro-ontology obtained by combining and reusing existing patterns. This is done through established modeling practices while keeping the amount of abstract ontological commitments to a minimum. An earlier version of this chapter with a bit more detail can be found in a separate technical report [Krisnadhi et al., 2014].

Figure 13.1 illustrates a general overview of the Cruise micro-ontology. As seen in the figure, the specification of the Cruise micro-ontology can be dissected into three parts. First, a cruise is an abstraction of a path/route undertaken by a vessel, which is a physical object, through space within a certain time duration. This is captured by modeling a cruise to have exactly one *trajectory*, consisting of spatiotemporal segments traversed by a vessel. Second, a cruise can be understood as an *event*, which provides role performed by agents, pertaining activities conducted in/during a cruise. Our modeling of event through the Event pattern in Chapter 3 views events as something that not only provide roles to agents, but also occurs within some spatiotemporal boundary. Consequently, the first and second parts mentioned earlier can be together seen as a kind of complicated event. In fact, both the first and second parts motivate the alignment with the Agent pattern of Chapter 1, Agent Role pattern of Chapter 2, and obviously, the Event pattern. Alignment with the Event pattern is in particular rather complicated since the spatiotemporal information of a cruise is implicitly given by its associated trajectory. Trajectory for a cruise and modeling a cruise as an event is elaborated in more detail in Section 13.1.2.

Next, for the third part, we understand that a cruise may have other, non-spatiotemporal information relevant to it. For instance, information about funding awards that fund it, or programs associated with it. This is modeled by reusing the Information Object pattern from Chapter 4. Note that every information object can be realized by a number of information realization, which includes digital objects (papers, reports, etc.) about the cruise. All the above modeling choices allows us to distinguish a cruise from both the information object that describes it and the physical object, i.e., vessel, by which it is undertaken.

In using this Cruise micro-ontology, one at least needs to specify different types of agent-roles relevant for populating cruise with data. These types of agent-roles, however, should not be considered part of the core modeling of Cruise because their specification essentially does not change the meaning of "cruise" according to Figure 13.1 above. Intuitively, these types of agent-roles could be better understood as part of *nomenclature* that data providers use. Nomenclature would be more easily changed. For agent-role types, this implies that we can add as many new agent-role types as we want in the future, and since these addition should not change the meaning of "cruise", it makese sense to have them separate from the core part of Cruise micro-ontology. We will discuss this in Section 13.2.

13.1.2 Cruise as Event: Trajectory and Agent Roles for Cruise

We model a cruise as a type of event. The spatiotemporal dimension of cruise as an event is represented by its trajectory as given by Figure 13.2, while the involvement of agent in a cruise is represented by agent-roles that is performed by agents, e.g., humans or organizations. More specifically, a cruise may provide an agent role, which is performed by an agent. This is represented by the AgentRole and Agent class with specific nomenclature for agent-roles are explained in Section 13.2. These entities are aligned to the analogous entities in the Event pattern from Chapter 3, the Agent Role pattern from Chapter 2, and the Agent pattern from Chapter 1. In addition, the alignment of the Cruise micro-ontology with the Event pattern assert that the Cruise class is as a subclass of the Event class.

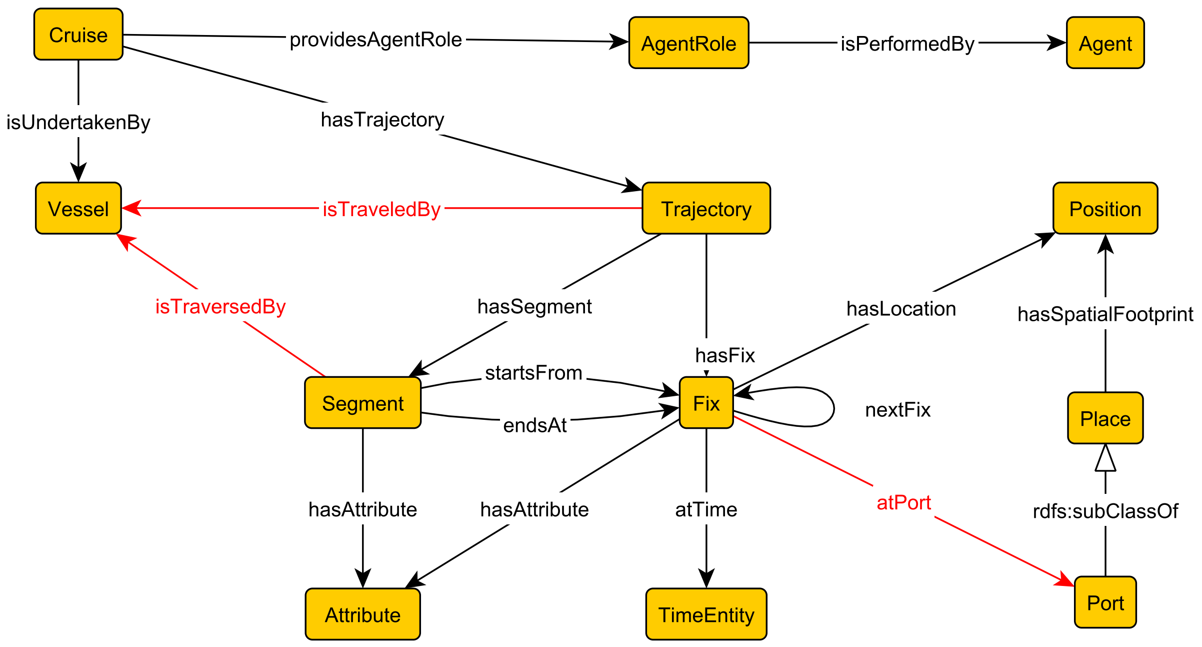

Figure 13.2: The Cruise as Events: Trajectory and Agent Roles

Cruise Trajectory

We assert that each cruise always has exactly one trajectory. 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 [Hu et al., 2013][1] to model cruise trajectory. The multi-granularity of that pattern accommodates the 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 and landmarks, 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 segment is 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 micro-ontology, 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 Figure 13.2 and imply that we cannot reuse the whole OWL axiomatization from [Hu et al., 2013] for our needs here. We thus present in Section 13.3 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 [Hu et al., 2013].

Note from Figure 13.2 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 and necessitate a rather complicated alignment with the Event pattern.

We now explain Figure 13.2 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.

[1]The OWL implementation of the Semantic Trajectory pattern is unfortunately not available online, hence we do not explicitly model the alignment to it in the OWL implementation of Cruise.

13.1.3 Cruise Information Object

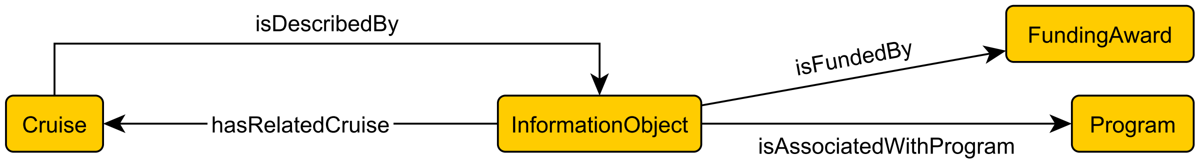

Figure 13.3: Cruise Information Object

Following Figure 13.3, we associate each instance of Cruise with exactly one InformationObject, which is aligned to the analogous class in the Information Object pattern from Chapter 4. This InformationObject acts as a proxy through which we describe various information about the cruise not covered by having the cruise as an event. In addition to the data properties inherited from the Information Object pattern, which includes identifier, description, webpage, etc., an InformationObject of a cruise may have information regarding funding award, program, as well as other cruises related to the cruise described by this instance of InformationObject.

13.2 Specific Nomenclature for Cruise

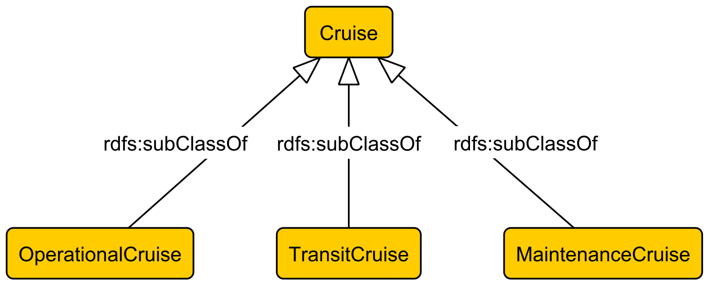

Figure 13.4: Cruise types

Nomenclature for cruise includes types of cruises, types of agent-roles for cruises, and attributes for fixes in the trajectory of a cruise. In the current version, cruise types are given as subclass of Cruise in Figure 13.4. Agent roles for cruises are given in Figure 13.5, while attributes of the fixes of the cruise are given in Figure 13.6. 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.

13.3 Axiomatization

13.3.1 IRI Declaration

Prefix: : <http://schema.geolink.org/dev/cruise#>

Ontology: <http://schema.geolink.org/dev/cruise>

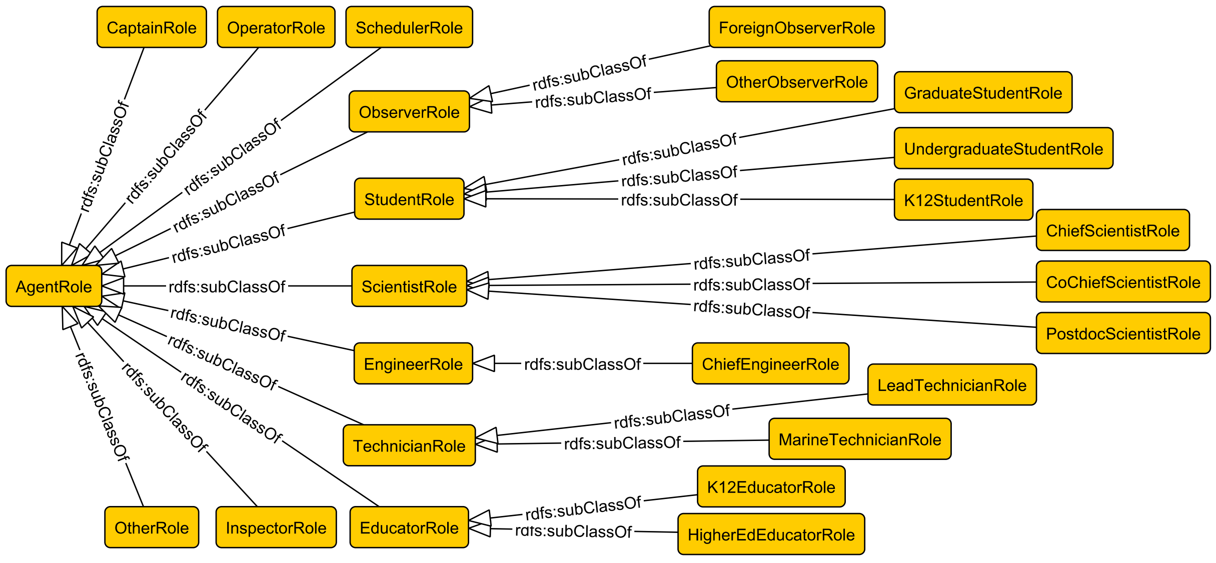

Figure 13.5: Types of agent-role for a cruise

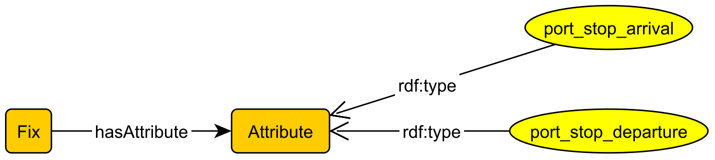

Figure 13.6: Attributes for fixes in a cruise trajectory

```
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
ObjectProperty: isTraversedBy
ObjectProperty: hasFix
ObjectProperty: hasRelatedCruise
ObjectProperty: isAssociatedWithProgram
ObjectProperty: atPort
ObjectProperty: hasSpatialFootprint
ObjectProperty: isTraveledBy
ObjectProperty: hasTrajectory
ObjectProperty: atTime
ObjectProperty: isUndertakenBy
ObjectProperty: nextFix
```

ObjectProperty: providesAgentRole
ObjectProperty: isDescribedBy
ObjectProperty: isFundedBy
ObjectProperty: rollifiedPort
ObjectProperty: hasSegment
ObjectProperty: hasAttribute
ObjectProperty: endsAt
ObjectProperty: isPerformedBy
ObjectProperty: hasLocation

Class: Port
Class: Vessel
Class: EndingFix
Class: Segment
Class: Place
Class: FundingAward
Class: Fix
Class: InformationObject
Class: Program
Class: AgentRole
Class: Position
Class: Trajectory
Class: Attribute
Class: TimeEntity
Class: Agent
Class: Cruise
Class: StartingFix

13.3.2 Core Axioms

We assert that a cruise is an event through the alignment with Event pattern (see Section 13.4.3). 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.

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{hasTrajectory}.\mathsf{Trajectory}) \tag{13.1}$$

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{isUndertakenBy}.\mathsf{Vessel}) \tag{13.2}$$

$$\mathsf{Cruise} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{13.3}$$

$$\mathsf{InformationObject} \sqsubseteq (=1\ \mathsf{isDescribedBy}^{-}.\mathsf{Cruise}) \tag{13.4}$$

$$\mathsf{hasTrajectory}^{-} \circ \mathsf{isUndertakenBy} \sqsubseteq \mathsf{isTraveledBy} \tag{13.5}$$

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.

$$\exists \mathsf{hasTrajectory}.\mathsf{Trajectory} \sqsubseteq \mathsf{Cruise} \tag{13.6}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{hasTrajectory}.\mathsf{Trajectory} \tag{13.7}$$

$$\exists \mathsf{isUndertakenBy}.\mathsf{Vessel} \sqsubseteq \mathsf{Cruise} \tag{13.8}$$

$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{isUndertakenBy}.\mathsf{Vessel} \tag{13.9}$$

$$\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Cruise} \tag{13.10}$$

$$\text{Cruise} \sqsubseteq \forall \text{isDescribedBy.InformationObject} \tag{13.11}$$

$$\exists \text{isTraveledBy.Vessel} \sqsubseteq \text{Trajectory} \tag{13.12}$$

$$\text{Trajectory} \sqsubseteq \forall \text{isTraveledBy.Vessel} \tag{13.13}$$

13.3.2.1 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 [Hu et al., 2013]. 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.

$$\text{Fix} \sqsubseteq \exists \text{hasLocation.Position} \sqcap \exists \text{atTime.TimeEntity} \sqcap (=1\ \text{hasFix}^{-}.\text{Trajectory}) \tag{13.14}$$

$$\text{Fix} \sqsubseteq (\leqslant 1\ \text{nextFix.Fix}) \sqcap \neg \exists \text{nextFix.Self} \tag{13.15}$$

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

$$\text{StartingFix} \equiv \text{Fix} \sqcap \neg \exists \text{nextFix}^{-}.\top \tag{13.16}$$

$$\text{EndingFix} \equiv \text{Fix} \sqcap \neg \exists \text{nextFix}.\top \tag{13.17}$$

$$\text{StartingFix} \sqcap \text{EndingFix} \sqsubseteq \bot \tag{13.18}$$

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.

$$\text{Trajectory} \sqsubseteq \exists \text{hasFix}.(\text{StartingFix} \sqcap \exists \text{nextFix.Fix}) \tag{13.19}$$

$$\text{hasFix} \circ \text{nextFix} \sqsubseteq \text{hasFix} \tag{13.20}$$

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.

$$\text{Segment} \sqsubseteq (=1\ \text{startsFrom.Fix}) \tag{13.21}$$

$$\exists \text{nextFix.Fix} \sqsubseteq (=1\ \text{startsFrom}^{-}.\text{Segment}) \tag{13.22}$$

$$\text{hasFix} \circ \text{startsFrom}^{-} \sqsubseteq \text{hasSegment} \tag{13.23}$$

$$\text{startsFrom} \circ \text{nextFix} \sqsubseteq \text{endsAt} \tag{13.24}$$

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 $\text{Fix}(f_1), \ldots, \text{Fix}(f_n), \text{nextFix}(f_1, f_2), \ldots, \text{nextFix}(f_{n-1}, f_n), \text{StartingFix}(f_1), \text{EndingFix}(f_n)$. 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: $\text{hasLocation}(x, y), \text{hasSpatialFootprint}(z, y), \text{Port}(z) \rightarrow \text{atPort}(x, z)$. 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.

$$\text{hasLocation} \circ \text{hasSpatialFootprint}^{-} \circ \text{rollifiedPort} \sqsubseteq \text{atPort} \tag{13.25}$$

$$\exists \text{rollifiedPort.Self} \equiv \text{Port} \tag{13.26}$$

$$\text{Port} \sqsubseteq \text{Place} \tag{13.27}$$

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

$$\mathsf{hasSegment}^{-} \circ \mathsf{isTraveledBy} \sqsubseteq \mathsf{isTraversedBy} \tag{13.28}$$

We assert the following guarded domain and range restrictions.

$$\exists \mathsf{hasFix}.\mathsf{Fix} \sqsubseteq \mathsf{Trajectory} \tag{13.29}$$
$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasFix}.\mathsf{Fix} \tag{13.30}$$
$$\exists \mathsf{nextFix}.\mathsf{Fix} \sqsubseteq \mathsf{Fix} \tag{13.31}$$
$$\mathsf{Fix} \sqsubseteq \forall \mathsf{nextFix}.\mathsf{Fix} \tag{13.32}$$
$$\exists \mathsf{hasLocation}.\mathsf{Position} \sqsubseteq \mathsf{Fix} \tag{13.33}$$
$$\mathsf{Fix} \sqsubseteq \forall \mathsf{hasLocation}.\mathsf{Position} \tag{13.34}$$
$$\exists \mathsf{atPort}.\mathsf{Port} \sqsubseteq \mathsf{Fix} \tag{13.35}$$
$$\mathsf{Fix} \sqsubseteq \forall \mathsf{atPort}.\mathsf{Port} \tag{13.36}$$
$$\exists \mathsf{atTime}.\mathsf{TimeEntity} \sqsubseteq \mathsf{Fix} \tag{13.37}$$
$$\mathsf{Fix} \sqsubseteq \forall \mathsf{atTime}.\mathsf{TimeEntity} \tag{13.38}$$
$$\exists \mathsf{hasSpatialFootprint}.\mathsf{Position} \sqsubseteq \mathsf{Place} \tag{13.39}$$
$$\mathsf{Place} \sqsubseteq \forall \mathsf{hasSpatialFootprint}.\mathsf{Position} \tag{13.40}$$
$$\exists \mathsf{hasSegment}.\mathsf{Segment} \sqsubseteq \mathsf{Trajectory} \tag{13.41}$$
$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasSegment}.\mathsf{Segment} \tag{13.42}$$
$$\exists \mathsf{startsFrom}.\mathsf{Fix} \sqsubseteq \mathsf{Segment} \tag{13.43}$$
$$\mathsf{Segment} \sqsubseteq \forall \mathsf{startsFrom}.\mathsf{Fix} \tag{13.44}$$
$$\exists \mathsf{endsAt}.\mathsf{Fix} \sqsubseteq \mathsf{Segment} \tag{13.45}$$
$$\mathsf{Segment} \sqsubseteq \forall \mathsf{endsAt}.\mathsf{Fix} \tag{13.46}$$
$$\exists \mathsf{isTraversedBy}.\mathsf{Vessel} \sqsubseteq \mathsf{Segment} \tag{13.47}$$
$$\mathsf{Segment} \sqsubseteq \forall \mathsf{isTraversedBy}.\mathsf{Vessel} \tag{13.48}$$
$$\exists \mathsf{hasAttribute}.\mathsf{Attribute} \sqsubseteq \mathsf{Segment} \sqcup \mathsf{Fix} \tag{13.49}$$
$$\mathsf{Fix} \sqsubseteq \forall \mathsf{hasAttribute}.\mathsf{Attribute} \tag{13.50}$$
$$\mathsf{Segment} \sqsubseteq \forall \mathsf{hasAttribute}.\mathsf{Attribute} \tag{13.51}$$

13.3.2.2 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.

$$\exists \mathsf{providesAgentRole}.\mathsf{AgentRole} \sqsubseteq \mathsf{Cruise} \tag{13.52}$$
$$\mathsf{Cruise} \sqsubseteq \forall \mathsf{providesAgentRole}.\mathsf{AgentRole} \tag{13.53}$$
$$\exists \mathsf{isPerformedBy}.\mathsf{Agent} \sqsubseteq \mathsf{AgentRole} \tag{13.54}$$
$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{isPerformedBy}.\mathsf{Agent} \tag{13.55}$$
$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy}.\mathsf{Agent}) \tag{13.56}$$

13.3.2.3 Axioms for Cruise Information Object

A cruise may be funded by some funding award, associated with some program, and related to some other cruise. These information are attached to the information object associated with a cruise. We thus assert the domain and range restrictions for the properties representing the aforementioned information. Note that domain and range restrictions for isDescribedBy have been asserted earlier in the beginning of this section.

$$\exists\mathsf{hasRelatedCruiseCruise} \sqsubseteq \mathsf{InformationObject} \tag{13.57}$$
$$\mathsf{InformationObject} \sqsubseteq \forall\mathsf{hasRelatedCruise.Cruise} \tag{13.58}$$
$$\exists\mathsf{isAssociatedWithProgram.Program} \sqsubseteq \mathsf{InformationObject} \tag{13.59}$$
$$\mathsf{InformationObject} \sqsubseteq \forall\mathsf{isAssociatedWithProgram.Program} \tag{13.60}$$
$$\exists\mathsf{isFundedBy.FundingAward} \sqsubseteq \mathsf{InformationObject} \tag{13.61}$$
$$\mathsf{InformationObject} \sqsubseteq \forall\mathsf{isFundedBy.FundingAward} \tag{13.62}$$

13.3.2.4 Class Disjointness Axioms

We assert the following class disjointness axioms:

$$\begin{aligned}&\mathsf{alldisjoint}(\mathsf{Cruise}, \mathsf{InformationObject}, \mathsf{AgentRole}, \mathsf{Agent}, \mathsf{FundingAward}, \mathsf{Program}, \mathsf{Trajectory}, \mathsf{Vessel},\\ &\qquad \mathsf{Fix}, \mathsf{Segment}, \mathsf{Attribute}, \mathsf{TimeEntity}, \mathsf{Place}, \mathsf{Position})\end{aligned} \tag{13.63}$$

13.3.2.5 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:

$$\mathsf{OperationalCruise} \sqcup \mathsf{MaintenanceCruise} \sqcup \mathsf{TransitCruise} \sqsubseteq \mathsf{Cruise} \tag{13.64}$$

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.

$$\begin{aligned}\mathsf{OperationalCruise} \equiv \mathsf{Cruise} &\sqcap \exists\mathsf{providesAgentRole.ChiefScientistRole}\\ &\sqcap \exists\mathsf{isDescribedBy}.\exists\mathsf{isFundedBy.FundingAward}\end{aligned} \tag{13.65}$$

The nomenclature part also establishes a predefined set of cruise agent-role types.

$$\mathsf{CaptainRole} \sqcup \mathsf{OperatorRole} \sqcup \mathsf{SchedulerRole} \sqcup \mathsf{ObserverRole} \sqcup \mathsf{InspectorRole} \sqsubseteq \mathsf{AgentRole} \tag{13.66}$$
$$\mathsf{ForeignObserverRole} \sqcup \mathsf{OtherObserverRole} \sqsubseteq \mathsf{ObserverRole} \tag{13.67}$$
$$\mathsf{EngineerRole} \sqcup \mathsf{ScientistRole} \sqcup \mathsf{TechnicianRole} \sqcup \mathsf{StudentRole} \sqcup \mathsf{EducatorRole} \sqsubseteq \mathsf{AgentRole} \tag{13.68}$$
$$\mathsf{ChiefEngineerRole} \sqsubseteq \mathsf{EngineerRole} \tag{13.69}$$
$$\mathsf{ChiefScientistRole} \sqcup \mathsf{CoChiefScientistRole} \sqcup \mathsf{PostdocScientistRole} \sqsubseteq \mathsf{ScientistRole} \tag{13.70}$$
$$\mathsf{LeadTechnicianRole} \sqcup \mathsf{MarineTechnicianRole} \sqsubseteq \mathsf{TechnicianRole} \tag{13.71}$$
$$\mathsf{GraduateStudentRole} \sqcup \mathsf{UndergraduateStudentRole} \sqcup \mathsf{K12StudentRole} \sqsubseteq \mathsf{StudentRole} \tag{13.72}$$
$$\mathsf{HigherEdEducatorRole} \sqcup \mathsf{K12EducatorRole} \sqsubseteq \mathsf{EducatorRole} \tag{13.73}$$
$$\mathsf{OtherRole} \sqsubseteq \mathsf{AgentRole} \tag{13.74}$$

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.

$$\mathsf{Attribute}(\mathsf{port\_stop\_departure}), \mathsf{Attribute}(\mathsf{port\_stop\_arrival}) \tag{13.75}$$

13.4 Alignment

13.4.1 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
ObjectProperty: ecglag:performsAgentRole

Class: ecglcr:Agent
Class: ecglcr:AgentRole
Class: ecglag:Agent
Class: ecglag:AgentRole

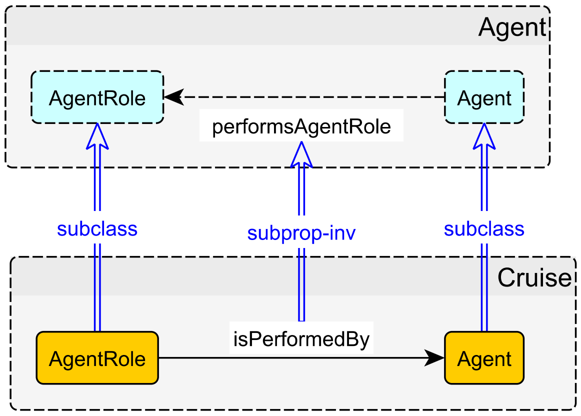

Figure 13.7: The Cruise micro-ontology alignment with Agent pattern

$$\text{ecglcr:Agent} \sqsubseteq \text{ecglag:Agent} \tag{13.76}$$

$$\text{ecglcr:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{13.77}$$

$$\text{ecglcr:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{13.78}$$

13.4.2 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
ObjectProperty: ecglcr:isPerformedBy
ObjectProperty: ecglar:providesAgentRole
ObjectProperty: ecglar:isPerformedBy

Class: ecglcr:AgentRole
Class: ecglcr:Agent
Class: ecglar:AgentRole
Class: ecglar:Agent

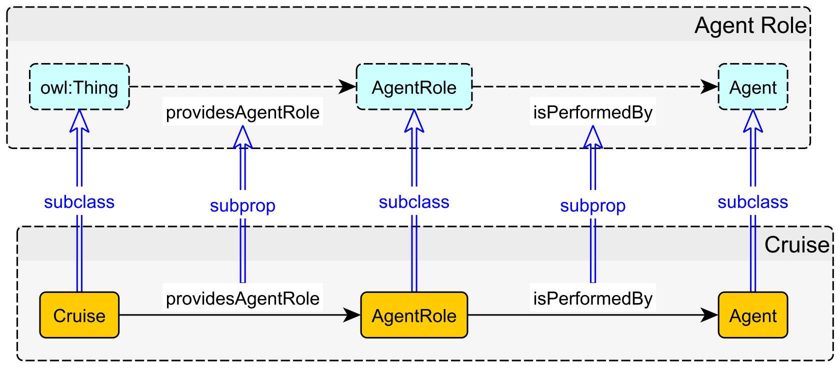

Figure 13.8: The Cruise micro-ontology alignment with Agent Role pattern

$$\text{ecglcr:Agent} \sqsubseteq \text{ecglar:Agent} \tag{13.79}$$

$$\text{ecglcr:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{13.80}$$

$$\text{ecglcr:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{13.81}$$

$$\text{ecglcr:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{13.82}$$

13.4.3 Alignment with Event pattern

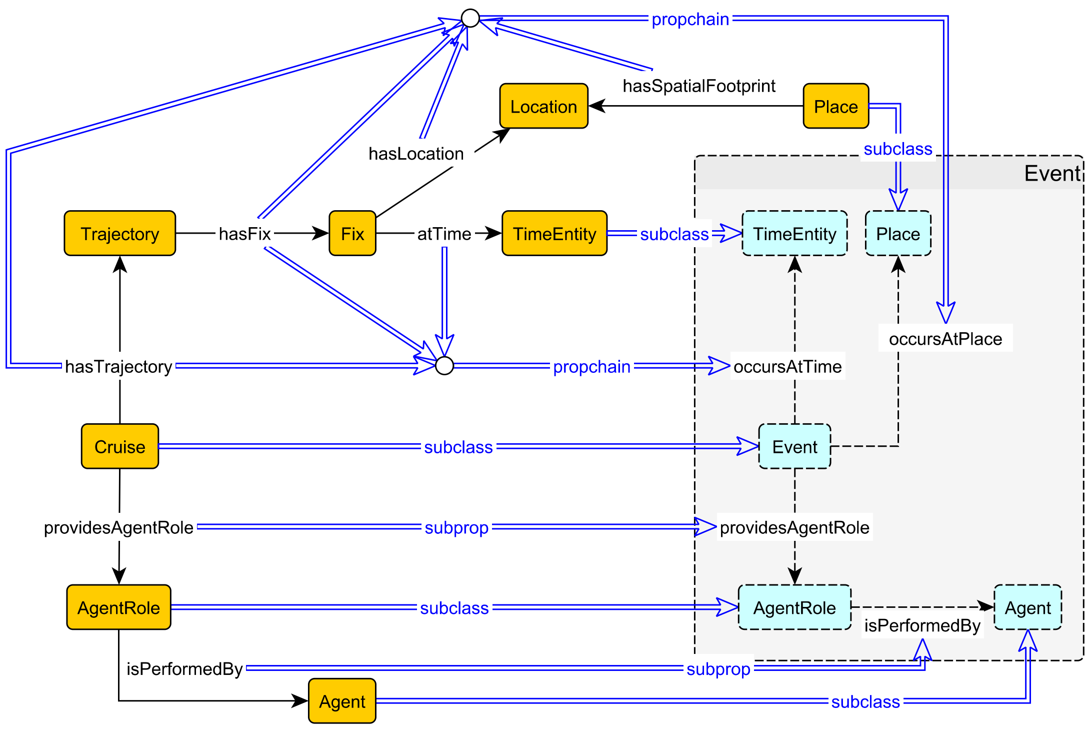

Figure 13.9: The Cruise micro-ontology alignment with Event pattern

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
ObjectProperty: ecglcr:hasSpatialFootprint
ObjectProperty: ecglcr:atPort
ObjectProperty: ecglcr:atTime
ObjectProperty: ecglcr:hasFix
ObjectProperty: ecglcr:hasTrajectory
ObjectProperty: ecglcr:providesAgentRole
ObjectProperty: ecglcr:isPerformedBy

ObjectProperty: ecglev:occursAtPlace
ObjectProperty: ecglev:occursAtTime
ObjectProperty: ecglev:providesAgentRole
ObjectProperty: ecglev:isPerformedBy

Class: ecglcr:AgentRole
Class: ecglcr:Agent
Class: ecglev:AgentRole
Class: ecglev:Agent
```

$$\text{ecglcr:Cruise} \sqsubseteq \text{ecglev:Event} \tag{13.83}$$

$$\text{ecglcr:Port} \sqsubseteq \text{ecglev:Place} \tag{13.84}$$

$$\text{ecglcr:TimeEntity} \sqsubseteq \text{ecglev:TimeEntity} \tag{13.85}$$

$$\text{ecglcr:AgentRole} \sqsubseteq \text{ecglev:AgentRole} \tag{13.86}$$

$$\text{ecglcr:Agent} \sqsubseteq \text{ecglev:Agent} \tag{13.87}$$

$$\text{ecglcr:hasTrajectory} \circ \text{ecglcr:hasFix} \circ \text{ecglcr:hasLocation} \circ \text{ecglcr:hasSpatialFootprint}^{-} \sqsubseteq \text{ecglev:occursAtPlace} \tag{13.88}$$

$$\text{ecglcr:hasTrajectory} \circ \text{ecglcr:hasFix} \circ \text{ecglcr:atTime} \sqsubseteq \text{ecglev:occursAtTime} \tag{13.89}$$

$$\text{ecglcr:providesAgentRole} \sqsubseteq \text{ecglev:providesAgentRole} \tag{13.90}$$

$$\text{ecglcr:isPerformedBy} \sqsubseteq \text{ecglev:isPerformedBy} \tag{13.91}$$

13.4.4 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
ObjectProperty: ecglfa:isFundedBy

Class: ecglcr:FundingAward
Class: ecglfa:FundingAward
```

$$\text{ecglcr:FundingAward} \sqsubseteq \text{ecglfa:FundingAward} \tag{13.92}$$

$$\text{ecglcr:isFundedBy} \sqsubseteq \text{ecglfa:isFundedBy} \tag{13.93}$$

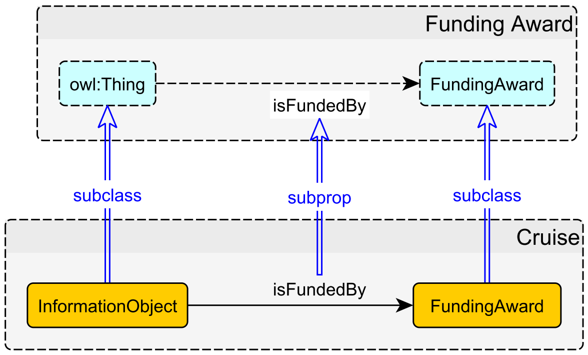

Figure 13.10: The Cruise micro-ontology alignment with Funding Award pattern

13.4.5 Alignment with Information Object pattern

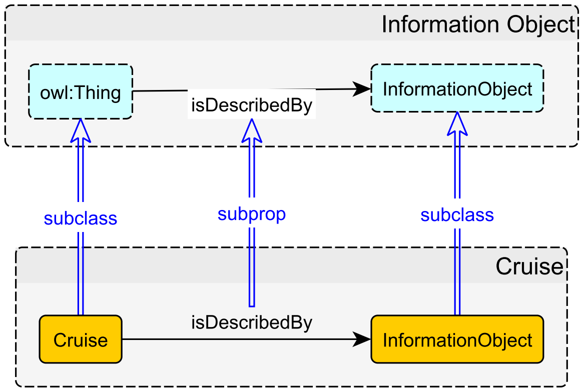

Figure 13.11: The Cruise micro-ontology alignment with Information Object pattern

```
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
ObjectProperty: ecglio:isDescribedBy

Class: ecglcr:InformationObject
Class: ecglio:InformationObject
```

$$\text{ecglcr:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{13.94}$$
$$\text{ecglcr:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{13.95}$$

13.4.6 Alignment with OWL Time ontology

```
Prefix: time: <http://www.w3.org/2006/time#>
```

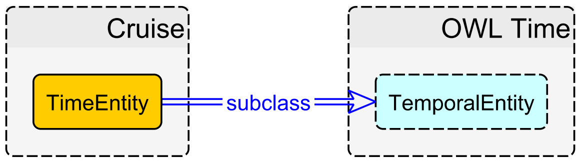

Figure 13.12: The Cruise micro-ontology alignment with OWL Time ontology

```
Prefix: ecglcr: <http://schema.geolink.org/dev/cruise#>

Ontology: <http://schema.geolink.org/dev/cruise-to-owltime>


Class: ecglcr:TimeEntity
Class: time:TemporalEntity
```

$$\text{ecglcr:TimeEntity} \sqsubseteq \text{time:TemporalEntity} \tag{13.96}$$

13.4.7 Alignment with Place pattern

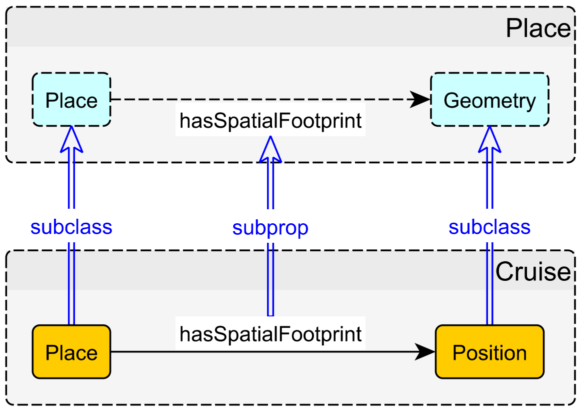

Figure 13.13: The Cruise micro-ontology alignment with Place pattern

```
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
ObjectProperty: ecglpl:hasSpatialFootprint

Class: ecglcr:Place
Class: ecglcr:Position
Class: ecglpl:Place
Class: ecglpl:Geometry
```

$$\text{ecglcr:Place} \sqsubseteq \text{ecglpl:Place} \tag{13.97}$$
$$\text{ecglcr:Position} \sqsubseteq \text{ecglpl:Geometry} \tag{13.98}$$
$$\text{ecglcr:hasSpatialFootprint} \sqsubseteq \text{ecglpl:hasSpatialFootprint} \tag{13.99}$$

13.4.8 Alignment with Program pattern

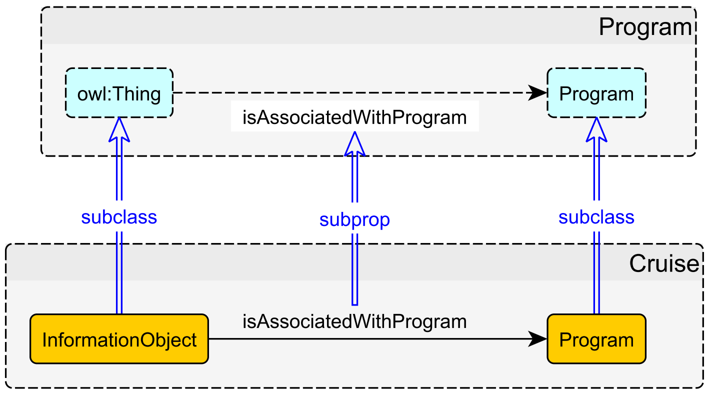

Figure 13.14: The Cruise micro-ontology alignment with Program pattern

```
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:isAssociatedWithProgram
ObjectProperty: ecglpg:isAssociatedWithProgram

Class: ecglcr:Program
Class: ecglpg:Program
```

$$\text{ecglcr:Program} \sqsubseteq \text{ecglpg:Program} \tag{13.100}$$
$$\text{ecglcr:isAssociatedWithProgram} \sqsubseteq \text{ecglpg:isAssociatedWithProgram} \tag{13.101}$$

13.4.9 Alignment with Vessel pattern

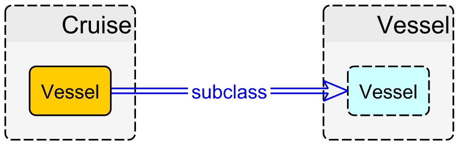

Figure 13.15: The Cruise micro-ontology alignment with Vessel pattern

```
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
Class: ecglvs:Vessel
```

$$\mathsf{ecglcr{:}Vessel} \sqsubseteq \mathsf{ecglvs{:}Vessel} \tag{13.102}$$

14 Platform Pattern Stub

14.1 Description

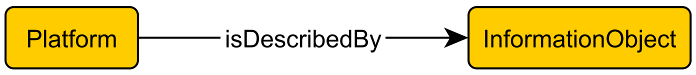

Figure 14.1: Platform pattern

This is a pattern stub, depicted in Figure 14.1, for representing ocean science platforms, which include vessels, submersibles, aircrafts, etc.

14.2 Specific Nomenclature for Platform pattern

Nomenclature for Platform pattern mainly consists of types of platform. Platform types are currently taken from the list maintained by SeaVoX L06 platform categories[1]. Other platform types are of course possible in the future. Table 14.1 contains all SeaVoX L06 platform categories, each is given as a pair of key and term.

14.3 Axiomatization

14.3.1 IRI Declarations

14.3.2 Core Axioms

$$\mathsf{Platform} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{14.1}$$

$$\exists \mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Platform} \tag{14.2}$$

$$\mathsf{Platform} \sqsubseteq \forall \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{14.3}$$

Class disjointness axioms:

$$\mathsf{Platform} \sqcap \mathsf{InformationObject} \sqsubseteq \bot \tag{14.4}$$

14.3.3 Axioms for Platform Nomenclature

Axioms for Platform pattern's nomenclature part includes a class hierarchy based on the platform types in Table 14.1.

14.4 Alignment

14.4.1 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>

[1]http://seadatanet.maris2.nl/v_bodc_vocab_v2/search.asp?lib=L06

Key	Term	Key	Term
0	unknown	47	float
10	land or seafloor	48	mooring
11	fixed benthic node	49	surface ice buoy
12	sea bed vehicle	50	buoyant aircraft
13	beach/intertidal zone structure	51	free-rising balloon
14	land/onshore structure	52	free-floating balloon
15	land/onshore vehicle	53	tethered balloon
16	offshore structure	54	airship
17	coastal structure	60	non-buoyant aircraft
18	river station	61	research aeroplane
19	mesocosm bag	62	aeroplane
20	submersible	63	rocket
21	propelled manned submersible	64	geostationary orbiting satellite
22	propelled unmanned submersible	65	orbiting satellite
23	towed unmanned submersible	66	manned spacecraft
24	drifting manned submersible	67	helicopter
25	autonomous underwater vehicle	68	satellite
26	lowered unmanned submersible	69	autogyro
27	sub-surface gliders	6A	glider
30	ship	6B	kite
31	research vessel	6C	parachute
32	vessel of opportunity	6Z	spacecraft
33	self-propelled small boat	70	organism
34	vessel at fixed position	71	human
35	vessel of opportunity on fixed route	72	diver
36	fishing vessel	73	flightless bird
37	self-propelled boat	74	seabird and duck
38	man-powered boat	75	cetacean
39	naval vessel	76	fish
3A	man-powered small boat	77	land-sea mammals
3B	autonomous surface water vehicle	90	cryosphere
3C	surface gliders	91	ice island
3Z	surface vessel	92	ice shelf
41	moored surface buoy	93	pack ice
42	drifting surface float	94	drift ice
43	subsurface mooring	95	amphibious vehicle
44	drifting subsurface float	96	amphibious crawler
45	fixed subsurface vertical profiler	9A	DUKW
46	drifting subsurface profiling float	9B	hovercraft

Table 14.1: SeaVoXL06 key-term pairs

ObjectProperty: ecglpf:isDescribedBy
ObjectProperty: ecglio:isDescribedBy

Class: ecglpf:InformationObject
Class: ecglio:InformationObject

$$\mathsf{ecglpf{:}InformationObject} \sqsubseteq \mathsf{ecglio{:}InformationObject} \tag{14.5}$$

$$\mathsf{ecglpf{:}isDescribedBy} \sqsubseteq \mathsf{ecglio{:}isDescribedBy} \tag{14.6}$$

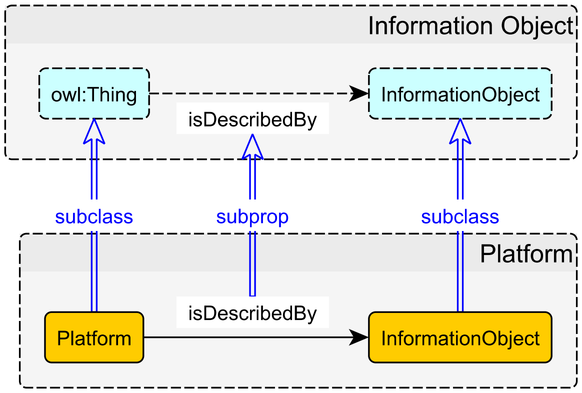

Figure 14.2: The Platform pattern alignment with Information Object pattern

15 Vessel pattern

15.1 Description

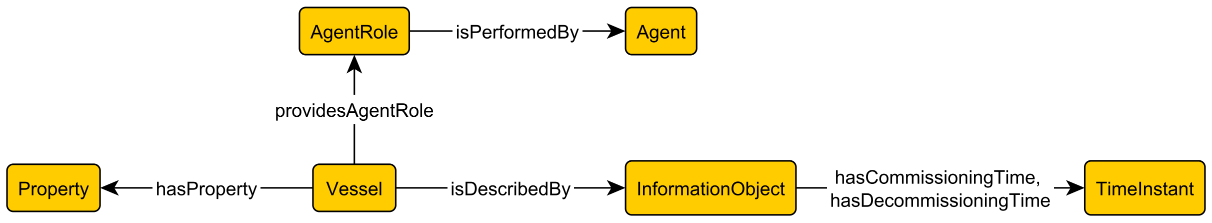

Figure 15.1: Vessel pattern

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 Chapter 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*[1].

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 decommssion, etc.

15.2 Specific Nomenclature for Vessel pattern

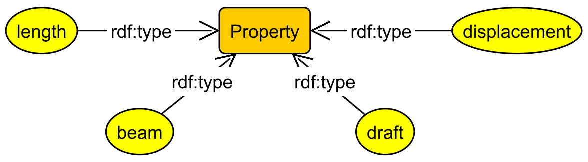

Figure 15.2: Vessel property

Nomenclature part of the Vessel pattern includes terms describing properties relevant for a vessel (Figure 15.2) and types of agent-roles provided by a vessel (Figure 15.3). 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.

[1] `http://www.qudt.org/`

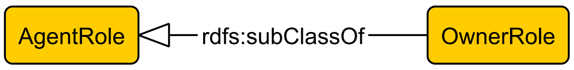

Figure 15.3: Vessel agent roles

15.3 Axiomatization

15.3.1 IRI Declarations

15.3.2 Core Axioms

Every vessel has exactly one information object. Also, we assert the guarded domain and range restrictions.

$$\mathsf{Vessel} \sqsubseteq (=1\ \mathsf{isDescribedBy}.\mathsf{InformationObject}) \tag{15.1}$$
$$\exists\mathsf{isDescribedBy}.\mathsf{InformationObject} \sqsubseteq \mathsf{Vessel} \tag{15.2}$$
$$\mathsf{Vessel} \sqsubseteq \forall\mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{15.3}$$
$$\exists\mathsf{hasProperty}.\mathsf{Property} \sqsubseteq \mathsf{Vessel} \tag{15.4}$$
$$\mathsf{Vessel} \sqsubseteq \forall\mathsf{hasProperty}.\mathsf{Property} \tag{15.5}$$
$$\exists\mathsf{providesAgentRole}.\mathsf{AgentRole} \sqsubseteq \mathsf{Vessel} \tag{15.6}$$
$$\mathsf{Vessel} \sqsubseteq \forall\mathsf{providesAgentRole}.\mathsf{AgentRole} \tag{15.7}$$
$$\exists\mathsf{isPerformedBy}.\mathsf{Agent} \sqsubseteq \mathsf{AgentRole} \tag{15.8}$$
$$\mathsf{AgentRole} \sqsubseteq \forall\mathsf{isPerformedBy}.\mathsf{Agent} \tag{15.9}$$
$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{isPerformedBy}.\mathsf{Agent}) \tag{15.10}$$

Class disjointness axiom:

$$\mathsf{alldisjoint}(\mathsf{Vessel}, \mathsf{InformationObject}, \mathsf{TimeInstant}, \mathsf{AgentRole}, \mathsf{Agent}, \mathsf{Property}) \tag{15.11}$$

15.3.3 Axioms for Vessel Nomenclature

$$\mathsf{OwnerRole} \sqsubseteq \mathsf{AgentRole} \tag{15.12}$$

$$\mathsf{Property}(\mathsf{length}), \mathsf{Property}(\mathsf{beam}), \mathsf{Property}(\mathsf{draft}), \mathsf{Property}(\mathsf{displacement}) \tag{15.13}$$

15.4 Alignment

15.4.1 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
ObjectProperty: ecglag:performsAgentRole

Class: ecglvs:Agent
Class: ecglvs:AgentRole
Class: ecglag:Agent
Class: ecglag:AgentRole
```

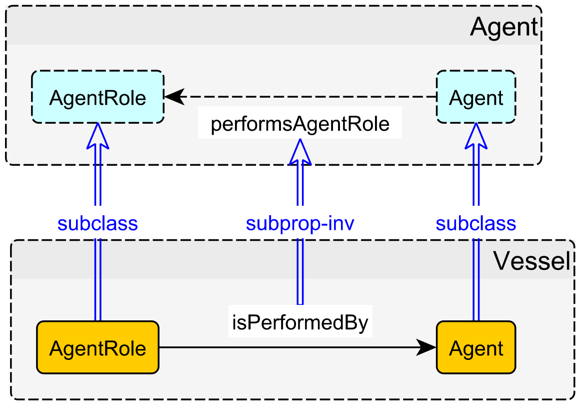

Figure 15.4: The Vessel pattern alignment with Agent pattern

$$\text{ecglvs:Agent} \sqsubseteq \text{ecglag:Agent} \tag{15.14}$$
$$\text{ecglvs:AgentRole} \sqsubseteq \text{ecglag:AgentRole} \tag{15.15}$$
$$\text{ecglvs:isPerformedBy} \sqsubseteq \text{ecglag:performsAgentRole}^{-} \tag{15.16}$$

15.4.2 Alignment with Agent Role pattern

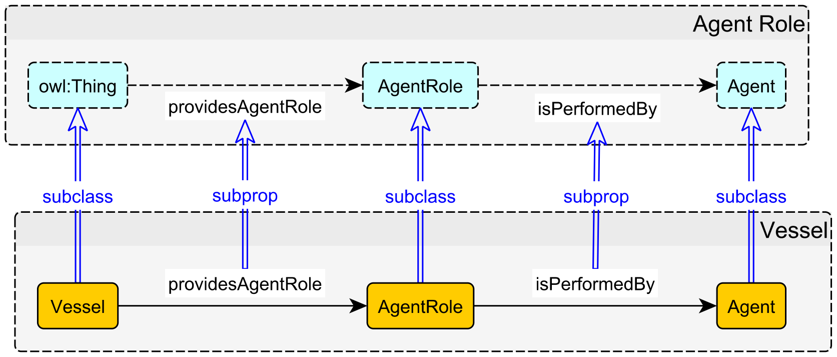

Figure 15.5: The Vessel pattern alignment with Agent Role pattern

```
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
ObjectProperty: ecglvs:isPerformedBy
ObjectProperty: ecglar:providesAgentRole
ObjectProperty: ecglar:isPerformedBy

Class: ecglvs:AgentRole
```

Class: ecglvs:Agent
Class: ecglar:AgentRole
Class: ecglar:Agent

$$\text{ecglvs:Agent} \sqsubseteq \text{ecglar:Agent} \tag{15.17}$$
$$\text{ecglvs:AgentRole} \sqsubseteq \text{ecglar:AgentRole} \tag{15.18}$$
$$\text{ecglvs:providesAgentRole} \sqsubseteq \text{ecglar:providesAgentRole} \tag{15.19}$$
$$\text{ecglvs:isPerformedBy} \sqsubseteq \text{ecglar:isPerformedBy} \tag{15.20}$$

15.4.3 Alignment with Information Object pattern

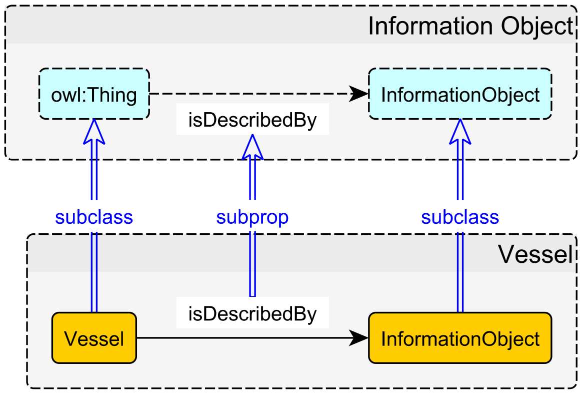

Figure 15.6: The Vessel pattern alignment with Information Object pattern

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
ObjectProperty: ecglio:isDescribedBy

Class: ecglvs:InformationObject
Class: ecglio:InformationObject

$$\text{ecglvs:InformationObject} \sqsubseteq \text{ecglio:InformationObject} \tag{15.21}$$
$$\text{ecglvs:isDescribedBy} \sqsubseteq \text{ecglio:isDescribedBy} \tag{15.22}$$

15.4.4 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>

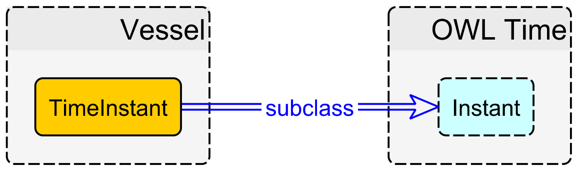

Figure 15.7: The Vessel pattern alignment with OWL Time ontology

```
Class: ecglvs:TimeEntity
Class: time:TemporalEntity
```

$$\text{ecglvs:TimeEntity} \sqsubseteq \text{time:TemporalEntity} \tag{15.23}$$

15.4.5 Alignment with Platform pattern

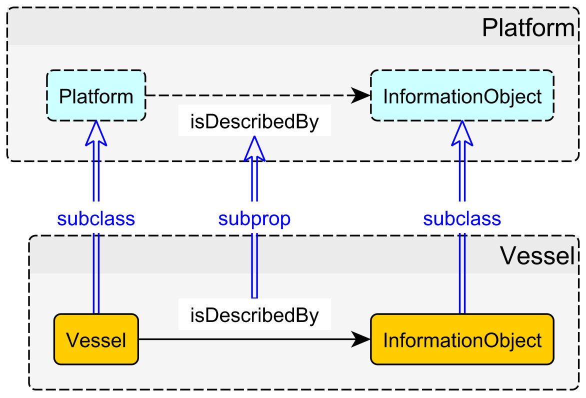

Figure 15.8: The Vessel pattern alignment with Platform pattern

```
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
ObjectProperty: ecglpf:isDescribedBy

Class: ecglvs:Vessel
Class: ecglvs:InformationObject
Class: ecglpf:Platform
Class: ecglpf:InformationObject
```

$$\text{ecglvs:Vessel} \sqsubseteq \text{ecglpf:Platform} \tag{15.24}$$

$$\text{ecglvs:InformationObject} \sqsubseteq \text{ecglpf:InformationObject} \tag{15.25}$$

$$\text{ecglvs:isDescribedBy} \sqsubseteq \text{ecglpf:isDescribedBy} \tag{15.26}$$

16 Physical Sample pattern

16.1 Description

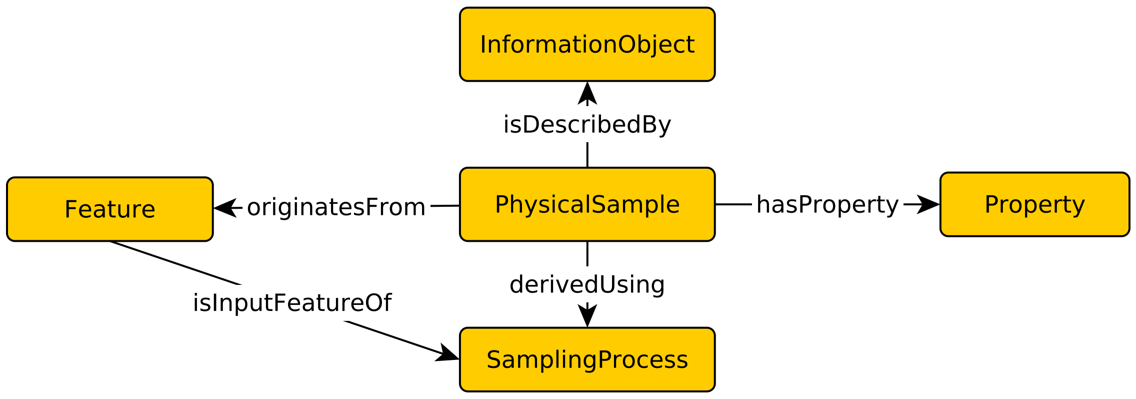

Figure 16.1: Physical Sample pattern

The **Physical Sample** pattern is designed to model data about *samples*, such as *section*, *wedge*, as well as other types of samples. By naming the pattern as *PhysicalSample*, we differentiate it from the general concept of *sample* from statistics. This pattern also models the process in which a physical sample is generated: it can be directly acquired from field work, but can also be obtained through some sampling process in a laboratory, in which a new (and possibly smaller) sample is derived from the original (and possibly bigger) sample. A PhysicalSample is derived using a SamplingProcess which should have an input Feature. In this pattern, we define the class PhysicalSample as a sub class of Feature, and as a result, a PhysicalSample can be derived from not only the original Feature but also another PhysicalSample. The SamplingProcess has been modeled as a sub class of Event which has timestamps, place, as well as agents (see the alignment to the Event pattern). A PhysicalSample is described by an InformationObject which provides additional information, such as name, ID, and so forth. A PhysicalSample also has its Property, such as material type, age, and so forth.

16.2 Axiomatization

16.2.1 IRIs Declarations

```
Prefix: ecglps: <http://schema.geolink.org/dev/physicalsample#>
Ontology: <http://schema.geolink.org/dev/physicalsample>

Import: <http://schema.geolink.org/dev/physicalsample-to-informationobject>
Import: <http://schema.geolink.org/dev/physicalsample-to-propertyvalue>
Import: <http://schema.geolink.org/dev/physicalsample-to-event>

ObjectProperty: ecglps:originatesFrom
ObjectProperty: ecglps:isInputFeatureOf
ObjectProperty: ecglps:derivedUsing
ObjectProperty: ecglps:isDescribedBy
```

```
ObjectProperty: ecglps:hasProperty

Class: ecglps:PhysicalSample
Class: ecglps:SamplingProcess
Class: ecglps:Feature
Class: ecglps:InformationObject
Class: ecglps:Property
```

16.2.2 Core Axioms

Firstly, we define that a PhysicalSample should be originated from some Feature.

$$\exists \mathsf{originatesFrom}.\mathsf{Feature} \sqsubseteq \mathsf{PhysicalSample} \tag{16.1}$$

$$\mathsf{PhysicalSample} \sqsubseteq \forall \mathsf{originatesFrom}.\mathsf{Feature} \tag{16.2}$$

A PhysicalSample should also be derived from a SamplingProcess. Thus, we have the following axioms.

$$\exists \mathsf{derivedUsing}.\mathsf{SamplingProcess} \sqsubseteq \mathsf{PhysicalSample} \tag{16.3}$$

$$\mathsf{PhysicalSample} \sqsubseteq \forall \mathsf{derivedUsing}.\mathsf{SamplingProcess} \tag{16.4}$$

A SamplingProcess has exactly an input Feature, and should derive an output PhysicalSample.

$$\mathsf{SamplingProcess} \sqsubseteq (=1)\mathsf{isInputFeatureOf}^{-}.\mathsf{Feature} \tag{16.5}$$

$$\mathsf{SamplingProcess} \sqsubseteq (=1)\mathsf{derivedUsing}^{-}.\mathsf{PhysicalSample} \tag{16.6}$$

A PhysicalSample is a subclass of a Feature which is a subclass of things.

$$\mathsf{PhysicalSample} \sqsubseteq \mathsf{Feature} \tag{16.7}$$

$$\mathsf{Feature} \sqsubseteq \top \tag{16.8}$$

When a Feature is the input feature of a SamplingProcess which derive another PhysicalSample, we say the PhysicalSample originates from the Feature.

$$\mathsf{derivedUsing} \circ \mathsf{isInputFeatureOf}^{-} \equiv \mathsf{originatesFrom} \tag{16.9}$$

A PhysicalSample can have InformationObject which provides additional information about the PhysicalSample. Similarly, it can also have Property to describe some attributes, such as material type, color, age, and so forth.

$$\mathsf{PhysicalSample} \sqsubseteq \forall \mathsf{isDescribedBy}.\mathsf{InformationObject} \tag{16.10}$$

$$\mathsf{PhysicalSample} \sqsubseteq \forall \mathsf{hasProperty}.\mathsf{Property} \tag{16.11}$$

16.3 Alignment

16.3.1 Alignment with Information Object pattern

```
Prefix: ecglps: <http://schema.geolink.org/dev/physicalsample#>
Prefix: ecglio: <http://schema.geolink.org/dev/informationobject#>

Ontology: <http://schema.geolink.org/dev/physicalsample-to-informationobject>

ObjectProperty: ecglps:isDescribedBy
ObjectProperty: ecglio:isDescribedBy

Class: ecglps:PhysicalSample

Class: ecglps:InformationObject
Class: ecglio:InformationObject
```

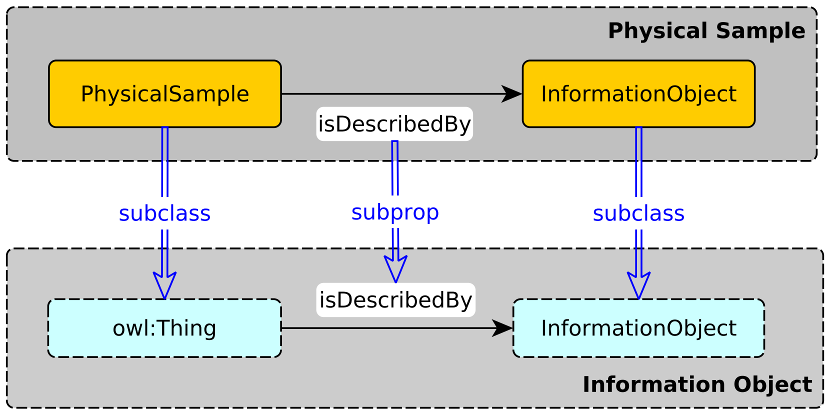

Figure 16.2: The Physical Sample pattern aligned to the Information Object pattern

$$\text{ecglps} : \text{isDescribedBy} \sqsubseteq \text{ecglio} : \text{isDescribedBy} \quad (16.12)$$
$$\text{ecglps} : \text{InformationObject} \sqsubseteq \text{ecglio} : \text{InformationObject} \quad (16.13)$$

16.3.2 Alignment with Event pattern

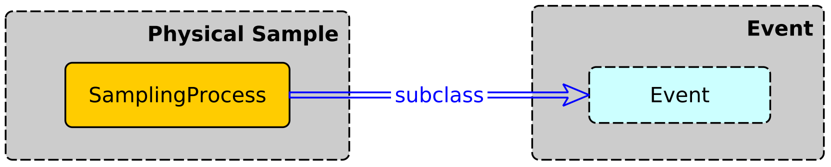

Figure 16.3: The Physical Sample pattern aligned to the Event pattern

```
Prefix: ecglps: <http://schema.geolink.org/dev/physicalsample#>
Prefix: ecglio: <http://schema.geolink.org/dev/event#>

Ontology: <http://schema.geolink.org/dev/physicalsample-to-event>

Class: ecglps:SamplingProcess
Class: ecglev:Event
```

$$\text{ecglps} : \text{SamplingProcess} \sqsubseteq \text{ecglev} : \text{Event} \quad (16.14)$$

16.3.3 Alignment with Property Value pattern

```
Prefix: ecglps: <http://schema.geolink.org/dev/physicalsample#>
Prefix: ecglpv: <http://schema.geolink.org/dev/propertyvalue#>

Ontology: <http://schema.geolink.org/dev/physicalsample-to-propertyvalue>
```

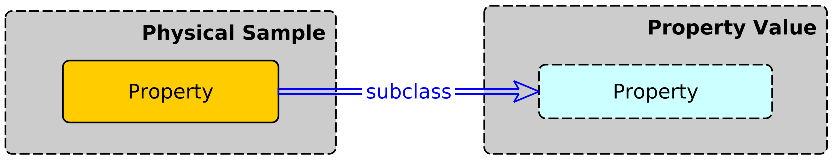

Figure 16.4: The Physical Sample pattern aligned to the Property Value pattern

Class: ecglps:Property
Class: ecglpv:Property

$$\mathsf{ecglps:Property} \sqsubseteq \mathsf{ecglpv:Property} \tag{16.15}$$

17 Property Value pattern

17.1 Description

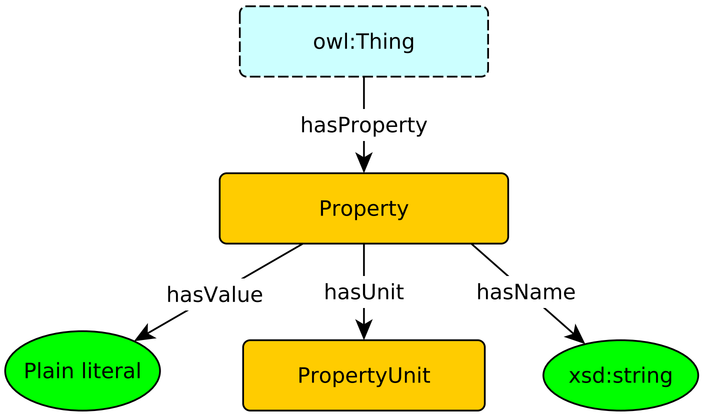

Figure 17.1: Property Value pattern

The *Property Value* pattern has been developed to model the properties of things. Currently, we differentiate the Property Value pattern from the Information Object pattern. Properties modeled by the Property Value pattern are those which are inherent to the target object, whereas the Information Object pattern captures additional information associated with the target thing. For example, *age*, *shape*, and *material type* should be modeled as the properties of a physical sample. On the contrary, *name*, *description*, and *website* should be considered as the information object for this physical sample. This distinction is derived from the philosophy literature (see http://plato.stanford.edu/entries/properties/).

A Property has a string to identify its name, such as "Age" and "Material Type". A Property has a value which can be a number, a string, a date, or some other value types. A PropertyUnit provides important metric information to interpret the value of the Property, and such metric units can be *meter*, *kilometer*, *gram*, and so forth. When the Property Value pattern is used to annotate data, we recommend to subclass the general class of Property with specific properties. For example, a data provider, who wants to publish some physical sample data, can first create a property called *MaterialTypeProperty*, and declare it as a subclass of Property. Then, he/she can populate the pattern with the actual *material type* values from data files.

17.2 Axiomatization

17.2.1 IRI Declaration

```
Prefix: ecglpv: <http://schema.geolink.org/dev/propertyvalue#>
Ontology: <http://schema.geolink.org/dev/propertyvalue>

ObjectProperty: hasProperty
ObjectProperty: hasUnit

DataProperty: hasValue
DataProperty: hasName
```

Class: Property
Class: PropertyUnit

17.2.2 Core Axioms

Property captures the attributes of things, and one thing can have zero or multiple Property which describe different aspects of the thing. A Property must be associated with exactly one thing, and a Property cannot be associated with another Property.

$$\top \sqsubseteq (\forall \mathsf{hasProperty}.\mathsf{Property}) \tag{17.1}$$

$$\mathsf{Property} \sqsubseteq (=1\ \mathsf{hasProperty}^{-}.\top) \sqcap \neg\exists \mathsf{hasProperty}^{-}.\mathsf{Property} \tag{17.2}$$

We then define the domain and range for hasUnit.

$$\exists \mathsf{hasUnit}.\mathsf{PropertyUnit} \sqsubseteq \mathsf{Property} \tag{17.3}$$

$$\mathsf{Property} \sqsubseteq \forall \mathsf{hasUnit}.\mathsf{PropertyUnit} \tag{17.4}$$

We also define the axioms for the property name and the property values.

$$\exists \mathsf{hasName}.\mathsf{xsd{:}string} \sqsubseteq \mathsf{Property} \tag{17.5}$$

$$\mathsf{Property} \sqsubseteq \forall \mathsf{hasName}.\mathsf{xsd{:}string} \tag{17.6}$$

$$dom(\mathsf{hasValue}) \sqsubseteq \mathsf{Property} \tag{17.7}$$

Bibliography

[Hobbs and Pan, 2006] Hobbs, J. and Pan, F. (2006). Time ontology in OWL. W3C working draft, W3C. http://www.w3.org/TR/2006/WD-owl-time-20060927/.

[Hu et al., 2013] Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., and Kolas, D. (2013). A geo-ontology design pattern for semantic trajectories. In Tenbrink, T., Stell, J. G., Galton, A., and Wood, Z., editors, *Spatial Information Theory – 11th International Conference, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings*, volume 8116 of *Lecture Notes in Computer Science*, pages 438–456. Springer, Heidelberg.

[Krisnadhi et al., 2014] Krisnadhi, A., Arko, R., Carbotte, S., Chandler, C., Cheatham, M., Finin, T., Hitzler, P., Janowicz, K., Narock, T., Raymond, L., Shepherd, A., and Wiebe, P. (2014). An ontology pattern for oceanograhic cruises: Towards an oceanograhper's dream of integrated knowledge discovery. Ocean-Link Technical Report 2014.1.

[Oberle et al., 2007] Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., Mougouie, B., Baumann, S., Vembu, S., Romanelli, M., Buitelaar, P., Engel, R., Sonntag, D., Reithinger, N., Loos, B., Zorn, H.-P., Micelli, V., Porzel, R., Schmidt, C., Weiten, M., Burkhardt, F., and Zhou, J. (2007). DOLCE ergo SUMO: On Foundational and Domain Models in the SmartWeb Integrated Ontology (SWIntO). *Journal of Web Semantics*, 5(3):156–174.

[van Hage et al., 2011] van Hage, W. R., Malaisé, V., Segers, R., Hollink, L., and Schreiber, G. (2011). Design and use of the Simple Event Model (SEM). *Journal of Web Semantics*, 9(2):128–136.