SOSA-SHACL: Shapes Constraint for the Sensor, Observation, Sample, and Actuator Ontology

Rui Zhu
University of California, Santa Barbara
Santa Barbara, CA, USA
ruizhu@ucsb.edu

Cogan Shimizu
Kansas State University
Manhattan, KS, USA
coganmshimizu@ksu.edu

Shirly Stephen
University of California, Santa Barbara
Santa Barbara, CA, USA
shirlystephen@ucsb.edu

Lu Zhou
Kansas State University
Manhattan, KS, USA
luzhou@ksu.edu

Ling Cai
University of California, Santa Barbara
Santa Barbara, CA, USA
lingcai@ucsb.edu

Gengchen Mai
Stanford University
Stanford, CA, USA
maigch@cs.stanford.edu

Krzysztof Janowicz
University of California, Santa Barbara
Santa Barbara, CA, USA
janowicz@ucsb.edu

Mark Schildhauer
National Center for Ecological Analysis & Synthesis
Santa Barbara, CA, USA
schild@nceas.ucsb.edu

Pascal Hitzler
Kansas State University
Manhattan, KS, USA
hitzler@ksu.edu

ABSTRACT

The explosive growth of the Linked Data on the Web has greatly facilitated collecting data from remote sensors, from air quality sensors spread out across a city, to seismograph stations spread across the entire world. Integrating these heterogeneous data can be quite challenging; however one can achieve this through the use of available W3C standards to create a knowledge graph. For this use case, the W3C also provides a standard, the Sensor, Observation, Sample, Actuator (SOSA) Ontology, that allows for the semantic encoding of sensors and their observations. However, even with the guidance of this standard, it may be difficult to produce a *correct* graph with high fidelity from heterogeneous sources. In this paper we present a set of (data) shape constraints, called SOSA-SHACL, for the SOSA ontology using a data validation language, namely the W3C standard SHACL (Shape Constraint Language). These constraints enable us to evaluate whether the modeled observations in our Knowledge Graph comply with the SOSA recommendations. Furthermore, we show through several case studies how the closed world assumption plays a role in the process of designing such shape constraints, especially as SOSA is based on the open world assumption.

CCS CONCEPTS

• **Information systems** → **Graph-based database models**; **Semantic web description languages**.

IJCKG'21, December 6–8, 2021, Virtual Event, Thailand

ACM ISBN 978-1-4503-9565-6/21/12.
https://doi.org/10.1145/3502223.3502235

KEYWORDS

RDF validation; sensors and observations; knowledge graph quality assessment and refinement

ACM Reference Format:
Rui Zhu, Cogan Shimizu, Shirly Stephen, Lu Zhou, Ling Cai, Gengchen Mai, Krzysztof Janowicz, Mark Schildhauer, and Pascal Hitzler. 2021. SOSA-SHACL: Shapes Constraint for the Sensor, Observation, Sample, and Actuator Ontology. In *The 10th International Joint Conference on Knowledge Graphs (IJCKG'21), December 6–8, 2021, Virtual Event, Thailand*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3502223.3502235

1 INTRODUCTION

In recent years, the ability to collect and store observations from various sensors has skyrocketed. Unfortunately, most of these observations remain *siloed* from each other. There are many potential ways to integrate these observations, e.g., along space and time, as well as what features of interest are being observed. However the heterogeneity of models for representing sensor data typically makes integration quite difficult. Knowledge graphs can facilitate the integration process, by leveraging relevant W3C standards in order to provide consistent syntactical representation and semantic interpretation of the data.

Two standards in particular, the Semantic Sensor Network Ontology (SSN) [3, 11] and the Sensor, Observation, Sample, Actuator (SOSA) Ontology [9], which is a lightweight version of SSN, are ideal for modeling heterogeneous and multi-sourced observations according to a common schema. Indeed, our overarching use case, KnowWhereGraph,[1] utilizes SSN/SOSA to integrate environmental observations from over 27 different datasets from 17 distinct data sources in order to provide highly detailed context regarding events and processes that have happened in particular regions of the Earth (which we call *area briefings*). This has required careful attention to

[1] http://knowwheregraph.org/

Table 1: Namespaces used in this paper

Prefix	Name	URI
sosa:	SOSA	http://w3.org/ns/sosa/
geo:	GeoSPARQL	http://opengis.net/ont/geosparql/
cdt:	Custom Datatype	https://w3id.org/cdt/
xsd:	XML Schema	http://www.w3.org/2001/XMLSchema#
sf:	Simple Feature	http://www.opengis.net/ont/sf#
kwg-ont:	KWG Ontology	http://www.knowwheregraph.org/lod/ont/
kwgr:	KWG Resource	http://www.knowwheregraph.org/lod/resource/

retaining the original foci and value of the original, diverse datasets, while enhancing their utility through integration with other disparate resources. This has been especially challenging when materialization tasks, i.e. converting raw data into graph-ready format, were carried out by a large, distributed team. Materializing a knowledge graph, or triplification, is the act of converting flat data (e.g., CSV or shapefiles) into *subject-predicate-object* triples that conform to some schema. Materialization is largely a software engineering endeavor, as opposed to the schema development, which falls under knowledge engineering. Consequently, there can be a conflict in base assumptions—namely, the differences between the Open World Assumption (OWA) and Closed World Assumption (CWA), as there may not always be a straightforward translation from one to another. For example, in an ontology under OWA, there may be an axiom that specifies the existence of some filler for some property (called an existential axiom). However, it may not be the case that this filler would always be recorded or measured by the sensor. Ontologically, we know that it *exists*, but empirically, we do not. Thus, when we are materializing the graph, it may be difficult to identify errors or omissions in the graph by using only the knowledge graph's schema, as specified in an ontology, as a guide. This error checking process is known as *validation*. The W3C also provides a recommendation for data validation—The Shapes Constraint Language (SHACL) [8, 10]. This language allows us to specify the shape of the graph, which is also a graph itself. That is, what triples should be present, in what quantity, and, conversely, what definitely *should not* be there.

In this paper, we have developed a comprehensive set of SHACL shapes that can be used to validate knowledge graphs that utilize SOSA, which we call SOSA-SHACL (Section 4). These shapes are built based on our experience of modeling numerous environment-related sensors and observations, including data themes of air quality, earthquakes, and wildfires (Section 3). The advantage of using the closed world assumption over SOSA's inherent open world assumption to validate the graph is discussed in Section 5. Beyond shapes tailored for SOSA, Section 6 further exemplifies the use of SHACL to define data-dependent shapes together with SOSA. Moreover, background and related work are discussed in Section 2, and Section 7 concludes by pointing to future research directions. The related prefixes used in this paper are summarized in Table 1. We also note that for clarity and brevity, when we refer to characteristics of an sosa:Observation, these also tend to hold for other aspects of SOSA (i.e., samplings, actuations, and observation collections).

2 BACKGROUND AND RELATED WORK

This section provides backgrounds about the Sensor, Observation, Sample, and Actuator (SOSA) ontology and the Shapes Constraint Language (SHACL). Moreover, we summarize existing applications of using SHACL.

2.1 Sensor, Observation, Sample, and Actuator (SOSA) Ontology

The SOSA ontology[2] [9] is a W3C/OGC standard developed using a subset of SSN's entities [3, 11] and contains concepts, properties, and annotations for describing sensors, observations, actuators, samples, features of interest, observable properties, and observation procedures. It is extensively used in the Semantic Web for applications ranging from representing data-streams [1, 5], smart-city data [6, 7], communication technologies and protocols [2, 14], and various kinds of geospatial information [12, 13, 15, 16]. The key uses of the ontology are: (1) linking observations (sosa:Observation) to sensors (sosa:Sensor) that measured them and the property (sosa:ObservableProperty) that was observed; (2) annotating observation measurements with the observed value (sosa:Result), the entity that was observed (sosa:FeatureOfInterest), or the event that originated the measurement (sosa:Stimulus); and (3) aggregating observations into observation collections (sosa:ObservationCollection) using aggregators such as spatial properties (of the observed feature or event), temporal properties (sosa:resultTime and sosa:phenomenonTime), and the observed property [17]. Figure 1 shows a schema diagram for the classes immediately adjacent to sosa:Observation.

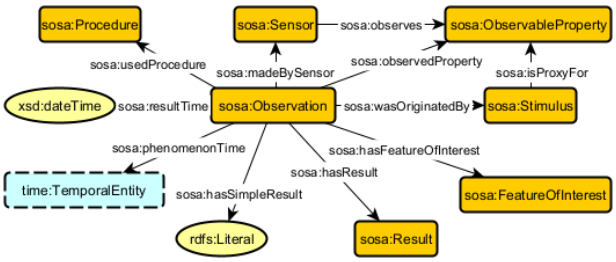

Figure 1: Fragment of SOSA's schema diagram with a focus on sosa:Observation class.

2.2 Shapes Constraint Language (SHACL)

The Shapes Constraint Language (SHACL) is a W3C standard for validating RDF graphs; the standard describes two graphs, the *data graph* and the *shapes graph*. The RDF graph to be validated is the *data graph*, while the set of constraints—expressed in RDF as well—that describe the shape of the graph, is collectively known as the *shapes graph*. This shapes graph formally describes the set of conditions the data graph needs to satisfy. SHACL provides a set of built-in core constraints that are often used in validating graphs, such as sh:class to restrict the focus node to be from a specific class, sh:minCount to restrict the number of value nodes for a focus node/property, and sh:or to restrict the logical "or" relation between several shapes.

[2] https://www.w3.org/TR/vocab-ssn/ and the extension https://www.w3.org/TR/vocab-ssn-ext/

In general, there are two types of SHACL shapes according to the target types: if the target is a node, it refers to be a node shape (sh:NodeShape); if the target is a property, it is a property shape (sh:PropertyShape). A SHACL validation engine uses shapes graphs to validate data graphs. If there are some parts of the data graphs violating conditions specified by the shapes graphs, an error will be added to the SHACL validation report. As such, we can use SHACL to assess and refine the data quality of any materialized RDF graphs. Many modern triple stores support SHACL validation, including GraphDB[3], Apache Jena[4], and Neo4j[5].

2.3 Applications of SHACL

So far, SHACL has been used to design validating shapes for ontologies such as Schema.org[6], W3C Provenance (PROV)[7], and GeoSPARQL [4]. In these use cases, SHACL has been leveraged to enforce the cardinality of a relation, to improve the interoperability between different systems, to declare metadata templates, and so forth. This work, in contrast, designs SHACL constraints for SOSA ontology. More importantly, our work emphasizes the role that the CWA plays in using SHACL to design shapes to assess and refine the quality of real-world environmental observations. Meanwhile, it is also worth noting that similar to SHACL, ShEx is an alternative designed to evaluate RDF graphs. A systematic comparision of these two can be found in [8].

3 SAMPLE DATASETS

The motivation of designing SOSA-SHACL is primarily based on our experience of using SOSA ontology to build a knowledge graph that integrates data from different environment-related sensors and observations. This section showcases three datasets, which are related to environmental themes of air quality, earthquakes, and wildfires, to empirically explain the need of SOSA-SHACL so as to assess and refine the graph quality. More specifically, we introduce the provenance of datasets, explain schema modeling using SOSA ontology, and discuss limitations of only relying on SOSA to validate the graph. These discussions further motivate the introduction of SOSA-SHACL.

3.1 Air Quality Observations

The Air Data[8] is an example of observation data that is made available by the Environmental Protection Agency (EPA) to report daily pollutant measurements as recorded at air quality monitoring stations in the US.

Each row in the raw CSV file contains a single observation of daily pollutant concentration, the Air Quality Index (AQI) value, as well as some attributes such as Parameter Occurrence Code (POC), geographic coordinates of the air quality site, and type of air pollutant measured (e.g., PM10, CO, Pb). By consulting with environmental scientists, the schema as depicted in Figure 2 was developed for this dataset. Specifically, the figure shows measurements of a

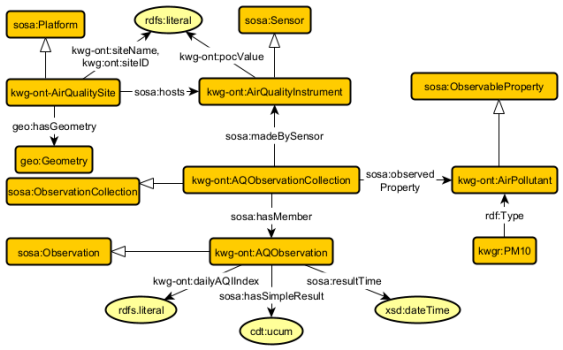

Figure 2: EPA Air Quality Observations Schema Diagram

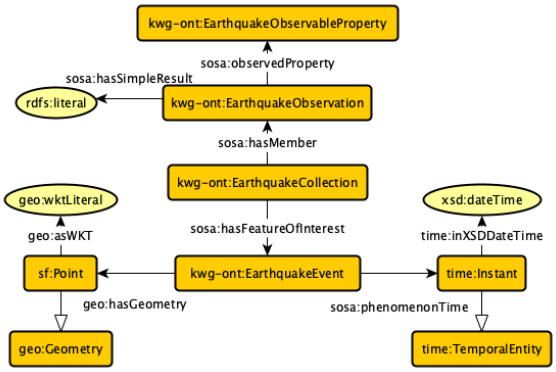

Figure 3: Schema Diagram for the Earthquake Observations Dataset

specific air pollutant (e.g., PM10) as observed by an air quality instrument (a subclass of sosa:Sensor), and the associated observation collection (i.e., kwg-ont:AQObservationCollection as a subclass of sosa:ObservationCollection). Such a collection contains individual observation members (i.e., instances of kwg-ont:AQObservation, which are subclasses of sosa:Observation) that are observed at a specific time (xsd:dateTime).

While materializing Air Data using such a schema, the most common mistake one can make is to collectively regard each daily measurement of PM10, CO, PM2.5, etc. as members of one observation collection. However, since each type of air pollutant is measured by a different sensor (i.e., an instance of kwg-ont:AirQualityInstrument), materializing data in such a manner would result in multiple sensors associated with one observation collection (i.e., an instance of kwg-ont:AQObservationCollection), which violates the constraint that a sosa:ObservationCollection can only be observed by *exactly* one sensor.

[3]https://graphdb.ontotext.com/documentation/free/shacl-validation.html
[4]https://jena.apache.org/documentation/shacl/index.html
[5]https://neo4j.com/labs/neosemantics/4.0/validation/
[6]https://datashapes.org/schema
[7]https://www.w3.org/TR/prov-constraints/
[8]https://www.epa.gov/outdoor-air-quality-data/download-daily-data

3.2 Earthquake Observations

Similar to air quality observations, earthquakes are observed at seismic stations through various sensors. The United States Geological Survey (USGS) API.[9] is a resource for earthquake events that occurred in the US. Each earthquake event contains information on its latitude and longitude as well as 20 attributes defined by the ANSS Comprehensive Earthquake Catalog (ComCat), e.g., magnitude and depth [10].

Based on the nature of the dataset and SOSA ontology, we materialized the sosa:Observation and sosa:ObservationCollection pattern to describe the earthquake event. As depicted in Figure 3, the class kwg-ont:EarthquakeEvent is modeled as a subclass of sosa:FeatureOfInterest in SOSA ontology and a subclass of geo:Feature in GeoSPARQL ontology. Each row in the USGS earthquake dataset (e.g., CSV file) is considered as one kwg-ont:EarthquakeObservation, which is a subclass of sosa:Observation in the SOSA ontology. For example, magnitude is defined as the best available estimate of the earthquake's size, at the time that the event page is created by USGS. As one of the earthquake observations, magnitude has a value (e.g., 4.6), through the property sosa:hasSimpleResults. Besides using observations, some important information which can be used for fast query and further data integration, like location and time are also modeled and stored using sf:Point and time:Instant.

Even with such a schema, we encountered several issues while materializing the data. First of all, there might be cases where we have an instance of kwg-ont:EarthquakeObservationCollection, a subclass of sosa:ObservationCollection, but it is not related to any instances of kwg-ont:EarthquakeObservation, which is a subclass of sosa:Observation through the predicate sosa:hasMember. It is apparently an error that once has being reported, we would have the chance to check the feasibility of the schema and the completeness of the raw data. Moreover, based on SOSA, the domain of the object property sosa:observedProperty has to be either sosa:Observation or sosa:ObservationCollection. However, simply using SOSA to model environmental observations does not prevent one from linking sosa:observedProperty to other classes, e.g., kwg-ont:MajorEarthquakeMeasurement. If a violation can be detected, we can consequently refine the schema either by adding kwg-ont:MajorEarthquakeMeasurement as a subclass of sosa:Observation or by deleting the violating triples from the data.

3.3 Wildfire Observations

Another sample dataset is of wildfires, provided by Monitoring Trends in Burn Severity program (MTBS), which was jointly conducted by the U.S. Geological Survey Center for Earth Resources Observation and Science (EROS) and the USDA Forest Service Geospatial Technology and Applications Center (GTAC)[11]. This dataset[12] records the burn severity and extent of large fires across the US from 1984 to present. Each wildfire event is stored as a row in a shapefile, including event ID, incident name, incident type, fire

[9]https://earthquake.usgs.gov/fdsnws/event/1/
[10]https://earthquake.usgs.gov/earthquakes/feed/v1.0/csv.php
[11]https://mtbs.gov/
[12]https://edcintl.cr.usgs.gov/downloads/sciweb1/shared/MTBS_Fire/data/composite_data/burned_area_extent_shapefile/mtbs_perimeter_data.zip

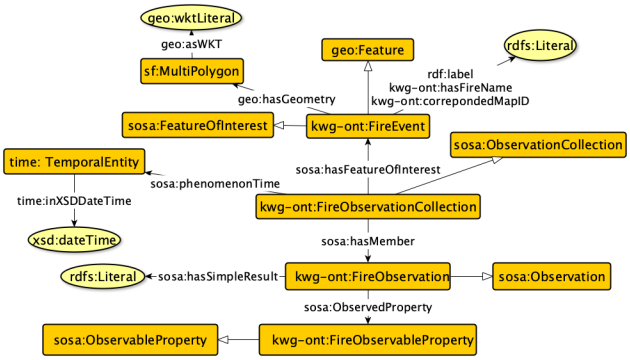

Figure 4: Fire Event Schema Diagram

mapping assessment label, number of acres burned, mean dNBR value, geometry, etc.

We picked 12 attributes out of the total 23 attributes to meet domain experts' interest. Particularly, we divided the selected attributes into three groups: identity attributes (e.g., fire name, map id), classification attributes (e.g., whether a fire event is a wildfire event or a complex fire event), and observation attributes (e.g., number of acres burned). The way of modeling these attributes differed for the three distinct attribute groups. The first two were modeled directly as ordinary RDF triples without using SOSA ontology. In contrast, observation attributes were all modeled with SOSA ontology (Figure 4), especially by using *sosa:ObservationCollection* and *sosa:Observation*. In specific, each fire event was modeled as a subclass of *sosa:FeatureOfInterest* and is associated with a *sosa:ObservationCollection* through the property *sosa:isFeatureOfInterestOf*. Plus, different attributes of a fire event are observation members of their collection construct.

With this schema, there are still challenges when materializing a wildfire dataset. For instance, since in reality observed attributes regarding the same fire event (e.g., number of acres burned and mean dNBR value) are usually observed at the same time, we attach sosa:phenomenonTime to sosa:ObservationCollection rather than to each individual observation. However, one might still relate the sosa:phenomenonTime to either kwg-ont:fireEvent or kwg-ont:FireObservation, a subclass of sosa:Observation in practice. The former is in fact a materialization error that should be detected and fixed, while the latter is more complicated as based on SOSA, an instance of sosa:Observation can be linked with sosa:phenomenonTime as well. So purely based on SOSA ontology, such an inconsistency would not be addressed, and thus a SHACL shape is needed.

4 SOSA-SHACL

To address aforementioned challenges of assessing and refining environmental observations using SOSA, this section introduces the design of SOSA-SHACL. It is designed to constrain the use of SOSA on materializing real world environmental observations. To do so, we leverage both the core and SPARQL-based constraint defined by SHACL. For simple shapes, SHACL's core components such as those designed for cardinality check, value type check, and

logical check, are leveraged. Whereas for those complicated ones (e.g., need to compare the result based on complicated queries), we apply the SPARQL-based constraints defined in SHACL. In total, we introduce 15 individual shapes for the core classes of SOSA (e.g., sosa:Observation and sosa:ObservationCollection), each of which works specifically to a SOSA class, as well as 23 shapes for SOSA properties (e.g., sosa:phenomenonTime and sosa:hasResult), each of which corresponds to a specific SOSA property. The shapes, together with examples and testing codes can be found online, in our our repository[13].

While designing these constraints, we have two strategies. First, we consult original SOSA ontology as well as its official recommendation documents, where we translate those meaningful restrictions (either in OWL or texts) into shapes using SHACL. One example is shapes for sosa:Observation, which is depicted in Figure 5. One of its shapes related to sosa:hasFeatureOfInterest is demonstrated in Listing 1. It states that an observation has to associate with *exactly* one feature of interest.

In addition, our second strategy is to combine the benefits of closed world assumption of SHACL with the nature of environmental observations to propose shapes that have yet been considered, or are hardly implemented, by using the open world assumption-based SOSA ontology. This will be the focus of the next section.

```
sosa-shacl:ObservationConstraint_FOI
 a sh:NodeShape ;
 sh:targetClass sosa:Observation ;
 sh:property
     [
      sh:path sosa:hasFeatureOfInterest ;
      sh:maxCount 1 ;
      sh:minCount 1 ;
      sh:class sosa:FeatureOfInterest ;
     ] ;
```

Listing 1: A shape of sosa:Observation related to sosa:hasFeatureOfInterest

5 VALIDATING "OPEN WORLD" DATA

Even though OWL, as well as RDF Schema, can be used to check the inconsistency of data in some sense, they are fundamentally different from SHACL in terms of validating knowledge graphs. OWL, specifically, defines some restriction properties such as owl:maxCardinality, which can be used to constrain properties of a class. However, these restrictions are not data constraints. They are used for the purpose of inference instead. For instance, if we assert the axiom of owl:maxCardinality to restrict the maximal number of sosa:FeatureOfInterest associated with an instance of sosa:Observation to be 1; if this instance is linked with two sosa:FeatureOfInterest, OWL will not report a violation. Instead, it will infer that the two sosa:FeatureOfInterest are in fact the same (i.e., they are just represented as distinct resources (URI) in the RDF graph, but they both refer to the same entity in reality). In short, OWL follows the "open world" assumption by inferring new knowledge to make the data conform to the defined restrictions, while SHACL obeys the "closed world" assumption directly constraining the target data using defined shapes, and reporting violations if any. The major benefit of using SHACL is not to infer new knowledge, but to determine compliance of data graphs with the constraints specified in SHACL shapes.

In fact, SOSA's formal ontology[14] only declares some basic constraints using OWL, RDF Schema, and Schema.org, including schema:domainIncludes, schema:rangeIncludes, rdfs:range, and owl:inverseOf. Despite the fact that they have been suggested by W3C[15], true "constraints" (rather than axioms) are missing in SOSA's formal representation, such as the constraint: an instance of sosa:Observation must be associated with *one and only one* instance of sosa:ObservableProperty.

Moreover, those W3C recommended constraints sometimes become problematic while being used to validate real-world data as discussed in Section 3. Hence, this section explains the use of SHACL to define more meaningful and flexible shapes to constrain environment-related observations using SOSA. We specifically choose to discuss those constraints that cannot be easily accomplished through other standards (e.g., OWL).

5.1 Temporal Information in SOSA

Spatial and temporal information are essential for environmental observations, as they provide context for interpreting observations. For example, a deadly earthquake that occurs after midnight rather than during the daytime can have drastically different impacts, altering what types of relief strategies can be executed in response. When using SOSA, there are two ways for connecting temporal information, sosa:phenomenonTime (an object property) and sosa:resultTime (a datatype property). The former is defined as the time when an observation applies to a feature of interest while the latter indicates the time when such an observation is completed. Listing 2 and 3 illustrate the existing axioms related to temporal information in SOSA: only very basic domain and range rules are included. However, to empirically validate environmental data requires more sophisticated constraints.

```
sosa:resultTime a owl:DatatypeProperty;
    rdfs:label "result time"@en ;
    schema:domainIncludes sosa:Actuation ;
    schema:domainIncludes sosa:Observation ;
    schema:domainIncludes sosa:Sampling ;
    rdfs:range xsd:dataTime .
```

Listing 2: Declaration of sosa:resultTime

```
sosa:phenomenonTime a owl:ObjectProperty;
    rdfs:label "phenomenon time"@en ;
    schema:domainIncludes sosa:Actuation ;
    schema:domainIncludes sosa:Observation ;
    schema:domainIncludes sosa:Sampling ;
    schema:rangeIncludes time:TemporalEntity .
```

Listing 3: Declaration of sosa:phenomenonTime

First of all, a valid observation (or an actuation and sampling) that is capable of supporting decision making needs to have a temporal scope. Either sosa:phenomenonTime or sosa:resultTime, or even both, can be used to represent such a temporal scope. However, no matter whether it is sosa:phenomenonTime or sosa:resultTime,

[13] https://github.com/KnowWhereGraph/KWG-SHACL

[14] https://www.w3.org/ns/sosa/

[15] https://www.w3.org/TR/vocab-ssn/

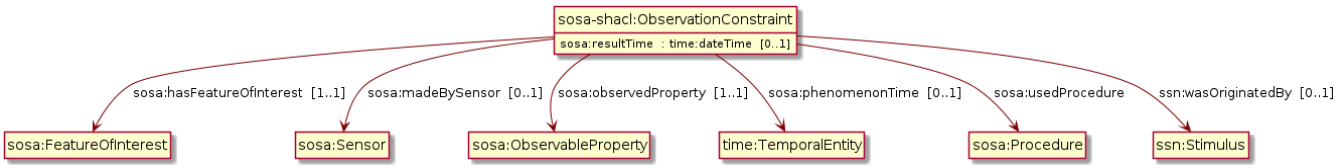

Figure 5: UML diagram for sosa:Observation shapes. Only part of the designed shapes are illustrated.

there should not be more than one corresponding value for either. That is, it is invalid to have measured a single observation of a phenomenon that happened at more than one distinct time. Such a constraint of existence and cardinality cannot be readily expressed in OWL, whereas by using SHACL's core constraints, we can design an observation shape that does so. As Listings 4 and 5 demonstrate, core constraints such as sh:or, sh:minCount, and sh:class are combined to comprise shapes that constrain on (1) there should be at least one piece of temporal information related to an observation, either sosa:phenomenonTime or sosa:resultTime (existential filler); (2) there cannot be more than one value associated with sosa:phenomenonTime or sosa:resultTime (cardinality filler); and (3) the value of sosa:phenomenonTime has to be a time:TemporalEntity, and the value of sosa:resultTime has to be an xsd:dateTime (range filler). It is worth noting that the SOSA ontology itself as shown in Listings 2 and 3 only covers the range filler.

```
sosa-shacl:SOSATimeConstraint_Existential a sh:NodeShape;
    sh:targetClass sosa:Observation ;
    sh:or (
    [
      sh:path sosa:phenomenonTime ;
      sh:minCount 1 ;
    ]
    [
      sh:path sosa:resultTime ;
      sh:minCount 1 ;
    ]
    ).
```

Listing 4: SHACL shape to constrain the existential filler

```
sosa-shacl:SOSATimeConstraint_Cardinality a sh:NodeShape;
    sh:targetClass sosa:Observation ;
    sh:property
    [
      sh:path sosa:phenomenonTime ;
      sh:maxCount 1 ;
      sh:class time:TemporalEntity ;
    ];
    sh:property
    [
      sh:path sosa:resultTime ;
      sh:maxCount 1 ;
      sh:classType xsd:dateTime ;
    ] .
```

Listing 5: SHACL shape to constrain the cardinality and range filler

More interestingly, when both sosa:phenomenonTime and sosa:resultTime are used, there should be an additional constraint that the result time will typically be later than the phenomenon time, due to latency in sensor actuation and value reporting. Such a constraint becomes valuable in detecting results from model forecasting, as opposed to actual observations. For example, a simulation model might predict air quality measures (e.g., PM2.5) at a monitoring station for the next 10 days. The result time is when the prediction is made (the time when the model, which can be regarded as a sensor here, returns the result) and the phenomenon time (when the prediction applies to) would be up to 10 days after the result time. An observation whose phenomenon time is after its result time, consequently, is either a "forecasted" data value, or an error. Note that phenomenon time and result time still frequently refer to the same time in many environmental datasets. For example, the time when an earthquake occurs is also the time when the sensor returns the result (i.e., the magnitude of the seismic waves). In practice, only one of these two SOSA temporal properties (mostly sosa:phenomenonTime) is typically recorded in the graph, to reduce redundancy. It is worth noting, however, that sosa:phenomenonTime can reference a temporal interval, while sosa:resultTime is always recorded as an instant in time. Finally, Listing 6 shows such a SPARQL-based shape constraint that sosa:phenomenonTime has to be equal or prior to sosa:resultTime[16].

```
sosa-shacl:SOSATimeConstraint_Comparision
    a sh:NodeShape ;
    sh:targetClass sosa:Observation ;
    sh:sparql [
    sh:prefixes sosa: ;
    sh:select """
  SELECT ?this ?phenTime_literal
  WHERE {
   ?this sosa:phenomenonTime ?phenTime ;
      sosa:resultTime ?resultTime_literal .
   ?phenTime time:inXSDDateTime ?phenTime_literal .
   FILTER (?phenTime_literal > ?resultTime_literal)
   }
  """ ;
] .
```

Listing 6: SHACL shape to constrain the relation between sosa:phenomenonTime and sosa:resultTime.

5.2 Observation Result

In the SOSA ontology, there are two ways to specify the result of an observation: sosa:hasResult and sosa:hasSimpleResult. The former is an object property designed to indicate the observed result (sosa:Result) in regards to a corresponding observable property (sosa:ObservableProperty) of an observation. The instance of sosa:Result is commonly modeled as a blank node for environmental observations, and external ontologies, such as QUDT[17],

[16]There are other ways to write this constraint, which are included in the Github repository.
[17]http://www.qudt.org/

can be applied to formally represent the quantity as well as its unit. To simplify the representation, particularly when the observation's result is simply a literal node of numeric value, the property sosa:hasSimpleResult can be used.

SOSA specifies that sosa:Observation has minimally one sosa:Result linked through sosa:hasResult. However, this tends to not be true in practice, especially when sosa:hasSimpleResult is used as well. That is, when materializing the graph, one may choose to only materialize one or the other, depending on how the raw data is provided to them. As such, we do not prescribe to both existentials, but instead use the shape in Listing 7, which indicates that a valid observation has a filler in either sosa:hasResult or sosa:hasSimpleResult. Having a filler for both is invalid. Moreover, the shape also declares that an observation can have maximally one sosa:hasSimpleResult in order to reduce the ambiguity of the simple result value. Such a consideration is due to the fact that the range of sosa:hasSimpleResult is not defined so that any types of data property can be used for the result. For instance, the impacted area (an instance of sosa:ObservableProperty) of a wildfire can be represented as either "120 acres" or "485623 m^2", and if both exist in the data, the comparison between different wildfires in terms of the impacted areas becomes impossible. It will be less likely an issue for sosa:hasResult though because by representing the value as an instance of sosa:Result together with QUDT, for example, it enables the conversion between different unit standards.

```
sosa-shacl:SOSAResultConstraint
     a sh:NodeShape ;
     sh:targetClass sosa:Observation ;
     sh:xone  (
        [
          sh:path sosa:hasResult ;
          sh:minCount 1 ;
          sh:class sosa:Result ;
         ]
         [
          sh:path sosa:hasSimpleResult ;
          sh:minCount 1 ;
          sh:maxCount 1 ;
         ]
      ) .
```

Listing 7: SHACL shape to constrain the result of an observation.

5.3 Implicit Sensor

Even though SOSA recommends that an observation must be made by exactly one sensor (sosa:Sensor), we find such a constraint to be too strong for environmental observations. In practice, the sensor that makes an observation is often not explicitly indicated by the raw data. For example, earthquake data collected by USGS does not provide details of the sensor that collects it. Similar is true for the aforementioned wildfire data. In addition, for forecasting observations, such as the prediction of PM2.5, the sensor becomes implicit because there is in fact no physically located sensor being deployed to collect such predictions; they are rather computed based on mathematical models. To address such an ontological inconsistency, we design the sensor-based shape for observation by only constraining that there is maximally one associated sensor and it must be an instance of sosa:Sensor. Listing 8 demonstrates the shape. The same shapes pattern is used on the sosa:usedProcedure and ssn:wasOriginatedBy properties as well, indicating that a valid observation does not necessarily link with an explicit procedure (sosa:Procedure) or stimulus (sosa:Stimulus). In fact, most raw data of environmental observations do not explicitly provide this information.

```
sosa-shacl:SOSASensorConstraint
     a sh:NodeShape ;
     sh:targetClass sosa:Observation ;
      sh:property
      [
          sh:path sosa:madeBySensor ;
          sh:maxCount 1 ;
          sh:class sosa:Sensor ;
      ] ;
```

Listing 8: SHACL shape to constrain the sensor of an observation.

5.4 Observation and Its Collection

As discussed in [17], the class of sosa:ObservationCollection can be used to reduce redundancy when expressing environmental observations in graph format. For example, multiple individual wildfire observations might relate to the same feature of interest – an instance of kwg-ont:FireEvent, so leveraging the collection construct integrating observations of the same feature of interest can reduce the number of triples without loss of information (see Figure 4). More robust SHACL shapes are required, however, to perform validation of SOSA observation collections. More concretely, previously discussed sosa-shacl:ObservationConstraint_FOI shape (Listing 1) has to be adjusted in order to consider the situation that a collection of observations might share the same filler for the property sosa:hasFeatureOfInterest. In such a case, the individual observation itself should not be linked to any sosa:FeatureOfInterest anymore. Listing 9 shows such a shape by using SPARQL-based SHACL, capturing the constraint that either an observation or its corresponding observation collection has to be linked to a feature of interest, but not both. Similarly, other constraints, such as those regarding sosa:observedProperty, sosa:phenomenonTime, and sosa:hasResult have their collection shape version as well.

```
sosa-shacl:Observation_Collection_FearureOfInterest
 a sh:NodeShape ;
 sh:targetClass sosa:Observation ;
 sh:or [
   sh:sparql [
     sh:prefixes sosa: ;
     sh:select """
    SELECT { ?this sosa:hasFeatureOfInterest ?foi  }
    WHERE {
    ?this a sosa:Observation .
   NOT EXISTS { ?this sosa:hasFeatureOfInterest ?x . }
      ?oc sosa:hasMember+ ?this .
       NOT EXISTS {?oc sosa:hasFeatureOfInterest ?foi . }
     } """ ;] ;
  sh:sparql [
    sh:prefixes sosa: ;
    sh:select """
   SELECT { ?this sosa:hasFeatureOfInterest ?foi  }
   WHERE {
      ?this a sosa:Observation ;
        sosa:hasFeatureOfInterest ?x .
     ?oc sosa:hasMember+ ?this ;
        sosa:hasFeatureOfInterest ?foi .
   } """ ;
   ];
 ].
```

Listing 9: SHACL shape to constrain the relation between sosa:Observation and sosa:ObservationCollection.

6 DATA-DEPENDENT SHAPES

The aforementioned shapes are all at the level of constraining the general usage of SOSA for materializing environmental data. Specifically to an individual dataset, it is also worthwhile defining data-dependent shapes that are further customized based on the nature of the specific data. Take the earthquake observation as an example (Section 3.2), we build two shapes in addition to SOSA-SHACL to facilitate the validation of USGS's Earthquake Dataset. First of all, according to SOSA-SHACL (Section 4), each earthquake observation (an instance of kwg-ont:EarthquakeObservation) should have exactly one observable property (an instance of kwg-ont:EarthquakeObservableProperty). However, SOSA-SHACL does not impose any restrictions on the use of observable property. Namely, if the input earthquake data mistakenly has an observable property of temperature, for example, SOSA-SHACL is incapable of detecting it. Consequently, we define a shape for USGS's Earthquake Dataset that only allows a closed set of observable properties (see Listing 10). Note that one can also use OWL to define a closed list as the range of sosa:observedProperty. However, it won't report any violation if a temperature property is included in the data.

```
sosa-shacl:USGSEarthquakeConstraint_observableProperty
    a sh:NodeShape ;
    sh:targetClass kwg-ont:EarthquakeObservation ;
    sh:property [
        sh:path sosa:observedProperty ;
        sh:class sosa:EarthquakeObservableProperty ;
        sh:in  (kwgr:observableproperty.depth
         kwgr:observableproperty.mag
         kwgr:observableproperty.magType
         ... ) .
        ]
```

Listing 10: SHACL shape to constrain the observable property of USGS's Earthquake Dataset. (... indicates the omission of observable properies.)

Additionally, the location of an earthquake observation is often reported using a position on the surface of the earth (i.e., epicenter), which is represented as a geographic point with latitude and longitude. So if the location of an earthquake was recorded as a polygon or a polyline, a warning of potential violation should be reported to the user. Listing 11 demonstrates such a shape using SHACL. This shape is similar to the scoped range of using OWL. But again its purpose is to validate the data rather than for reasoning.

```
sosa-shacl:USGSEarthquakeConstraint_location
    a sh:NodeShape ;
    sh:targetClass kwg-ont:EarthquakeObservation ;
    sh:property [
        sh:path geo:hasGeometry ;
        sh:class sf:Point ;
        ] .
```

Listing 11: SHACL shape to constrain the location geometry of USGS's Earthquake Dataset.

In summary, this section simply uses the USGS's Earthquake Dataset to extend SOSA-SHACL to more customized shapes. Additional shapes can be defined for other environmental data such as EPA's air quality observation and MTBS's wildfire observation by considering their inherent characteristics.

7 CONCLUSION AND FUTURE WORK

Environmental observations are critical to understanding earth system processes, and are becoming a vital part of the Semantic Web – allowing major challenges such as climate change, disease outbreaks, and disaster responses to be addressed more effectively in an interdisciplinary and holistic way. While standards such as the Sensor, Observation, Sample, and Actuator (SOSA) ontology are available to formally represent environmental observation as a RDF graph, adhere to such standards is hard to determine given the Open World assumption of RDF/OWL. This work provides a solution to evaluate whether the data in a Knowledge Graph are in compliance with ontologies like SOSA, by using the Shapes Constraint Language (SHACL). Specifically, we proposed a number of shapes to conform, and confirm, the representation of different environmental datasets according to specifications of the SOSA ontology as well as the nature of environmental observations. If a violation was found while constraining the shapes graph on the data graph, an error would be reported to the user. Hence, these SHACL shapes assist one to assess and refine the quality of RDF graphs by using the SOSA ontology, providing a consistent representation of observational data, with adequate classes and properties to inform scientific interpretation and re-use.

Beyond developing a number of SOSA-SHACL patterns that can be readily used by environmental scientists to conform their data for ingestion into our KnowWhereGraph, other key contributions of this work include: (1) a comparison between the fundamentals underlying OWA-based OWL ontologies and SHACL, which motivates the use of SHACL to validate more complete and consistent reporting of environmental observations; and (2) the fact that SOSA-SHACL is not simply built based on a one-to-one translation of the SOSA ontology to its SHACL version; we instead use three real-world use cases – air quality, earthquakes, and wildfires, to take into account the complexity and diversity of environmental data. In the future, we plan to apply SOSA-SHACL to validate a wider range of environmental applications, to design more customized SHACL shapes for environmental data on top of SOSA, and to explore the feasibility of using SOSA-SHACL to summarize other large-scale, heterogeneous, and multi-model environmental knowledge graphs.

ACKNOWLEDGMENTS

The authors acknowledge support by the National Science Foundation under Grant 2033521 A1: KnowWhereGraph: Enriching and Linking Cross-Domain Knowledge Graphs using Spatially-Explicit AI Technologies. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Daniel Alvarez-Coello, Daniel Wilms, Adnan Bekan, and Jorge Marx Gómez. 2021. Generic semantization of vehicle data streams. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*. IEEE, 112–117.

[2] Víctor Caballero, Sergi Valbuena, David Vernet, and Agustín Zaballos. 2019. Ontology-Defined middleware for internet of things architectures. *Sensors* 19, 5 (2019), 1163.

[3] Michael Compton, Payam Barnaghi, Luis Bermudez, Raul Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. 2012. The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics* 17 (2012), 25–32.

[4] Christophe Debruyne and Kris McGlinn. 2021. Reusable SHACL Constraint Components for Validating Geospatial Linked Data. In *Proceedings of the 4th International Workshop of Geospatial Linked Data*. CEUR-WS.

[5] Tarek Elsaleh, María Bermudez-Edo, Shirin Enshaeifar, Sahr Thomas Acton, Roonak Rezvani, and P Barnaghi. 2019. IoT-stream: a lightweight ontology for internet of things data streams. In *2019 Global IoT Summit (GIoTS)*. IEEE, 1–6.

[6] Tarek Elsaleh, Shirin Enshaeifar, Roonak Rezvani, Sahr Thomas Acton, Valentinas Janeiko, and Maria Bermudez-Edo. 2020. IoT-Stream: A lightweight ontology for internet of things data streams and its use with data analytics and event detection services. *Sensors* 20, 4 (2020), 953.

[7] Paola Espinoza-Arias, María Poveda-Villalón, Raúl García-Castro, and Oscar Corcho. 2019. Ontological representation of smart city data: From devices to cities. *Applied Sciences* 9, 1 (2019), 32.

[8] Jose Emilio Labra Gayo, Eric Prud'Hommeaux, Iovka Boneva, and Dimitris Kontokostas. 2017. Validating RDF data. *Synthesis Lectures on Semantic Web: Theory and Technology* 7, 1 (2017), 1–328.

[9] Krzysztof Janowicz, Armin Haller, Simon JD Cox, Danh Le Phuoc, and Maxime Lefrançois. 2019. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics* 56 (2019), 1–10.

[10] Holger Knublauch. 2015. *Shapes Constraint Language (SHACL). W3C editor's draft, World Wide Web Consortium.* http://w3c.github.io/data-shapes/shacl/

[11] Holger Neuhaus and Michael Compton. 2009. The semantic sensor network ontology. In *AGILE workshop on challenges in geospatial data harmonisation, Hannover, Germany*. 1–33.

[12] Hoan Nguyen Mau Quoc, Martin Serrano, Han Mau Nguyen, John G Breslin, and Danh Le-Phuoc. 2019. EAGLE—A scalable query processing engine for linked sensor data. *Sensors* 19, 20 (2019), 4362.

[13] Ba-Huy Tran, Nathalie Aussenac-Gilles, Catherine Comparot, and Cassia Trojahn. 2020. Semantic integration of raster data for earth observation: An RDF dataset of territorial unit versions with their land cover. *ISPRS International Journal of Geo-Information* 9, 9 (2020), 503.

[14] Marc Vila, Maria-Ribera Sancho, Ernest Teniente, and Xavier Vilajosana. 2021. Semantics for Connectivity Management in IoT Sensing. In *International Conference on Conceptual Modeling*. Springer, 297–311.

[15] Chao Wang, Xinyan Zhuo, Pengfei Li, Nengcheng Chen, Wei Wang, and Zeqiang Chen. 2020. An Ontology-Based Framework for Integrating Remote Sensing Imagery, Image Products, and In Situ Observations. *Journal of Sensors* 2020 (2020).

[16] Marcos Zárate, Germán Braun, Mirtha Lewis, and Pablo Fillottrani. [n.d.]. Observational/Hydrographic data of the South Atlantic Ocean published as LOD. *Semantic Web* Preprint ([n. d.]), 1–12.

[17] Rui Zhu, Shirly Ambrose, Lu Zhou, Cogan Shimizu, Ling Cai, Gengchen Mai, Krzysztof Janowicz, Pascal Hitzler, and Mark Schildhauer. 2021. Environmental Observations in Knowledge Graphs. In *2nd Workshop on Data and research objects management for Linked Open Science*, 1–11.