Ontology Modules for Grain Supply Chain Tracing

Version 1.0

Contributors:
COGAN SHIMIZU — Kansas State University; coganmshimizu@ksu.edu
ABHILEKHA DALAL — Kansas State University; adalal@ksu.edu
PASCAL HITZLER — Kansas State University; hitzler@ksu.edu
EVAN WALLACE — National Institute of Standards and Technology
FRANK RIDDICK — National Institute of Standards and Technology
SCOTT NIEMAN — Land O'Lakes, Inc.
JOE TEVIS — VIS Consulting, Inc.
FARHAD AMERI — Texas State University

Document Date: November 15, 2024

This work was performed under financial assistance award 70NANB19H094 from U.S. Department of Commerce, National Institute of Standards and Technology.

Contents

List of Figures

1 Overview

We are presenting core ontology modules aimed at food supply chain tracking related to grains. This work was performed under financial assistance award 70NANB19H094 from U.S. Department of Commerce, National Institute of Standards and Technology.

Development of the modules was a collaborative effort and was carried out using the principles laid out in, e.g., [Krisnadhi and Hitzler, 2016, Krisnadhi et al., 2016, Shimizu et al., 2020, Hitzler and Shimizu, 2018]. The modeling team included domain experts, data experts, software developers, and ontology engineers.

The (partial) ontology constituted by these modules has, in particular, been developed as a *modular* ontology [Hitzler et al., 2017, Hitzler and Shimizu, 2018, Shimizu et al., 2023] based on ontology design patterns [Hitzler et al., 2016, Shimizu et al., 2019b]. This means, in a nutshell, that we first identified key terms relating to the data content and expert perspectives on the domain to be modeled, and then developed ontology modules for these terms. The resulting modules, which were informed by corresponding ontology design patterns, are listed and discussed in Chapter 3. In Chapter 5 we discuss possible next steps.

A use case description which drove the development of these modules can be phrased as follows: *The end goal is to provide queryable (meta)data which makes it possible to trace agri-food supply chains relevant to grains. For this, we will develop an ontology which is an adequate data schema for all relevant data, from all relevant sources, that is needed to trace these supply chains. Of relevance is all data that pertains to (possible) food contamination that may affect health.*

A more restricted perspective on the use case – which is still adequate for the modules provided herein – is: *Identify all Traceable Resource Units (TRUs), relevant events, containers, which are involved in the past (or future) of a given TRU, together with the information how these TRUs, events, containers relate to the given TRU.*

For background regarding Semantic Web standards, in particular the Web Ontology Language OWL, including its relation to description logics, we refer the reader to [Hitzler et al., 2012, Hitzler et al., 2010]. For a general overview on the Semantic Web field, see [Hitzler, 2021].

All feedback is most welcome, and can be directed to Cogan Shimizu, coganmshimizu@ksu.edu, and Pascal Hitzler, hitzler@ksu.edu.

2 A Guide to Axioms and Diagrams

Primer on Ontology Axioms

Logical axioms are presented (mostly) in description logic notation, which can be directly translated into the Web Ontology Language OWL [Hitzler et al., 2010]. We use description logic notation because it is, in the end, easier for humans to read than any of the other serializations.[1]

Logical axioms serve many purposes in ontology modeling and engineering [Hitzler and Krisnadhi, 2016]; in our context, the primary reason why we choose a strong axiomatization is to disambiguate the ontology.

Almost all axioms which are part of our modules are of the straightforward and local types. We will now describe these types in more detail, as it will make it much easier to understand the axiomatization below.

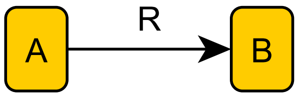

Figure 2.1: Generic node-edge-node schema diagram for explaining systematic axiomatization

There is a systematic way to look at each node-edge-node triple in a schema diagram in order to decide on some of the axioms which should be added: Given a node-edge-node triple with nodes A and B and edge R from A to B, as depicted in Figure 2.1, we check all of the following axioms whether they should be included.[2] We list them in natural language, see Figure 2.2 for the formal versions in description logic notation, and Figure 2.3 for the same in Manchester syntax, where we also list our names for these axioms.

1. A is a subClass of B.
2. A and B are disjoint.
3. The domain of R is A.
4. For every B which has an inverse R-filler, this inverse R-filler is in A. In other words, the domain of R, scoped by B, is A.
5. The range of R is B.
6. For every A which has an R-filler, this R-filler is in B. In other words, the range of R, scoped by A, is B.
7. For every A there has to be an R-filler in B.

[1]Preliminary results supporting this claim can be found in [Shimizu, 2017].

[2]The OWLAx Protégé plug-in [Sarker et al., 2016] provides a convenient interface for adding these axioms. An empirical evaluation in [Eberhart et al., 2021] shows that these axioms cover the large majority of axioms actually used in OWL ontologies.

1. $A \sqsubseteq B$
2. $A \sqcap B \sqsubseteq \bot$
3. $\exists R.\top \sqsubseteq A$
4. $\exists R.B \sqsubseteq A$
5. $\top \sqsubseteq \forall R.B$
6. $A \sqsubseteq \forall R.B$
7. $A \sqsubseteq \exists R.B$
8. $B \sqsubseteq \exists R^{-}.A$
9. $\top \sqsubseteq \leq 1 R.\top$
10. $\top \sqsubseteq \leq 1 R.B$
11. $A \sqsubseteq \leq 1 R.\top$
12. $A \sqsubseteq \leq 1 R.B$
13. $\top \sqsubseteq \leq 1 R^{-}.\top$
14. $\top \sqsubseteq \leq 1 R^{-}.A$
15. $B \sqsubseteq \leq 1 R^{-}.\top$
16. $B \sqsubseteq \leq 1 R^{-}.A$
17. $A \sqsubseteq \geq 0 R.B$

Figure 2.2: Most common axioms which could be produced from a single edge R between nodes A and B in a schema diagram: description logic notation.

1.	A SubClassOf B	(subClass)
2.	A DisjointWith B	(disjointness)
3.	R some `owl:Thing` SubClassOf A	(domain)
4.	R some B SubClassOf A	(scoped domain)
5.	`owl:Thing` SubClassOf R only B	(range)
6.	A SubClassOf R only B	(scoped range)
7.	A SubClassOf R some B	(existential)
8.	B SubClassOf inverse R some A	(inverse existential)
9.	`owl:Thing` SubClassOf R max 1 `owl:Thing`	(functionality)
10.	`owl:Thing` SubClassOf R max 1 B	(qualified functionality)
11.	A SubClassOf R max 1 `owl:Thing`	(scoped functionality)
12.	A SubClassOf R max 1 B	(qualified scoped functionality)
13.	`owl:Thing` SubClassOf inverse R max 1 `owl:Thing`	(inverse functionality)
14.	`owl:Thing` SubClassOf inverse R max 1 A	(inverse qualified functionality)
15.	B SubClassOf inverse R max 1 `owl:Thing`	(inverse scoped functionality)
16.	B SubClassOf inverse R max 1 A	(inverse qualified scoped functionality)
17.	A SubClassOf R min 0 B	(structural tautology)

Figure 2.3: Most common axioms which could be produced from a single edge R between nodes A and B in a schema diagram: Manchester syntax.

8. For every B there has to be an inverse R-filler in A.
9. R is functional.
10. R has at most one filler in B.
11. For every A there is at most one R-filler.
12. For every A there is at most one R-filler in B.
13. R is inverse functional.
14. R has at most one inverse filler in A.
15. For every B there is at most one inverse R-filler.
16. For every B there is at most one inverse R-filler in A.
17. An A may have an R-filler in B.

Domain and range axoims are items 2–5 in this list. Items 6 and 7 are extistential axioms. Items 8–15 are about variants of functionality and inverse functionality. All axiom types except disjointness and those utilizing inverses also apply to datatype properties.

Structural tautologies are, indeed, tautologies, i.e., they do not carry any formal logical content. However as argued in [Hitzler and Krisnadhi, 2016] they can help humans to understand the ontology, by indicating *possible* relationships, i.e., relationships intended by the modeler which, however, cannot be cast into non-tautological axioms.

Explanations Regarding Schema Diagrams

We utilize schema diagrams to visualize the ontology. In our experience, simple diagrams work best for this purpose [Shimizu et al., 2019a]. The reader needs to bear in mind, though, that these diagrams are ambiguous and incomplete visualizations of the ontology (or module), as the actual ontology (or module) is constituted by the set of axioms provided.

We use the following visuals in our diagrams:

rectangular box with solid frame and orange fill: a class
rectangual box with dashed frame and blue fill: a module, which is described in more detail elsewhere in the document
rectangular box with dashed frame and purple fill: a set of URIs constituting a controlled vocabulary
oval with solid frame and yellow fill: a data type
arrow with white head and no label: a subClass relationship
arrow with solid tip and label: a relationship (or property) other than a subClass relationship

3 Modules

We list the individual modules of the ontology, together with their axioms and explanations thereof. Each axiom is listed only once (for now), i.e. some axioms pertaining to a module may be found in the axiom set listed for an earlier listed module. Schema diagrams are provided throughout, but the reader should keep in mind that while schema diagrams are very useful for understanding an ontology [Karima et al., 2017, Shimizu et al., 2019a], they are also inherently ambiguous.

The following are the modules which together constitute our partial ontology for some track and trace use cases within the agricultural industry.[1] Each of them will be presented in detail further below.

TRU
Containers
Identifier
Trajectory
Quantity of Material
Master Event with these specialized sub-modules

- **Transfer Event**
- **Transformation Event**
- **Custody Change Event**
- **Owner Change Event**
- **Transport Event**
- **Observation Event**

Before we discuss each of them in turn, let us provide a partial overview of the modules. This is given in Figure 3.1; a much more complete diagram can be found in Chapter 4. Central to our modeling are the notions of (1) Traceable Resource Unit (TRU, Section 3.1), which is an amount of material with an identifier, that is moving along the supply chain, (2) Events (Section 3.6) which are relevant for tracing TRUs, and (3) Containers (Section 3.2) which hold the TRUs, e.g. for transport. Both TRUs and Containers participate in Events in various roles (appropriate to the specific type of Event). Containers and TRUs may move along trajectories.

[1]Supported and worked scenarios (e.g. filling of containers, field operations, grain storage and distribution, testing, drying, blending, and shipment within the food supply chain) will be provided in some other document.

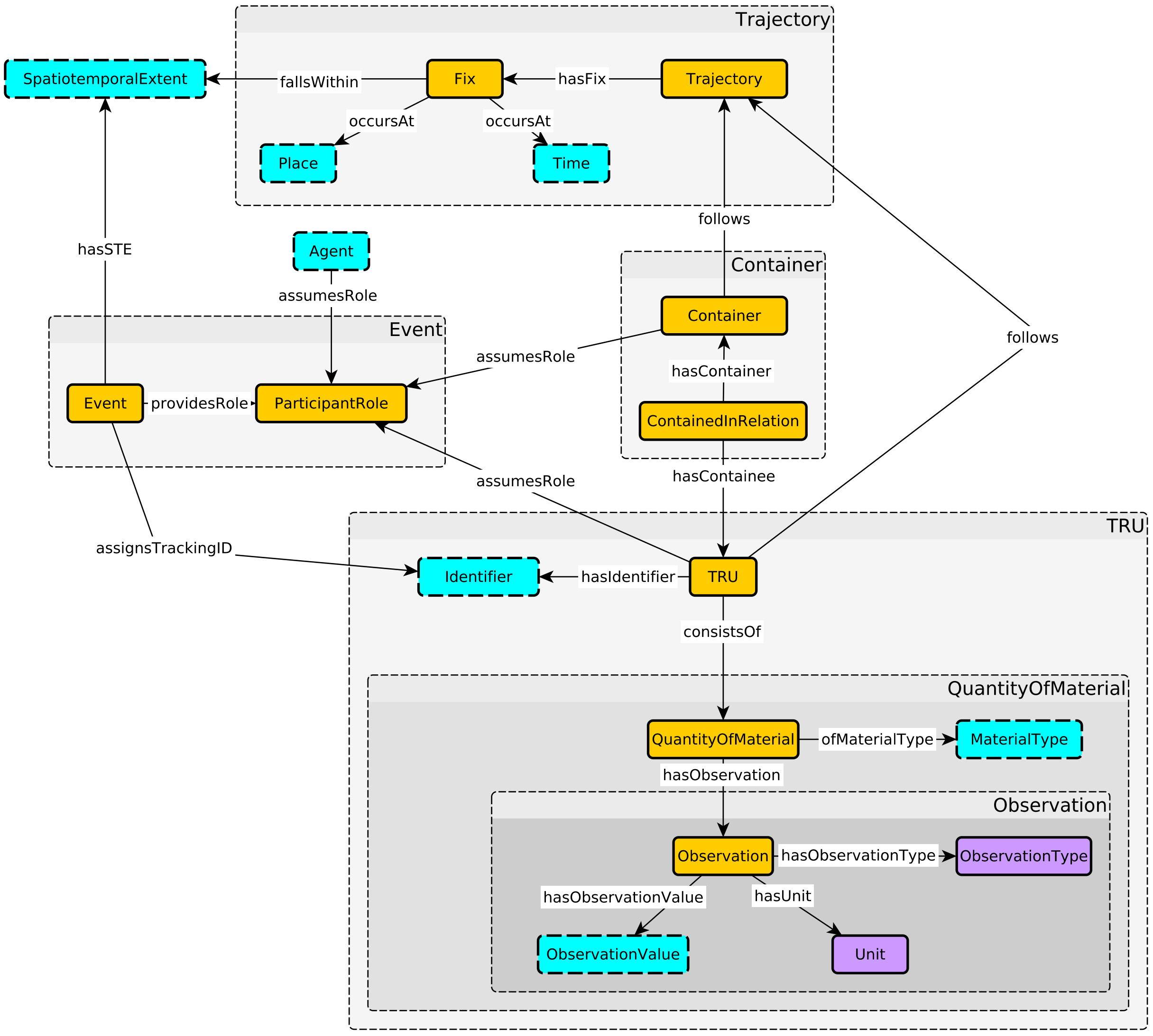

Figure 3.1: Overview of interconnected modules.

3.1 TRU

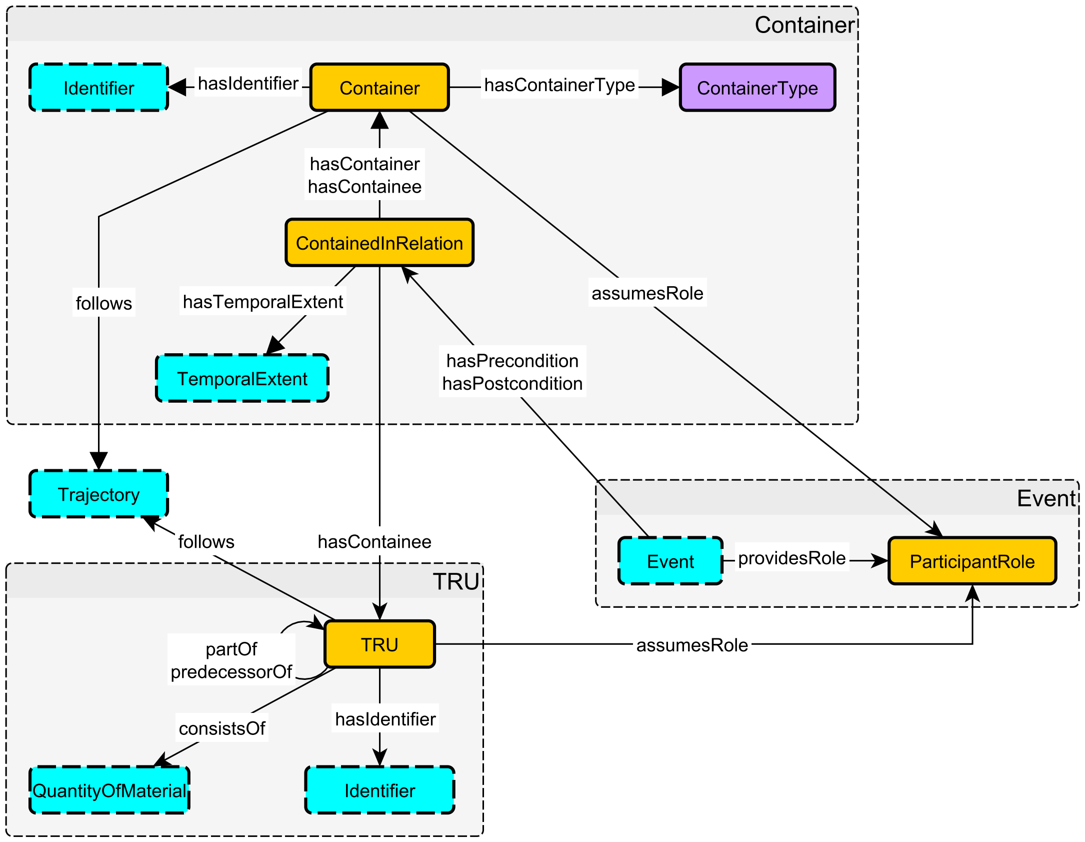

Figure 3.2: Schema Diagram for the TRU and Container modules. Note that the Event module is only partially depicted.

A Traceable Resource Unit (TRU) consists of a quantity of something (e.g. wheat grain) of interest, together with one of several identifiers for the TRU. TRUs are considered to be in containers (more about containers below in Section 3.2) and participate in Events relevant to tracing, in appropriate roles. TRUs can be part of other TRUs, and we have a predecessorOf relation between TRUs, relevant to tracing.

A schema diagram is given in Figure 3.2 on the lower left. The diagram also contains a diagram for the Container module (top) which we will discuss next. The Event module is only very partially depicted and will be spelled out in more detail further below.

Axioms:

$$\mathsf{TRU} \sqsubseteq {\geq}0\mathsf{partOf}.\mathsf{TRU} \tag{1}$$

$$\mathsf{TRU} \sqsubseteq {\geq}0\mathsf{predecessorOf}.\mathsf{TRU} \tag{2}$$

$$\mathsf{TRU} \sqsubseteq \forall\mathsf{hasIdentifier}.\mathsf{Identifier} \tag{3}$$

$$\mathsf{TRU} \sqsubseteq \exists \mathsf{hasIdentifier.Identifier} \tag{4}$$

$$\mathsf{Identifier} \sqsubseteq {\leq}1 \mathsf{hasIdentifier}^{-}.\top \tag{5}$$

$$\mathsf{TRU} \sqsubseteq \forall \mathsf{consistsOf.QuantityOfMaterial} \tag{6}$$

$$\mathsf{TRU} \sqsubseteq {\leq}1 \mathsf{consistsOf}.\top \tag{7}$$

$$\mathsf{TRU} \sqsubseteq \exists \mathsf{consistsOf.QuantityOfMaterial} \tag{8}$$

$$\mathsf{TRU} \sqsubseteq \forall \mathsf{assumesRole.ParticipantRole} \tag{9}$$

$$\mathsf{TRU} \sqsubseteq \forall \mathsf{follows.Trajectory} \tag{10}$$

$$\mathsf{TRU} \sqsubseteq \exists \mathsf{follows.Trajectory} \tag{11}$$

$$\mathsf{TRU} \sqsubseteq \exists \mathsf{hasContainee}^{-}.\mathsf{ContainedInRelation} \tag{12}$$

$$\mathsf{prececessorOf} \circ \mathsf{predecessorOf} \sqsubseteq \mathsf{predecessorOf} \tag{13}$$

Axiom Explanations:

1. Structural Tautology: A TRU may be partOf another TRU.
2. Structural Tautology: A TRU may be a predecessorOf of another TRU.
3. Scoped Range: The range of hasIdentifier is Identifier, scoped by TRU.
4. Existential: A TRU has a least one Identifier.
5. Inverse Scoped Functionality: An Identifier identifies at most one thing.
6. Scoped Range: The range of consistsOf is QuantityOfMaterial, scoped by TRU.
7. Scoped Functionality: consistsOf is functional, scoped by TRU.
8. Existential: A TRU consists of at least one QuantityOfMaterial.
9. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TRU.
10. Scoped Range: The range of follows is Trajectory, scoped by TRU.
11. Existential: A TRU follows a Trajectory.
12. Inverse Existential: A ContainedInRelation refers to at most one TRU.
13. Transitivity: predecessorOf is transitive.

Remarks:

1. A TRU always has an identifier (even though it may not be known); a TRU may have several identifiers.
2. A TRU is always contained in at least one container (even though it may not be known or an unusual type of container, such as a (part of a) field. Since containers can be in other containers, a TRU can be in several containers.
3. The ContainedInRelation is discussed and axiomatized further in Section 3.2.

Questions:

1. Should the partOf relationship be replaced with a more appropriate appropriate version from [Shimizu et al., 2019b], in which case it could then be declared transitive? The question is: which of those types? Probably member-collection. But could also be portion-mass or stuff-object?

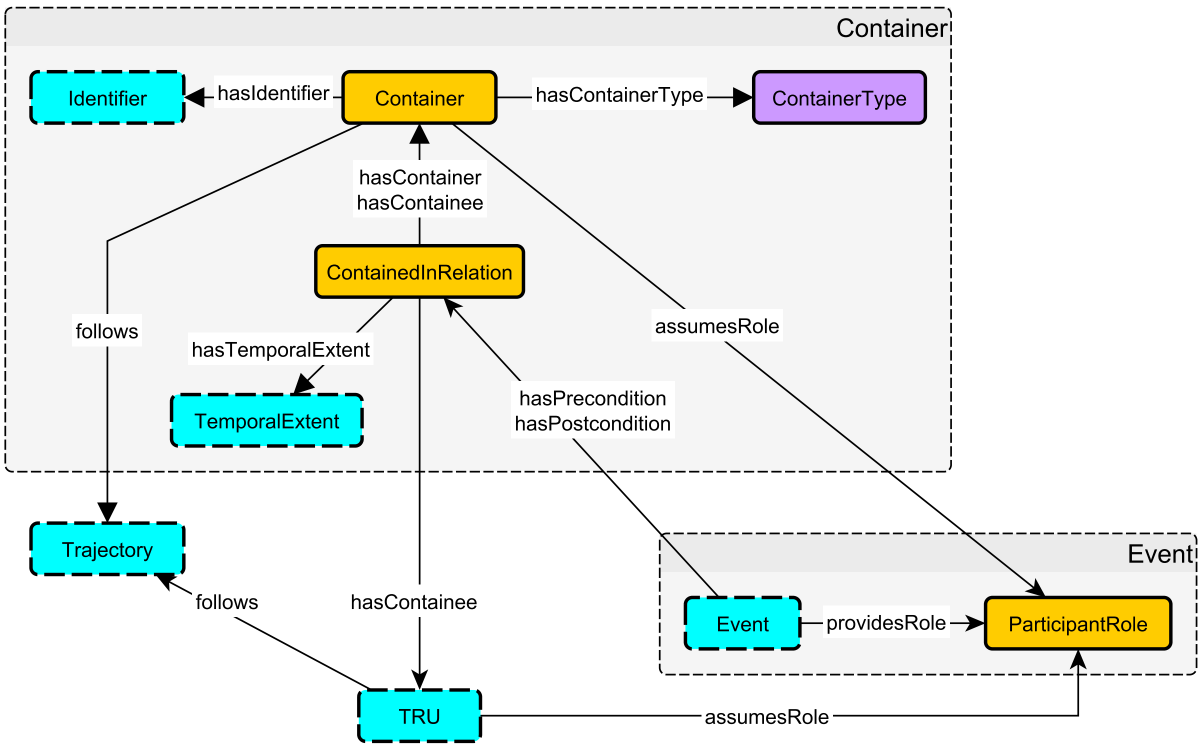

Figure 3.3: Schema Diagram for the Container module

3.2 Container

In our context, by Container we mean anything that can hold TRUs, i.e., we use the term in a rather abstract sense. A container may even be stationary (such as a grain elevator), moving in a continuous manner (such as a harvester/combine) or even lack a physical enclosure (such as a grain field which is then defined by its geospatial polygon boundary). At the same time, for our use case, we do not need to be very specific about Containers: they carry identifiers, contain TRUs, and follow trajectories.

Axioms:

$$\text{Container} \sqsubseteq \forall \text{follows.Trajectory} \tag{1}$$

$$\text{Container} \sqsubseteq \exists \text{follows.Trajectory} \tag{2}$$

$$\text{Container} \sqsubseteq \forall \text{hasIdentifier.Identifier} \tag{3}$$

$$\text{Container} \sqsubseteq \exists \text{hasIdentifier.Identifier} \tag{4}$$

$$\text{Identifier} \sqsubseteq {\leq}1 \text{hasIdentifier}^{-}.\top \tag{5}$$

$$\top \sqsubseteq \forall \text{hasContainerType.ContainerType} \tag{6}$$

$$\exists \text{hasContainerType}.\top \sqsubseteq \text{Container} \tag{7}$$

$$\text{Container} \sqsubseteq \exists \text{hasContainerType.ContainerType} \tag{8}$$

$$\mathsf{Container} \sqsubseteq \geq 0 \mathsf{assumesRole}.\mathsf{ParticipantRole} \tag{9}$$
$$\exists \mathsf{hasContainer}.\top \sqsubseteq \mathsf{ContainedInRelation} \tag{10}$$
$$\top \sqsubseteq \forall \mathsf{hasContainer}.\mathsf{Container} \tag{11}$$
$$\exists \mathsf{hasContainee}.\top \sqsubseteq \mathsf{ContainedInRelation} \tag{12}$$
$$\mathsf{ContainedInRelation} \sqsubseteq \exists \mathsf{hasContainee}.\top \tag{13}$$
$$\mathsf{ContainedInRelation} \sqsubseteq \exists \mathsf{hasContainer}.\top \tag{14}$$
$$\mathsf{ContainedInRelation} \sqsubseteq \geq 0 \mathsf{hasContainee}.(\mathsf{Container} \sqcup \mathsf{TRU}) \tag{15}$$
$$\top \sqsubseteq \forall \mathsf{hasTemporalExtent}.\mathsf{TemporalExtent} \tag{16}$$
$$\mathsf{ContainedInRelation} \sqsubseteq \exists \mathsf{hasTemporalExtent}.\mathsf{TemporalExtent} \tag{17}$$

Axiom Explanations:

1. Scoped Range: The range of follows is Trajectory, scoped by Container.
2. Existential: A Container follows a Trajectory.
3. Scoped Range: The range of hasIdentifier is Identifier, scoped by Container.
4. Existential: A Container has an Identifier.
5. Inverse Scoped Functionality: An Identifier identifies at most one thing.
6. Range: The range of hasContainerType is ContainerType.
7. Domain: The domain of hasContainerType is Container.
8. Existential: hasContainerType
9. Structural Tautology: A Container may assume a ParticipantRole.
10. Domain: The domain of hasContainer is ContainedInRelation.
11. Range: The range of hasContainer is Container.
12. Domain: The domain of hasContainee is ContainedInRelation.
13. Existential: A ContainedInRelation has some containee.
14. Existential: A ContainedInRelation has some container.
15. Structural Tautology: A ContainedInRelation may have a containee that is a Container or TRU.
16. Range: The range of hasTemporalExtent is TemporalExtent.
17. Scoped Existential: A ContainedInRelation has a TemporalExtent.

Remarks:

- We would like to state that a container cannot be in two places at the same time, but it is not clear at this stage whether that can be done in OWL. To be revisited when looking at spatiotemporal extents.
- Every Container has a unique identifier, even though that identifier may be a very local reference (such as *on the back parking lot*). The Identifier module details can reflect this.
- Trajectories of TRU and Container have to be compatible, as has the temporality of the ContainedInRelation.
- hasPre- and hasPostcondition axioms are included in the Event module description below.
- The contained-in relation has been reified in order to make it possible to record a temporal extent for it, and also (if relevant) that certain containment relations are relevant to certain types of critical tracking events, e.g. for the TransferEvent which is described further below. Furthermore, contained-in relations may be complex, e.g. it may be relevant whether contamination of a container can cause contamination of container content. By reifying the contained-in relation, it will be easier to extend the model when and if desired.
- It is possible to specify transitivity of the contained-in relation, using complex axioms, if desired.

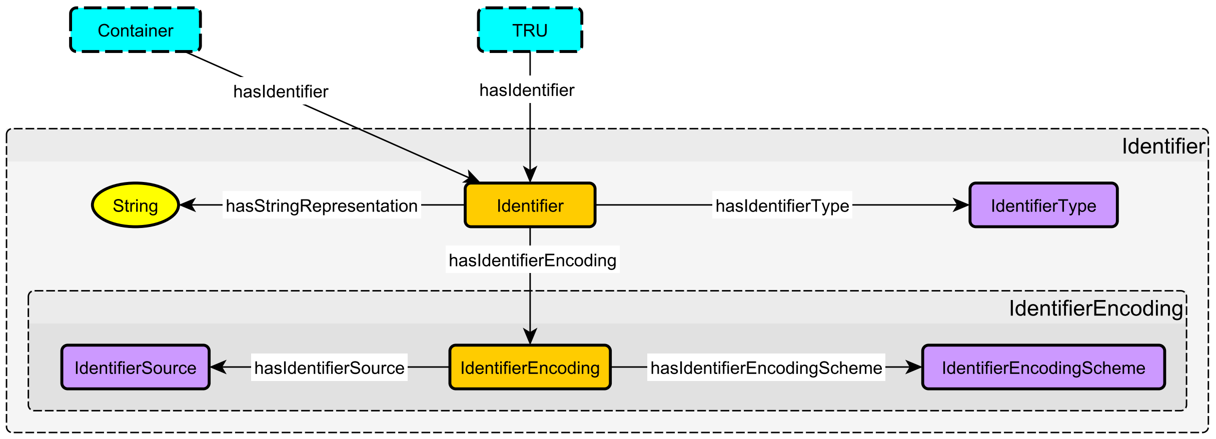

Figure 3.4: Schema Diagram for the Identifier module

3.3 Identifier

Identifiers are the labels or serial-number through which a TRU (Traceable Resource Unit) or Container can be traced in the supply chain. Identifiers have a type and representation; we also specify a sub-module IdentifierEncoding to represent encoding information for identifiers.

Recall that purple boxes (as in Figure 3.3) indicate that the class should be treated as a controlled vocabulary, and is particular to the use-case at hand. For example, entities in the class IdentifierEncodingScheme will be URIs that uniquely identify particular encoding schemes (such as, the DOI scheme for online documents). There is no need in our use case to provide further scheme information as part of the ontology as, in general, these controlled vocabularies are structured and maintained by other organizations, and we can thus use these entities to point to their documentation.

Axioms:

We use hIES in place of hasIdentifierEncodingScheme.

$$\mathsf{TRU} \sqsubseteq \forall \mathsf{hasIdentifier}.\mathsf{Identifier} \tag{1}$$
$$\mathsf{Identifier} \sqsubseteq {\leq}1\mathsf{hasIdentifier}^{-}.\top \tag{2}$$
$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasStringRepresentation}.\mathsf{xsd{:}string} \tag{3}$$
$$\mathsf{Identifier} \sqsubseteq \exists \mathsf{hasStringRepresentation}.\mathsf{xsd{:}string} \tag{4}$$
$$\exists \mathsf{hasIdentifierType}.\mathsf{IdentifierType} \sqsubseteq \mathsf{Identifier} \tag{5}$$
$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasIdentifierType}.\mathsf{IdentifierType} \tag{6}$$
$$\exists \mathsf{hasIdentifierEncoding}.\mathsf{IdentifierEncoding} \sqsubseteq \mathsf{Identifier} \tag{7}$$
$$\mathsf{Identifier} \sqsubseteq \forall \mathsf{hasIdentifierEncoding}.\mathsf{IdentifierEncoding} \tag{8}$$
$$\mathsf{Identifier} \sqsubseteq \exists \mathsf{hasIdentifierEncoding}.\mathsf{IdentifierEncoding} \tag{9}$$
$$\exists \mathsf{hasIdentifierSource}.\mathsf{IdentifierSource} \sqsubseteq \mathsf{IdentifierEncoding} \tag{10}$$
$$\mathsf{IdentifierEncoding} \sqsubseteq \forall \mathsf{hasIdentifierSource}.\mathsf{IdentifierSource} \tag{11}$$

$$\mathsf{IdentifierEncoding} \sqsubseteq \exists \mathsf{hasIdentifierSource}.\mathsf{IdentifierSource} \quad (12)$$

$$\exists \mathsf{hIES}.\mathsf{IdentifierEncodingScheme} \sqsubseteq \mathsf{IdentifierEncoding} \quad (13)$$

$$\mathsf{IdentifierEncoding} \sqsubseteq \forall \mathsf{hIES}.\mathsf{IdentifierEncodingScheme} \quad (14)$$

Axiom Explanations:

1. Scoped Range: The range of hasIdentifier is Identifier, scoped by TRU.
2. Inverse Scoped Functionality: hasIdentifier
3. Scoped Range: The range of hasStringRepresentation is xsd:String, scoped by Identifier.
4. Existential: hasStringRepresentation
5. Scoped Domain: The domain of hasIdentifierType is Identifier, scoped by IdentifierType.
6. Scoped Range: The range of hasIdentifierType is IdentifierType, scoped by Identifier.
7. Scoped Domain: The domain of hasIdentifierEncoding is Identifier, scoped by IdentifierEncoding.
8. Scoped Range: The range of hasIdentifierEncoding is IdentifierEncoding, scoped by Identifier.
9. Existential: hasIdentifierEncoding
10. Scoped Domain: The domain of hasIdentifierSource is IdentifierEncoding, scoped by IdentifierSource.
11. Scoped Range: The range of hasIdentifierSource is IdentifierSource, scoped by IdentifierEncoding.
12. Existential: hasIdentifierSource
13. Scoped Domain: The domain of hasIdentifierEncodingScheme is IdentifierEncoding, scoped by IdentifierEncodingScheme.
14. Scoped Range: The range of hasIdentifierEncodingScheme is IdentifierEncodingScheme, scoped by IdentifierEncoding.

3.4 Trajectory

Trajectory is associated with the Container that is carrying something to be transported. In our modeling, a trajectory is essentially a sequence of Fixes in time and space. As such, a Fix occurs at some time and at some place. We also specify that a Fix falls within some spatiotemporal extent, which will be used to connect fixes with events.

This module is adapted from [Shimizu et al., 2019b].

Axioms:

$$\mathsf{Fix} \sqsubseteq \exists \mathsf{occursAt}.\mathsf{Place} \quad (1)$$

$$\mathsf{Fix} \sqsubseteq \exists \mathsf{occursAt}.\mathsf{Time} \quad (2)$$

$$\exists \mathsf{hasFix}.\mathsf{Fix} \sqsubseteq \mathsf{Trajectory} \quad (3)$$

$$\mathsf{Trajectory} \sqsubseteq \forall \mathsf{hasFix}.\mathsf{Fix} \quad (4)$$

$$\mathsf{Fix} \sqsubseteq \geq 0 \mathsf{hasNext}.\mathsf{Fix} \quad (5)$$

$$\mathsf{Fix} \sqsubseteq \exists \mathsf{fallsWithin}.\mathsf{SpatiotemporalExtent} \quad (6)$$

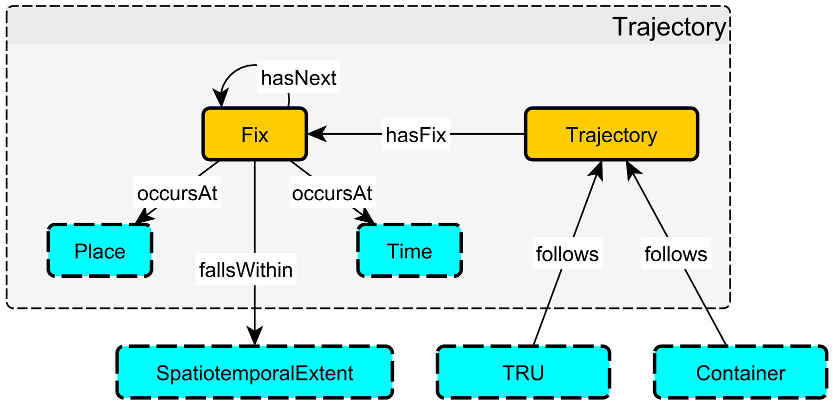

Figure 3.5: Schema Diagram for the Trajectory module

Axiom Explanations:

1. Existential: A Fix must occur at a Place.
2. Existential: A Fix must occur at a Time.
3. Scoped Domain: The domain of hasFix is Trajectory, scoped by Fix.
4. Scoped Range: The range of hasFix is Fix, scoped by Trajectory.
5. Structural Tautology: A Fix may have a next Fix.
6. Existential: A Fix must fall within a SpatiotemporalExtent.

Remarks:

- There should be specified compatibility axioms between Place, Time, and SpatiotemporalExtent for Fixes, but this is likely outside the modeling capacities for OWL.
- Time (Trajectory module), TemporalScope (Container module) and SpatioTemporalExtent will have to be properly related when modeled.

3.5 Quantity of Material

In our context, a Quantity of Material refers to whatever constitutes the relevant commodity making up a TRU; e.g., a certain amount of grain. In this example grain would be the MaterialType (a separate submodule), and corresponding measurements (e.g., mass, volume, or humidity) providing information about it would be produced at ObservationEvents (see module below).

Measurements (a submodule) come with a measurement type (such as, humidity, mass, length), and have a measurement value (which may be numeric, but could also be qualitative) and a unit (such as, meters or grams, or "numeric" for a count).

This module is adapted from [Shimizu et al., 2019b] – in particular the Quantity pattern from [Shimizu et al., 2019b] has been generalized to the Measurement submodule.

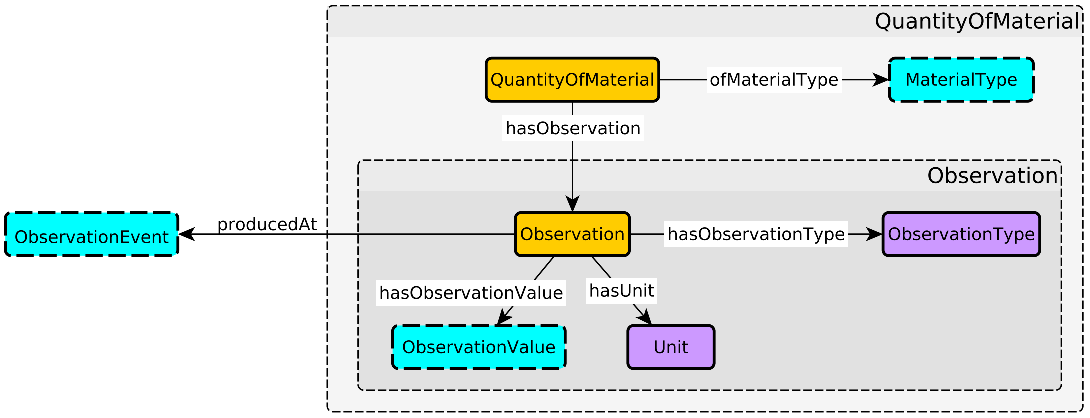

Figure 3.6: Schema Diagram for the Quantity of Material module

Axioms:

We use hMV for hasMeasurementValue.

$$\exists \text{ofMaterialType}.\text{MaterialType} \sqsubseteq \text{QuantityOfMaterial} \quad (1)$$
$$\text{QuantityOfMaterial} \sqsubseteq \forall \text{ofMaterialType}.\text{MaterialType} \quad (2)$$
$$\text{QuantityOfMaterial} \sqsubseteq \exists \text{ofMaterialType}.\text{MaterialType} \quad (3)$$
$$\text{QuantityOfMaterial} \sqsubseteq \forall \text{hasMeasurement}.\text{Measurement} \quad (4)$$
$$\exists \text{hasMeasurementType}.\text{MeasurementType} \sqsubseteq \text{Measurement} \quad (5)$$
$$\text{Measurement} \sqsubseteq \forall \text{hasMeasurementType}.\text{MeasurementType} \quad (6)$$
$$\text{Measurement} \sqsubseteq \exists \text{hasMeasurementType}.\text{MeasurementType} \quad (7)$$
$$\exists \text{hMV}.\text{MeasurementValue} \sqsubseteq \text{Measurement} \quad (8)$$
$$\text{Measurement} \sqsubseteq \forall \text{hMV}.\text{MeasurementValue} \quad (9)$$
$$\text{Measurement} \sqsubseteq \exists \text{hMV}.\text{MeasurementValue} \quad (10)$$
$$\text{Measurement} \sqsubseteq \forall \text{hasUnit}.\text{Unit} \quad (11)$$
$$\text{Measurement} \sqsubseteq \exists \text{hasUnit}.\text{Unit} \quad (12)$$
$$\text{Measurement} \sqsubseteq \forall \text{producedAt}.\text{ObservationEvent} \quad (13)$$
$$\text{Measurement} \sqsubseteq \exists \text{producedAt}.\text{ObservationEvent} \quad (14)$$

Axiom Explanations:

1. Scoped Domain: The domain of ofMaterialType is QuantityOfMaterial, scoped by MaterialType.
2. Scoped Range: The range of ofMaterialType is MaterialType, scoped by QuantityOfMaterial.
3. Existential: ofMaterialType
4. Scoped Range: The range of hasMeasurement is Measurement, scoped by QuantityOfMaterial.
5. Scoped Domain: The domain of hasMeasurementType is Measurement, scoped by MeasurementType.

6. Scoped Range: The range of hasMeasurementType is Measurement, scoped by MeasurementType.
7. Existential: hasMeasurementType
8. Scoped Domain: The domain of hasMeasurementValue is Measurement, scoped by MeasurementValue.
9. Scoped Range: The range of hasMeasurementValue is MeasurementValue, scoped by Measurement.
10. Existential: hasMeasurementValue
11. Scoped Range: The range of hasUnit is Unit, scoped by Measurement.
12. Existential: hasUnit
13. Scoped Range: The range of producedAt is ObservationEvent, scoped by Measurement.
14. Existential: producedAt

Remarks:

1. A Measurement is a type of Observation. We use Observation for more general uses or qualitative assessments, such as smells. The same structure may be used for Measurement.

3.6 Master Event

Master Event refers to an event model that can be understood as a generalized version of several of the relevant tracking events. While *Master Event* by itself may not be of independent interest for the model, we include it here to show how the models for the different relevant tracking events relate to each other. After discussing this, we move to the more specific tracking event models, below. This module is a very significantly adapted version from the (very simple) Event pattern from [Shimizu et al., 2019b].

We give some explanations based on the schema diagram in Figure 3.7. The Event class can be found in the middle of the diagram. Events, in this conceptualization, happen in space and time (i.e., have spatiotemporal extents), and provide roles for participants (which may be agents or or other things relevant to the event under discussion). Participant roles are in particular provided for TRUs, and some TRUs may have *source* roles, some may have *target* roles. For example, content from source TRUs may get merged and re-divided, resulting in target TRUs at the end of the event. There may be multiple source and target TRUs (as depicted in the alternative diagram in Figure 3.8). For some types of event, the ContainedInRelation (see the Container module) is relevant for the event, i.e. such a ContainedInRelation may be a constituent of a "pre" situation (meaning that it should be the case at the beginning of the event) and/or of a "post" situation (meaning that it should be the case at the end of the event). TRUs, Containers, ContainedInRelations may persist throughout the event (remain unchanged), or may not. Other roles which may be relevant may be that of owner or custodian, used equipment, etc. Each type of critical tracking event will define corresponding relevant roles. We also provide a *DocumentationSource* sub-module for information where further documentation for the event may be available.

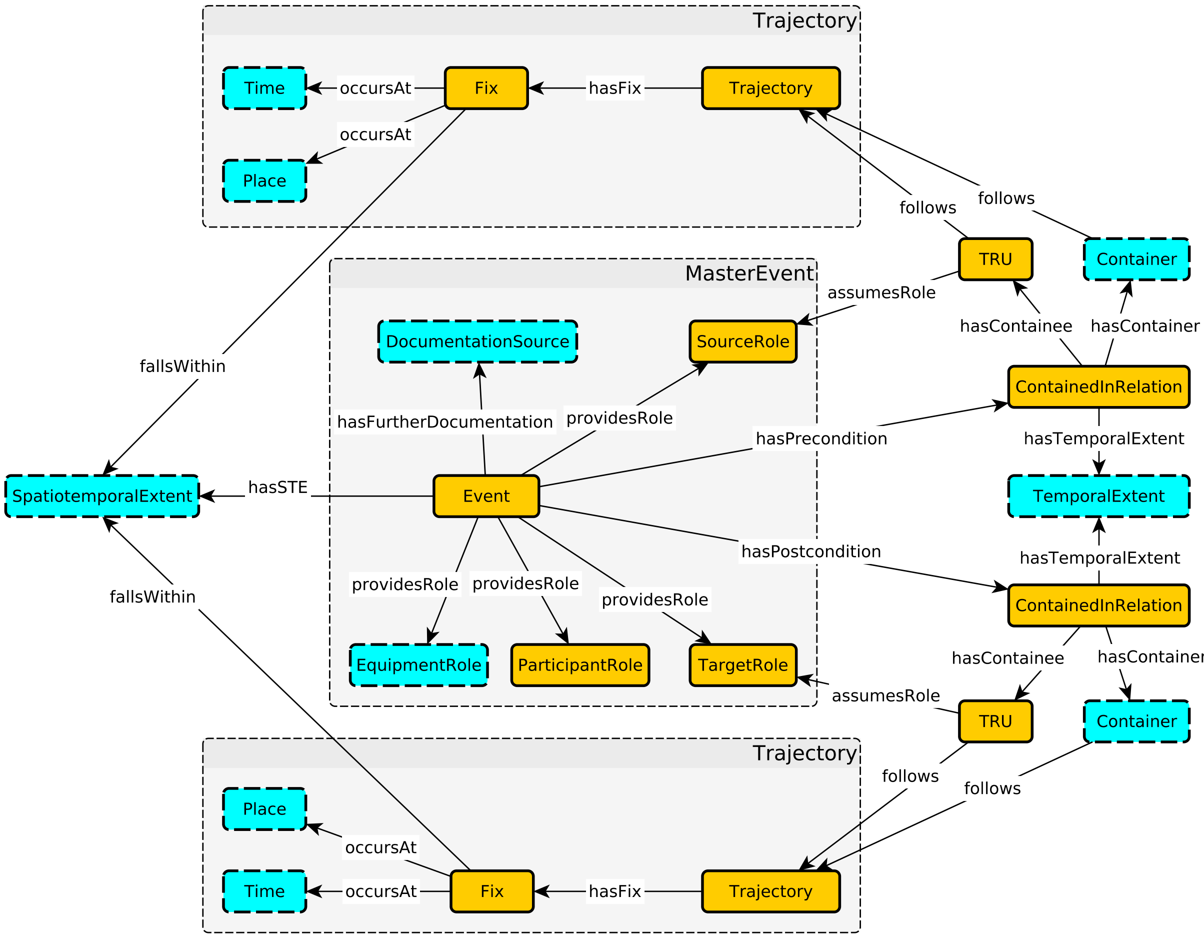

Figure 3.7: Schema Diagram for the MasterEvent module.

Axioms:

$$\top \sqsubseteq \forall \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \tag{1}$$
$$\mathsf{Event} \sqsubseteq \exists \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \tag{2}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{providesRole}.\mathsf{ParticipantRole} \tag{3}$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{providesRole}^{-}.\top \tag{4}$$
$$\mathsf{TRU} \sqsubseteq \forall \mathsf{assumesRole}.\mathsf{ParticipantRole} \tag{5}$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{assumesRole}^{-}.\top \tag{6}$$
$$\mathsf{Event} \sqsubseteq \forall \mathsf{hasFurtherDocumentation}.\mathsf{DocumentationSource} \tag{7}$$
$$\mathsf{SourceRole} \sqsubseteq \mathsf{ParticipantRole} \tag{8}$$
$$\mathsf{TargetRole} \sqsubseteq \mathsf{ParticipantRole} \tag{9}$$
$$\mathsf{EquipmentRole} \sqsubseteq \mathsf{ParticipantRole} \tag{10}$$
$$\mathsf{Event} \sqsubseteq \geq 0 \mathsf{hasPrecondition}.\mathsf{ContainedInRelation} \tag{11}$$

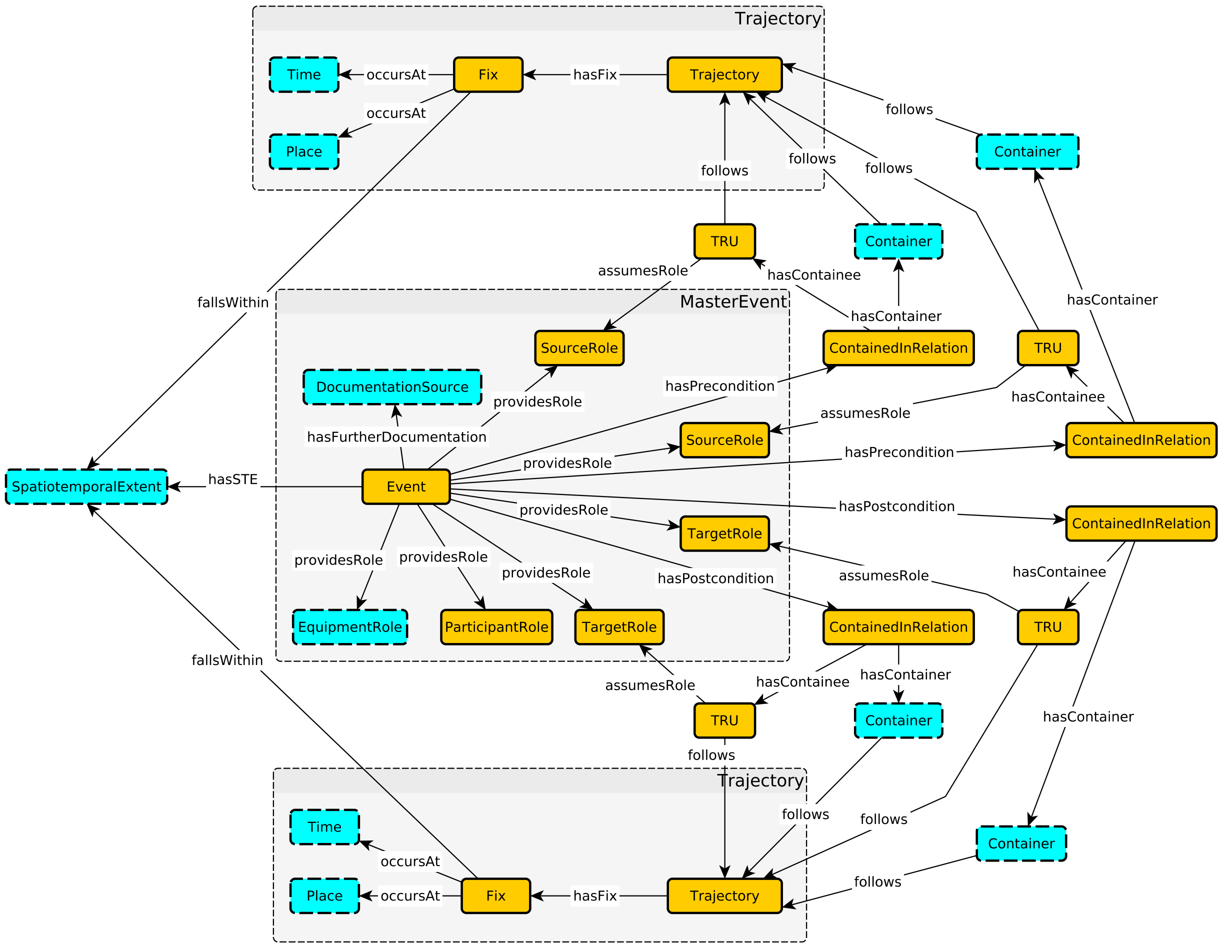

Figure 3.8: Schema Diagram for the MasterEvent module. This diagram deliberatly contains redundant elements to depict that there can be several source and target TRUs.

$$\text{Event} \sqsubseteq \geq 0 \text{hasPostcondition.ContainedInRelation} \tag{12}$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by Event.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is Event, scoped by TRU.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentationSource, scoped by Event.
8. Subclass: Every SourceRole is a ParticipantRole.
9. Subclass: Every TargetRole is a ParticipantRole.
10. Subclass: Every EquipmentRole is a ParticipantRole.

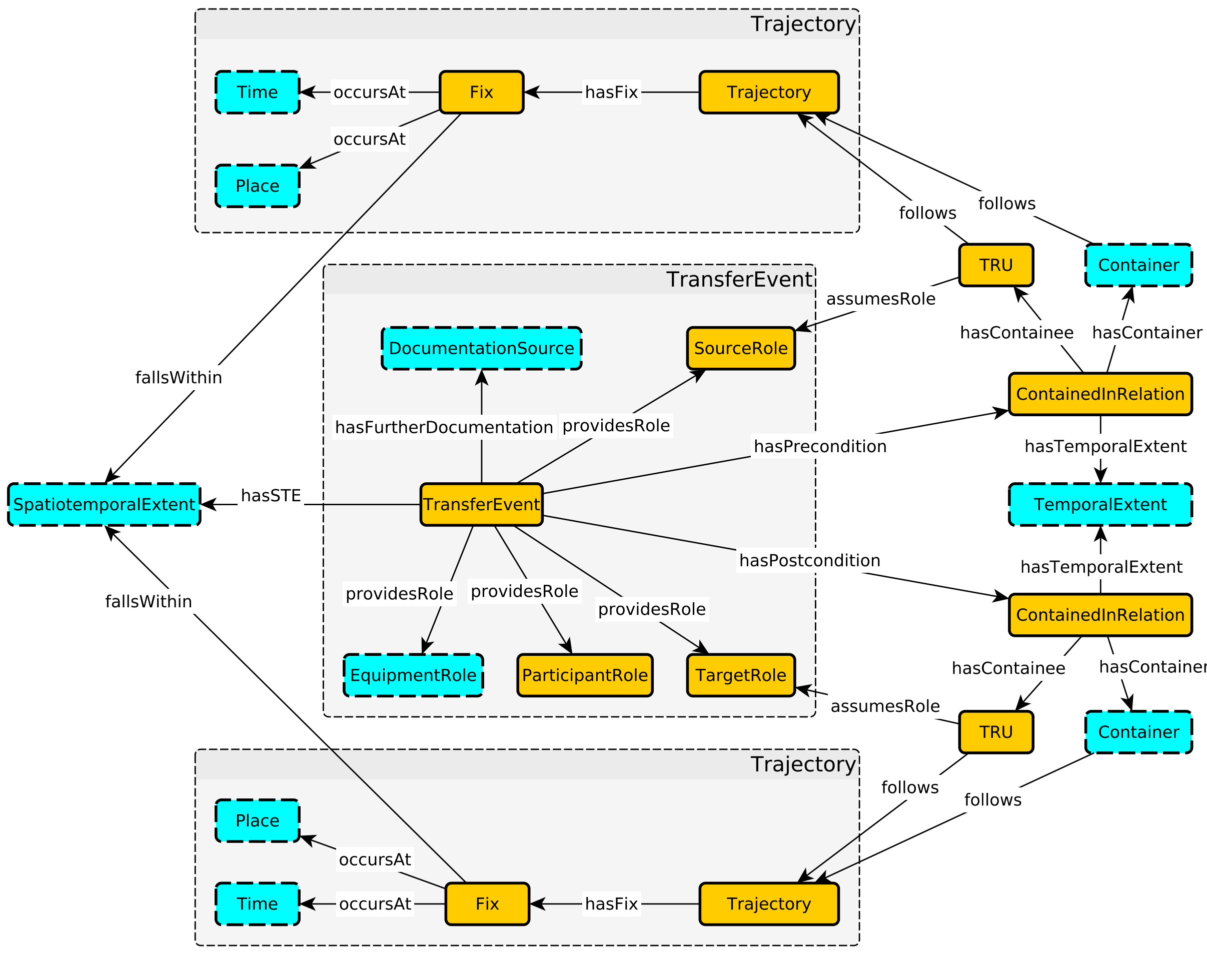

Figure 3.9: Schema Diagram for the TransferEvent module

11. Structural Tautology: An Event may have a ContainedInRelation via hasPrecondition.
12. Structural Tautology: An Event may have a ContainedInRelation via hasPostcondition.

3.6.0.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.
-
- Time (Trajectory module), TemporalScope (Container module) and SpatioTemporalExtent will have to be properly related when modeled.

3.6.1 Transfer Event

A Transfer Event happens when TRUs are combined, merged, split and/or moved into other containers. The diagram (and model) is essentially the Master Event model.

Axioms:

$$\top \sqsubseteq \forall \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \quad (1)$$
$$\mathsf{TransferEvent} \sqsubseteq \exists \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \quad (2)$$
$$\mathsf{TransferEvent} \sqsubseteq \forall \mathsf{providesRole}.\mathsf{ParticipantRole} \quad (3)$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{providesRole}^{-}.\top \quad (4)$$
$$\mathsf{TRU} \sqsubseteq \forall \mathsf{assumesRole}.\mathsf{ParticipantRole} \quad (5)$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{assumesRole}^{-}.\top \quad (6)$$
$$\mathsf{TransferEvent} \sqsubseteq \forall \mathsf{hasFurtherDocumentation}.\mathsf{DocumentationSource} \quad (7)$$
$$\mathsf{SourceRole} \sqsubseteq \mathsf{ParticipantRole} \quad (8)$$
$$\mathsf{TargetRole} \sqsubseteq \mathsf{ParticipantRole} \quad (9)$$
$$\mathsf{TransferEvent} \sqsubseteq \exists \mathsf{providesRole}.(\mathsf{SourceRole} \sqcap \exists \mathsf{assumesRole}^{-}.\mathsf{TRU}) \quad (10)$$
$$\mathsf{TransferEvent} \sqsubseteq \exists \mathsf{providesRole}.(\mathsf{TargetRole} \sqcap \exists \mathsf{assumesRole}^{-}.\mathsf{TRU}) \quad (11)$$
$$\mathsf{TransferEvent} \sqsubseteq \exists \mathsf{hasPrecondition}.\mathsf{ContainedInRelation} \quad (12)$$
$$\mathsf{TransferEvent} \sqsubseteq \exists \mathsf{hasPostcondition}.\mathsf{ContainedInRelation} \quad (13)$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by TransferEvent.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TransferEvent.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentationSource, scoped by TransferEvent.
8. Subclass: Every SourceRole is a Participantrole.
9. Subclass: Every TargetRole is a ParticipantRole.
10. existential and inverse functionality
11. existential and inverse functionality
12. Existential: hasPrecondition
13. Existential: hasPostcondition

3.6.1.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.

3.6.2 Transformation Event

A Transformation Event occurs when there is a change of the nature of source TRU content that is different from the original form. E.g., grain may be transformed into flour. The transformation itself is understood to be a transformation activity. Other than that, this module is essentially the same as the MasterEvent module.

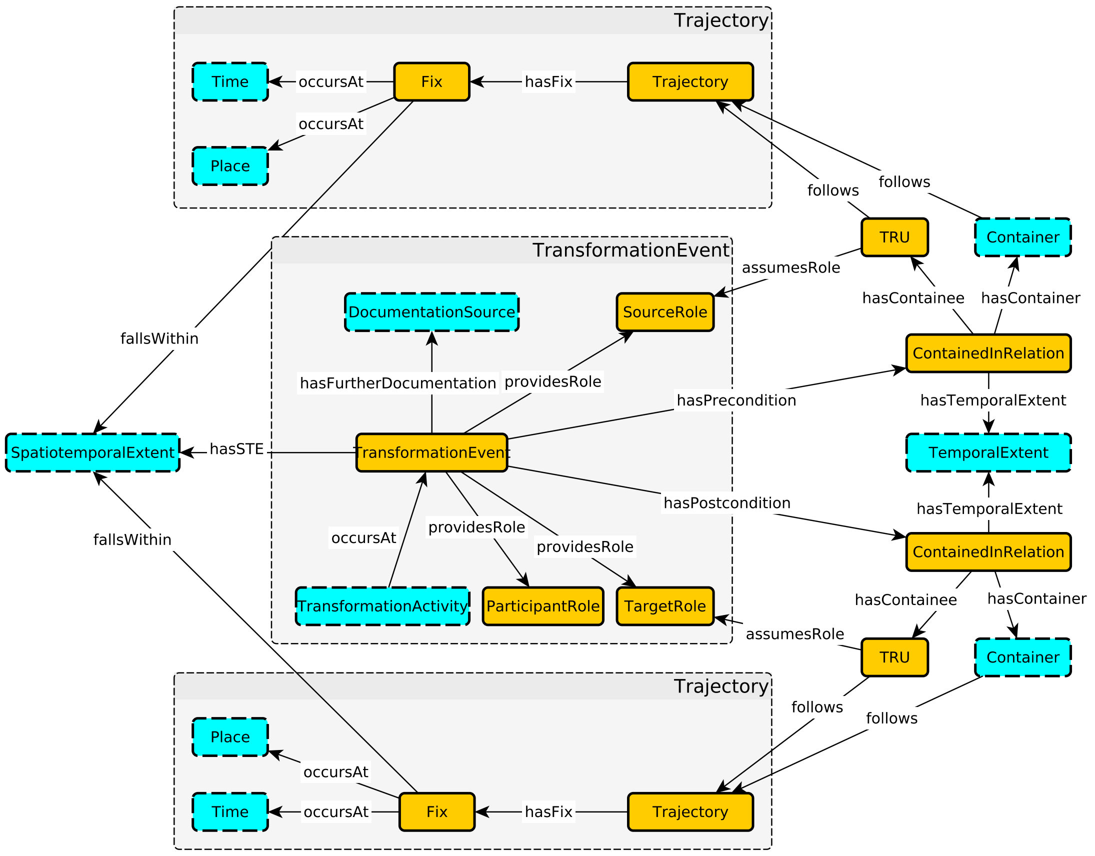

Figure 3.10: Schema Diagram for the TransformationEvent module

Axioms:

$$\top \sqsubseteq \forall \text{hasSTE}.\text{SpatiotemporalExtent} \tag{1}$$

$$\text{TransformationEvent} \sqsubseteq \exists \text{hasSTE}.\text{SpatiotemporalExtent} \tag{2}$$

$$\text{TransformationEvent} \sqsubseteq \forall \text{providesRole}.\text{SourceRole} \tag{3}$$

$$\text{ParticipantRole} \sqsubseteq \leq 1 \text{providesRole}^{-}.\top \tag{4}$$

$$\text{TRU} \sqsubseteq \forall \text{assumesRole}.\text{ParticipantRole} \tag{5}$$

$$\text{ParticipantRole} \sqsubseteq \leq 1 \text{assumesRole}^{-}.\top \tag{6}$$

$$\text{TransformationEvent} \sqsubseteq \forall \text{hasFurtherDocumentation}.\text{DocumentationSource} \tag{7}$$

$$\text{TransformationEvent} \sqsubseteq \exists \text{occursAt}^{-}.\text{TransformationActivity} \tag{8}$$

$$\text{SourceRole} \sqsubseteq \text{ParticipantRole} \tag{9}$$

$$\text{TargetRole} \sqsubseteq \text{ParticipantRole} \tag{10}$$

$$\text{TransformationEvent} \sqsubseteq \exists \text{providesRole}.(\text{SourceRole} \sqcap \exists \text{assumesRole}^{-}.\text{TRU}) \tag{11}$$

$$\text{TransformationEvent} \sqsubseteq \exists \text{providesRole}.(\text{TargetRole} \sqcap \exists \text{assumesRole}^{-}.\text{TRU}) \tag{12}$$

$$\text{TransformationEvent} \sqsubseteq \exists \text{hasPrecondition}.\text{ContainedInRelation} \tag{13}$$

$$\mathsf{TransformationEvent} \sqsubseteq \exists \mathsf{hasPostcondition}.\mathsf{ContainedInRelation} \tag{14}$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by TransformationEvent.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TRU.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentationSource, scoped by TransformationEvent.
8. Inverse Existential: occursAt
9. Subclass: Every SourceRole is a ParticipantRole.
10. Subclass: Every TargetRole is a ParticipantRole.
11. Existential and Inverse Functionality
12. Existential and Inverse Functionality
13. Existential: hasPrecondition
14. Existential: hasPostcondition

3.6.2.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.

3.6.3 Custody Change Event

A Custody Change Event occurs when the custodian of a TRU and/or Container changes. For this event, the classes SourceCustodianRole and TargetCustodianRole, as subclasses of ParticipantRole, are central. The diagram in Figure 3.11 avoids the visual duplication of nodes used in the previous event diagrams.

Axioms:

$$\top \sqsubseteq \forall \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \tag{1}$$
$$\mathsf{CustodyChangeEvent} \sqsubseteq \exists \mathsf{hasSTE}. \tag{2}$$
$$\mathsf{CustodyChangeEvent} \sqsubseteq \forall \mathsf{providesRole}.\mathsf{ParticipantRole} \tag{3}$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{providesRole}^{-}.\top \tag{4}$$
$$\mathsf{TRU} \sqsubseteq \forall \mathsf{assumesRole}.\mathsf{ParticipantRole} \tag{5}$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{assumesRole}^{-}.\top \tag{6}$$
$$\mathsf{CustodyChangeEvent} \sqsubseteq \forall \mathsf{hasFurtherDocumentation}.\mathsf{DocumentationSource} \tag{7}$$
$$\mathsf{SourceCustodianRole} \sqsubseteq \mathsf{ParticipantRole} \tag{8}$$
$$\mathsf{TargetCustodianRole} \sqsubseteq \mathsf{ParticipantRole} \tag{9}$$
$$\mathsf{CustodyChangeEvent} \sqsubseteq \exists \mathsf{providesRole}.(\mathsf{SourceCustodianRole} \sqcap \exists \mathsf{assumesRole}^{-}.\mathsf{Agent}) \tag{10}$$

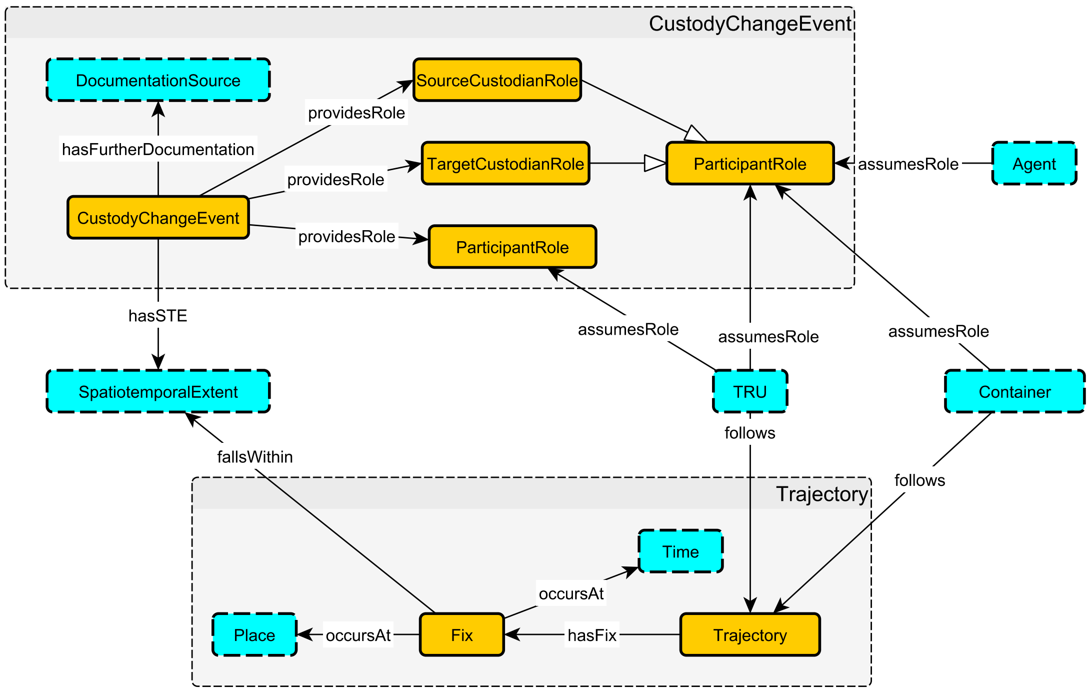

Figure 3.11: Schema Diagram for the CustodyChangeEvent module

$$\mathsf{CustodyChangeEvent} \sqsubseteq \exists \mathsf{providesRole}.(\mathsf{TargetCustodianRole} \sqcap \exists \mathsf{assumesRole}^{-}.\mathsf{Agent}) \tag{11}$$

$$\mathsf{CustodyChangeEvent} \sqsubseteq \leqslant 1 \mathsf{providesRole}.\mathsf{SourceCustodianRole} \tag{12}$$

$$\mathsf{CustodyChangeEvent} \sqsubseteq \leqslant 1 \mathsf{providesRole}.\mathsf{TargetCustodianRole} \tag{13}$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by Custody-ChangeEvent.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TRU.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentationSource, scoped by CustodyChangeEvent.
8. Subclass: Every SourceCustodianRole is a ParticipantRole.
9. Subclass: Every TargetCustodianRole is a ParticipantRole.
10. Existential and Inverse Functionality
11. Existential and Inverse Functionality
12. Functionality
13. Functionality

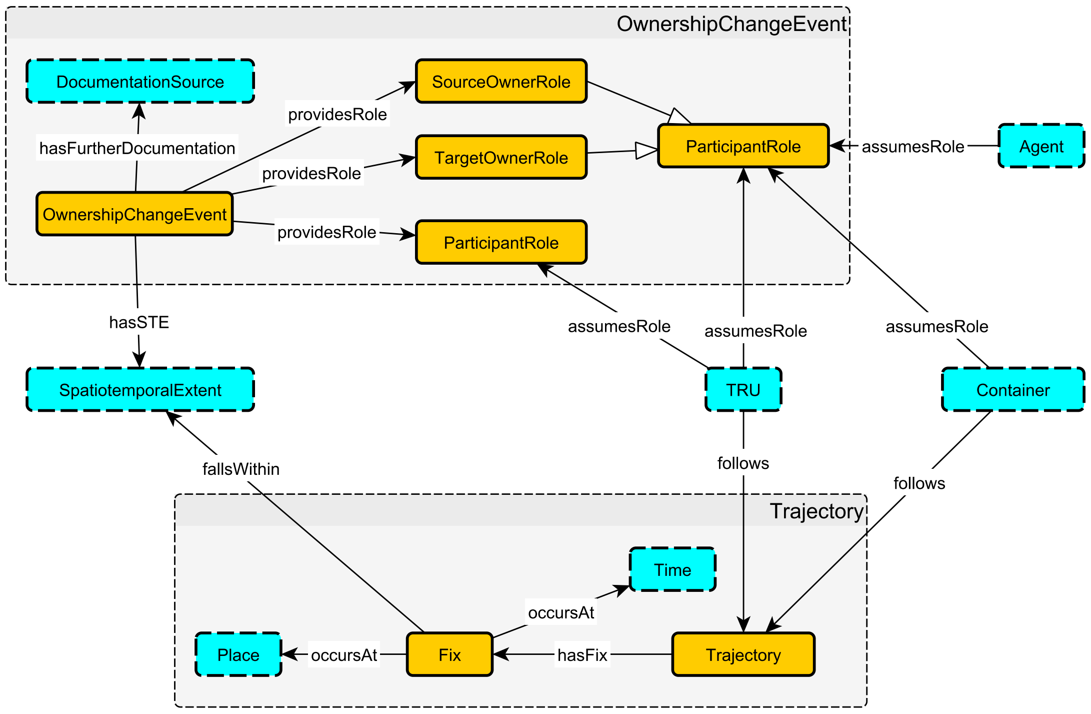

Figure 3.12: Schema Diagram for the OwnerChangeEvent module

3.6.3.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.
- ContainedInRelations are not depicted as they are considered unchanged by this type of Event.

3.6.4 Ownership Change Event

An Ownership Change Event occurs when the owner of a TRU and/or Container changes. For this event, the classes SourceOwnerRole and TargetOwnerRole, as subclasses of ParticipantRole, are central. The diagram in Figure 3.12 avoids the visual duplication of nodes used in the previous event diagrams.

Axioms:

$$\top \sqsubseteq \forall \text{hasSTE.SpatiotemporalExtent} \quad (1)$$

$$\text{OwnerChangeEvent} \sqsubseteq \exists \text{hasSTE.SpatiotemporalExtent} \quad (2)$$

$$\text{OwnerChangeEvent} \sqsubseteq \forall \text{providesRole.ParticipantRole} \quad (3)$$

$$\text{ParticipantRole} \sqsubseteq \leq 1 \text{providesRole}^{-}.\top \quad (4)$$

$$\text{TRU} \sqsubseteq \forall \text{assumesRole.ParticipantRole} \quad (5)$$

$$
\begin{aligned}
\mathsf{ParticipantRole} &\sqsubseteq {\leq}1\mathsf{assumesRole}^{-}.\top && (6)\\
\mathsf{OwnerChangeEvent} &\sqsubseteq \forall\mathsf{hasFurtherDocumentation.DocumentationSource} && (7)\\
\mathsf{SourceOwnerRole} &\sqsubseteq \mathsf{ParticipantRole} && (8)\\
\mathsf{TargetOwnerRole} &\sqsubseteq \mathsf{ParticipantRole} && (9)\\
\mathsf{OwnerChangeEvent} &\sqsubseteq \exists\mathsf{providesRole}.(\mathsf{SourceOwnerRole} \sqcap \exists\mathsf{assumesRole}^{-}.\mathsf{Agent}) && (10)\\
\mathsf{OwnerChangeEvent} &\sqsubseteq \exists\mathsf{providesRole}.(\mathsf{TargetOwnerRole} \sqcap \exists\mathsf{assumesRole}^{-}.\mathsf{Agent}) && (11)\\
\mathsf{OwnerChangeEvent} &\sqsubseteq {\leq}1\mathsf{providesRole.SourceOwnerRole} && (12)\\
\mathsf{OwnerChangeEvent} &\sqsubseteq {\leq}1\mathsf{providesRole.TargetOwnerRole} && (13)
\end{aligned}
$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by OwnershipChangeEvent.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TRU.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentationSource, scoped by OwnerChangeEvent.
8. Subclass: Every SourceOwnerRole is a ParticipantRole.
9. Subclass: Every TargetOwnerRole is a ParticipantRole.
10. Existential and Inverse Functionality
11. Existential and Inverse Functionality
12. Functionality
13. Functionality

3.6.4.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.
- ContainedInRelations are not depicted as they are considered unchanged by this type of Event.

3.6.5 Transport Event

A Transport Event occurs when a TRU is moved (being shipped) from one location to another location over an elapsed period of time. A container containing this TRU moves a long a trajectory which is entirely within the spatiotemporal extent of the Transport Event. It is important to note that ObservationEvents may happen concurrently, e.g. for measuring temperatures.

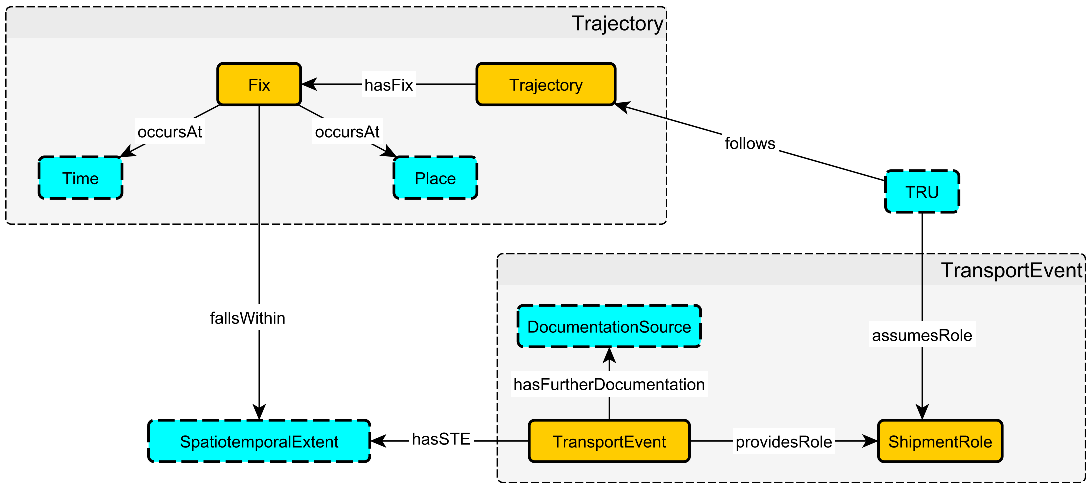

Figure 3.13: Schema Diagram for the TransportEvent module

Axioms:

$$\top \sqsubseteq \forall \text{hasSTE.SpatiotemporalExtent} \quad (1)$$
$$\text{TransportEvent} \sqsubseteq \exists \text{hasSTE.SpatiotemporalExtent} \quad (2)$$
$$\text{TransportEvent} \sqsubseteq \forall \text{providesRole.ParticipantRole} \quad (3)$$
$$\text{ParticipantRole} \sqsubseteq \leq 1 \text{providesRole}^{-}.\top \quad (4)$$
$$\text{TRU} \sqsubseteq \forall \text{assumesRole.ParticipantRole} \quad (5)$$
$$\text{ParticipantRole} \sqsubseteq \leq 1 \text{assumesRole}^{-}.\top \quad (6)$$
$$\text{TransportEvent} \sqsubseteq \forall \text{hasFurtherDocumentation.DocumentationSource} \quad (7)$$
$$\text{ShipmentRole} \sqsubseteq \text{ParticipantRole} \quad (8)$$
$$\text{TransportEvent} \sqsubseteq \exists \text{providesRole.}(\text{ShipmentRole} \sqcap \exists \text{assumesRole}^{-}.\text{TRU}) \quad (9)$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by TransportEvent.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TRU.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentSource, scoped by TransportEvent.
8. Subclass: Every TransportEvent is a ParticipantRole.
9. existential and inverse functionality

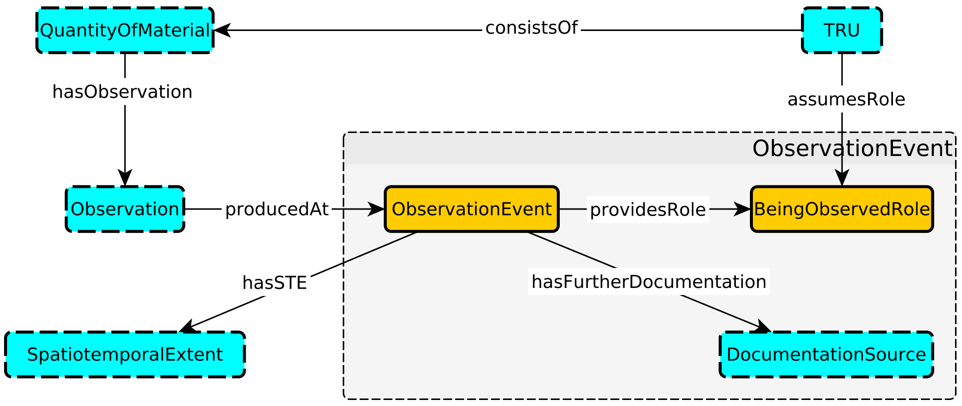

Figure 3.14: Schema Diagram for the ObservationEvent module

3.6.5.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.
- ContainedInRelations are not depicted as they are considered unchanged by this type of Event.

3.6.6 Observation Event

An Observation Event occurs when a Measurement or other result on a TRU is captured at a specific point in time.

Axioms:

$$\top \sqsubseteq \forall \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \quad (1)$$
$$\mathsf{ObservationEvent} \sqsubseteq \exists \mathsf{hasSTE}.\mathsf{SpatiotemporalExtent} \quad (2)$$
$$\mathsf{ObservationEvent} \sqsubseteq \forall \mathsf{providesRole}.\mathsf{ParticipantRole} \quad (3)$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{providesRole}^{-}.\top \quad (4)$$
$$\mathsf{TRU} \sqsubseteq \forall \mathsf{assumesRole}.\mathsf{ParticipantRole} \quad (5)$$
$$\mathsf{ParticipantRole} \sqsubseteq \leq 1 \mathsf{assumesRole}^{-}.\top \quad (6)$$
$$\mathsf{ObservationEvent} \sqsubseteq \forall \mathsf{hasFurtherDocumentation}.\mathsf{DocumentationSource} \quad (7)$$
$$\mathsf{BeingObservedRole} \sqsubseteq \mathsf{ParticipantRole} \quad (8)$$
$$\mathsf{Measurement} \sqsubseteq \forall \mathsf{producedAt}.\mathsf{ObservationEvent} \quad (9)$$
$$\mathsf{ObservationEvent} \sqsubseteq \exists \mathsf{producedAt}^{-}.\mathsf{Measurement} \quad (10)$$
$$\mathsf{ObservationEvent} \sqsubseteq \exists \mathsf{providesRole}.(\mathsf{BeingObservedRole} \sqcap \exists \mathsf{assumesRole}^{-}.\mathsf{TRU}) \quad (11)$$
$$\mathsf{TRU}(x) \land \mathsf{consistsOf}(x,y) \land \mathsf{QuantityOfMaterial}(y) \land \mathsf{hasMeasurement}(y,z) \land \mathsf{Measurement}(z) \land \mathsf{producedAt}(z,w) \land \mathsf{ObservationEvent}(w)$$

$$\rightarrow \exists v(\mathsf{BeingObservedRole}(v) \wedge \mathsf{providesRole}(w, v) \wedge \mathsf{assumesRole}(x, v)) \tag{12}$$

Axiom Explanations:

1. Range: The range of hasSTE is SpatiotemporalExtent.
2. Existential: hasSTE
3. Scoped Range: The range of providesRole is ParticipantRole, scoped by ObservationEvent.
4. Inverse Scoped Functionality: providesRole
5. Scoped Range: The range of assumesRole is ParticipantRole, scoped by TRU.
6. Inverse Scoped Functionality: assumesRole
7. Scoped Range: The range of hasFurtherDocumentation is DocumentationSource, scoped by ObservationEvent.
8. Subclass: Every BeingObservedRole is a ParticipantRole.
9. Scoped Range: The range of producedAt is ObservationEvent, scoped by Measurement.
10. Inverse Existential: producedAt
11. existential and inverse functionality
12. existential rule

3.6.6.1 Remarks:

- The following should be the case: If an Event E hasSTE S, then if a TRU assumes a role for this event, any container this TRU is currently contained in, should have a fix that fallsWithin S.
- Axiom 12 is given as an existential rule. This rule is translatable into OWL DL, however this requires several OWL axioms and will render providesRole and assumesRole to be non-simple [Krisnadhi et al., 2011].
- ContainedInRelations are not depicted as they are considered unchanged by this type of Event.

4 Putting It All together

Figure 4.1 provides a relatively complete overview of the assembled modules.

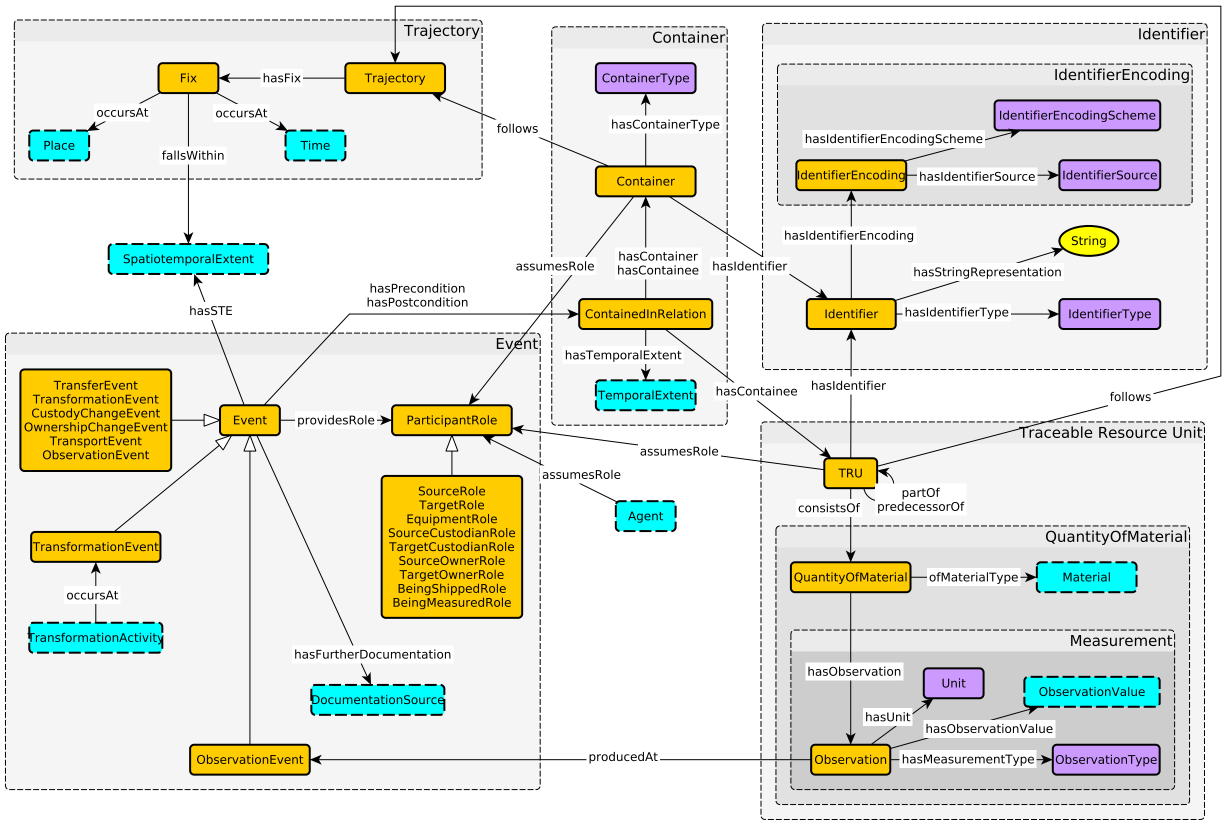

Figure 4.1: Schema Diagram for the combined modules

5 Open Issues

We list some open issues that we may pick up going forward to extend the ontology.

- Appropriate class disjointness axioms could be added.

Undefined Modules

- SpatioTemporalExtent
- Place
- Time
- TemporalExtent (temporal scope)
- StuffType (in Quantity of Stuff)
- MeasurementValue
- EquipmentRole (Event)
- DocumentationSource (Event)
- Agent
- EquipmentRole (TransferEvent)
- Value (Measurement)
- Weather Events (and things orthogonal to food safety)

Undefined Controlled Vocabularies

- ContainerType
- IdentifierType
- IdentifierSource
- IdentifierEncodingScheme
- MeasurementType
- Unit (as in Measurement)

Bibliography

[Eberhart et al., 2021] Eberhart, A., Shimizu, C., Chowdhury, S., Sarker, M. K., and Hitzler, P. (2021). Expressibility of OWL axioms with patterns. In Verborgh, R., Hose, K., Paulheim, H., Champin, P., Maleshkova, M., Corcho, Ó., Ristoski, P., and Alam, M., editors, *The Semantic Web – 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*, volume 12731 of *Lecture Notes in Computer Science*, pages 230–245. Springer.

[Hitzler, 2021] Hitzler, P. (2021). A review of the semantic web field. *Commun. ACM*, 64(2):76–83.

[Hitzler et al., 2016] Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., and Presutti, V., editors (2016). *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*, volume 25 of *Studies on the Semantic Web*. IOS Press.

[Hitzler et al., 2017] Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A. A., and Presutti, V. (2017). Towards a simple but useful ontology design pattern representation language. In Blomqvist, E., Corcho, Ó., Horridge, M., Carral, D., and Hoekstra, R., editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Hitzler and Krisnadhi, 2016] Hitzler, P. and Krisnadhi, A. (2016). On the roles of logical axiomatizations for ontologies. In Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., and Presutti, V., editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 73–80. IOS Press.

[Hitzler et al., 2012] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P., and Rudolph, S., editors (11 December 2012). *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation. Available at http://www.w3.org/TR/owl2-primer/.

[Hitzler et al., 2010] Hitzler, P., Krötzsch, M., and Rudolph, S. (2010). *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.

[Hitzler and Shimizu, 2018] Hitzler, P. and Shimizu, C. (2018). Modular ontologies as a bridge between human conceptualization and data. In Chapman, P., Endres, D., and Pernelle, N., editors, *Graph-Based Representation and Reasoning – 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings*, volume 10872 of *Lecture Notes in Computer Science*, pages 3–6. Springer.

[Karima et al., 2017] Karima, N., Hammar, K., and Hitzler, P. (2017). How to document ontology design patterns. In Hammar, K., Hitlzer, P., Lawrynowicz, A., Krisnadhi, A., Nuzzolese, A., and Solanki, M., editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 15–28. IOS Press / AKA Verlag.

[Krisnadhi and Hitzler, 2016] Krisnadhi, A. and Hitzler, P. (2016). Modeling with ontology design patterns: Chess games as a worked example. In Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., and Presutti, V., editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 3–21. IOS Press.

[Krisnadhi et al., 2016] Krisnadhi, A., Karima, N., Hitzler, P., Amini, R., Rodríguez-Doncel, V., and Janowicz, K. (2016). Ontology design patterns for linked data publishing. In Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., and Presutti, V., editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 201–232. IOS Press.

[Krisnadhi et al., 2011] Krisnadhi, A., Maier, F., and Hitzler, P. (2011). OWL and Rules. In Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., and Patel-Schneider, P. F., editors, *Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures*, volume 6848 of *Lecture Notes in Computer Science*, pages 382–415. Springer, Heidelberg.

[Sarker et al., 2016] Sarker, M. K., Krisnadhi, A. A., and Hitzler, P. (2016). OWLAx: A Protégé plugin to support ontology axiomatization through diagramming. In Kawamura, T. and Paulheim, H., editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016.*, volume 1690 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Shimizu, 2017] Shimizu, C. (2017). Rendering OWL in LaTeX for improved readability: Extensions to the OWLAPI. Master's thesis, Department of Computer Science and Engineering, Wright State University, Dayton, Ohio.

[Shimizu et al., 2019a] Shimizu, C., Eberhart, A., Karima, N., Hirt, Q., Krisnadhi, A., and Hitzler, P. (2019a). A method for automatically generating schema diagrams for OWL ontologies. In Villazón-Terrazas, B. and Hidalgo-Delgado, Y., editors, *Knowledge Graphs and Semantic Web – First Iberoamerican Conference, KGSWC 2019, Villa Clara, Cuba, June 23-30, 2019, Proceedings*, volume 1029 of *Communications in Computer and Information Science*, pages 149–161. Springer.

[Shimizu et al., 2023] Shimizu, C., Hammar, K., and Hitzler, P. (2023). Modular ontology modeling. *Semantic Web*, 14(3):459–489.

[Shimizu et al., 2019b] Shimizu, C., Hirt, Q., and Hitzler, P. (2019b). MODL: A modular ontology design library. In Janowicz, K., Krisnadhi, A. A., Villalón, M. P., Hammar, K., and Shimizu, C., editors, *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019*, volume 2459 of *CEUR Workshop Proceedings*, pages 47–58. CEUR-WS.org.

[Shimizu et al., 2020] Shimizu, C., Hitzler, P., Hirt, Q., Rehberger, D., Estrecha, S. G., Foley, C., Sheill, A. M., Hawthorne, W., Mixter, J., Watrall, E., Carty, R., and Tarr, D. (2020). The enslaved ontology: Peoples of the historic slave trade. *Journal of Web Semantics*, 63:100567.