Transferring Remote Ontologies to the Edge of Internet of Things Systems

Xiang Su[1], Pingjiang Li[1], Huber Flores[2], Jukka Riekki[1], Xiaoli Liu[3], Yuhong Li[4], and Christian Prehofer[5]

[1] Center for Ubiquitous Computing, University of Oulu, Finland,
[2] Department of Computer Science, University of Helsinki, Finland,
[3] Biomimetics and Intelligent Systems Group, University of Oulu, Finland,
[4] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China,
[5] fortiss, An-Institut Technische Universität München, Munich, Germany

Abstract. Edge computing paradigm allows computation to be moved from the central high powered Cloud or data center to the edge of the network. This paradigm often enables more efficient data processing near its source and sends only the data and knowledge that have value over the network. Our study focuses on performing semantic reasoning at the edge computing devices, which requires transferring ontologies to the edge devices. This paper presents different representations for transferring Web Ontology language (OWL) version 2 ontologies to the edge. We evaluate different representations in an experimental IoT system with edge nodes and compare lengths of different syntaxes and their computation effort of building models in Cloud and edge computing devices in terms of processing time.

Keywords: Internet of Things, Edge computing, OWL 2, Ontology

1 Introduction

Internet of Things (IoT) systems not only gather a large quantity of data generated by things, but also focus on how data can be processed, visualized, and possibly acted upon. A new computing paradigm, edge computing, calls for performing data analytics and knowledge generation to occur at the periphery of the network. In the edge computing paradigm, sensors and connected devices transmit data to nearby edge computing devices, such as gateway devices that process or analyze the data, instead of delivering it to the Cloud or a remote data center. Major benefits of edge computing are improving time to action and reducing response time; conserving network resources and addressing battery life constraint; supporting security and privacy sensitive services and applications; and enabling scalable distributed data processing. In general, to enable the vision of edge computing in a typical IoT system, the information at edge computing devices is pushed from Cloud services and pulled from IoT devices [1][2].

Compared with traditional base stations, just simply forwarding data traffic but do not actively processing the data, edge computing devices include more

computing and storage capabilities. Hence, more advanced functions can be deployed on edge computing devices. For example, when the increment of the raw data is produced from the IoT devices but not all raw data is useful, edge computing devices process a considerable amount of the raw data, which saves bandwidth and reduces the latency. Moreover, the edge computing devices are close to end users, the response time will be predictable if the data is processed at edge computing devices [1].

We focus on utilising Semantic Web technologies in the edge computing devices of IoT systems. Semantic Web technologies give information well-defined meaning, better enabling computers and people to work in cooperation. One essential property is its universality powered by the "anything can link to anything" property of hypertext links. Semantic reasoning derives facts that are not explicitly expressed. To enable semantic reasoning functions, computers have to access structured collections of the information and sets of inference rules that they can use to conduct automated reasoning. Hence, Semantic Web community offers a set of languages that express data, knowledge, and rules for reasoning about the data. Knowledge is typically modeled with ontologies, which is a taxonomy defining the classes of objects and relations among them.

The components in IoT systems could reach a shared understanding by exchanging knowledge. This is especially important for edge computing devices in IoT systems, because they often share a part of an ontology from a comprehensive ontology on the Cloud or a server machine. In our earlier research, we studied how semantics could be embedded in the data generated by tiny IoT devices, and how the benefits of semantics could be utilized without sacrificing the efficiency of IoT in terms of energy consumption and reasoning latency [3][4]. In this paper, we focus on transferring ontologies to edge computing devices and developing knowledge models to perform semantic reasoning. We compare and evaluate different syntaxes for Web Ontology Language (OWL) version 2 [5] in an experimental IoT system with edge computing devices. We present a comparison of lengths of different syntaxes and their required computation effort of building models in Cloud and edge computing devices in terms of processing time. In our experiments, we demonstrate that by changing from a standard syntax to a lightweight syntax, it is possible to reduced 47.2% of data when we transfer the Semantic Sensor Network (SSN) ontology [6] ontology.

The remainder of this article is organized as follows. We present different syntaxes with examples in Section 2 and details of our experiment results in Section 3. We conclude the article with proposing future research directions in Section 4.

2 OWL 2 and Syntaxes for Transferring Ontologies

2.1 OWL 2

OWL 2 is a W3C standardized ontology language for the Semantic Web with formally defined meaning. Before the development and standardization of OWL

and OWL 2, there were plenty of ontology languages, such as KL-ONE [7], F-logic [8], SHOE [9], DAML-ONT [10], OIL [11], and DAML+OIL [12]. These efforts finally lead to developing OWL, a comprehensive ontology language for the Semantic Web. OWL is a standard Semantic Web language based on Description Logic [13]. The main building blocks of OWL are concepts representing sets of objects, roles representing relationships between objects, and individuals representing specific objects. With OWL, complex concepts can be described through constructors that define the conditions on concept membership. OWL 2 is a revised extension of OWL, which is now commonly called OWL 1. OWL 2 extends OWL 1 with qualified cardinality restrictions and property chains. Moreover, OWL 2 provides support for defining properties to be reflexive, irreflexive, transitive, asymmetric, and to define disjoint pairs of properties. Three profiles, namely OWL 2 EL, OWL 2 QL, and OWL 2 RL, have been developed for balancing expressive power and reasoning efficiency, targeting different application scenarios. OWL 2 EL is suitable for applications utilizing ontologies to define very large numbers of classes and properties. OWL 2 QL can be tightly integrated with Relational Database Management Systems and can hence benefit from relational database technology. OWL 2 RL is suitable for applications that require scalable reasoning without sacrificing too much expressive power [14].

OWL 2 provides a rich collection of constructs for forming descriptions, and is compatible with existing Web standards. An OWL 2 ontology consists of a set of axioms which place constraints on sets of individuals and on the types of relationships between them. These axioms provide semantics by allowing systems to explicitly infer additional knowledge based on the data provided. OWL 2 provides a bidirectional mapping from the OWL Functional Syntax to Resource Descriptoin Framework(RDF) Graphs. This means that OWL can then be serialized into any RDF representations such as RDF/XML and Notation 3 (N3). Most current OWL tools utilize RDF/XML as the default syntax for serializing ontologies.

2.2 OWL 2 Syntaxes

An ontology can be developed with IRIs and a set of axioms. There are different languages for storing, sharing, and editing IRIs and axioms in OWL 2 ontologies. Among them, RDF/XML is the officially recommended exchange syntax by W3C, and others have been designed for particular purposes and applications. In this section, we compare different syntaxes for OWL in brief and we illustrates different syntaxes with an example in Appendix, which is a part of SSN ontology [6]. This small part of SSN ontology consists of simple concepts of *Input* class and *Output* classes and some relations of them, including *label*, *source*, *disjoint Class*, etc.

RDF/XML is the primary and widely supported syntax, as it is recommended by W3C. Because OWL supports a bidirectional mapping to RDF triples, it is convenient to be combined in RDF/XML. Other RDF based representations share this benefit, for example N3 and Turtle. However, RDF/XML is a verbose representation for OWL and can hence be difficult to read by human users.

For example, RDF/XML requires nesting and reification for complex class expressions, which results RDF verbose and difficult to read. Moreover, parsing RDF/XML requires two steps which means that much more memory is required for storing the ontology in memory. Turtle is more readable than RDF/XML and also widely supported. W3C selects Turtle as one of the syntaxes for OWL 2 and widely used tools and APIs such as Jena and the OWL API support Turtle.

OWL functional syntax is designed to be easier for OWL 2 specification purposes and to provide a foundation for the implementation of OWL 2 tools. Functional syntax is a simple text based syntax that is used as a bridge between the structural specification and concrete representations.

The design of RDF/XML makes it difficult to utilize off-the-shelf XML tools for tasks other than parsing and rendering it. Standard XML tools like XPath or XSLT do not work well with RDF/XML representations of ontologies [15]. Moreover, serializing OWL and OWL 2 requires resources, as OWL needs to be first mapped to RDF, and then RDF needs to be serialized to XML. To overcome these difficulties, OWL/XML is invented as a concrete syntax for a more regular and simpler XML syntax. The syntax is essentially derived directly from the Functional Syntax. However, OWL/XML suffers from verbose syntax and this often slows down parsing.

Manchester syntax [16] is designed for editing and presentation purposes. It provides for OWL ontologies a compact text based representation that is easy to read and write. The primary motivation for the design of the Manchester OWL syntax is to produce a syntax that can be used for editing class expressions. This effort has been extended so that it is possible to represent complete ontologies, and Manchester Syntax is now standardized by W3C. Different from above mentioned syntaxes, the Manchester syntax gathers together information about names in a frame-like manner. Manchester Syntax is cumbersome for representing some axioms in OWL, such as general class axioms.

Another recent effort is to serialize RDF in JSON format. The W3C RDF working group compares alternative JSON formats for RDF with examples. Based on this comparison, JSON-LD [17] is considered as the most promising format and becomes a W3C recommendation in early 2014. JSON-LD is designed to be completely compatible with JSON and it has a slightly better expressive power than the RDF model. This means that in practice it can be considered to be a JSON serialization for RDF. JSON-LD requires minimal effort from developers to transform normal JSON to JSON-LD. Only two keywords (@context and @id) need to be known for utilizing the basic features.

Entity Notation (EN) and Entity Notation Schema (EN Schema) [18] are lightweight knowledge representations for resource-constrained devices. Resource usage of CPU, memory, bandwidth, and energy for encoding and decoding these representations are considered at design time. EN is designed as a syntax of RDF and EN Schema extends the design for representing OWL 2 ontologies. Both EN and EN Schema have complete packet and short packet. The complete packet has a structure resembles the triple structure of RDF and OWL. The compact format can shorten the representation with templates and prefixes.

Attempto Controlled English [19] is a machine-oriented Controlled Natural Language (CNL), that is, a precisely defined subset of the English language, designed for writing unambiguous and precise specification texts for knowledge representation. Attempto Controlled English supports a bidirectional translation into and from OWL 2. Other representations with similar design goals are Sydney OWL syntax [20] and Ordnance Survey's Rabbit [21]. However, we are not aware of its compatibility with OWL 2. Schwitter et al. compared these three representations with examples [22]. However, we do not consider these languages suitable for transferring ontologies in IoT systems.

3 Experiments and Analysis

We present two experiments to evaluate different syntaxes of OWL 2 ontologies in a previous Section. We focus on 1) length comparison of different OWL 2 syntaxes and 2) computation effort of building a same Jena model in the Cloud and an edge computing device with different OWL 2 syntaxes in terms of processing time.

3.1 Length comparision for OWL2 syntaxes

Our first experiment is to evaluate different OWL 2 syntaxes with a set of seven ontologies, which are well known and widely utilized in pervasive computing, IoT, and other domains. These ontologies include COBRA-ONT [23], IoT-Lite ontology [24], Socially Interconnected Online Communities(SIOC) ontology[25], SSN ontology, and the Organization Ontology[26]. COBRA-ONT includes a collection of ontologies for describing vocabularies in an intelligent meeting room use case. We are testing with COBRA-ONT ontologies version 0.6 which contains ebiquity-geo, ebiquity-meetings and ebiquity-actions ontologies. The IoT-Lite ontology is a lightweight ontology to represent IoT resources, entities, and services. It is also a meta ontology that can be extended in different domains. The SSN ontology describes the sensors, observations and related concepts in pervasive environments. SIOC ontology is designed for describing online communities such as forums, blogs, mailing lists, and wikis. The Organization ontology is designed to enable publication of information on organizations and organizational structure including governmental organizations etc. It is designed as a generic, reusable core ontology that can be extended or specialized for use in particular situations. It is also a meta ontology that can be extended in different domains.

Figure 1 presents the comparison of the lengths of different syntaxes, including Functional syntax, EN Schema, Turtle, Manchester syntax, RDF/XML, JSON-LD, N-Triple, and OWL/XML (from left to right). Our experiment shows that sizes of different syntaxes vary for ontologies. Taking SSN ontology as an example, the length of EN Schema is about 36.6% of N-Triple and about 52.8% of RDF/XML. For IoT-Lite Ontology, the JSON-LD syntax is 5.2 times than the Manchester syntax. In ebiquity-meeting Ontology, the ratio is 2.3 which is

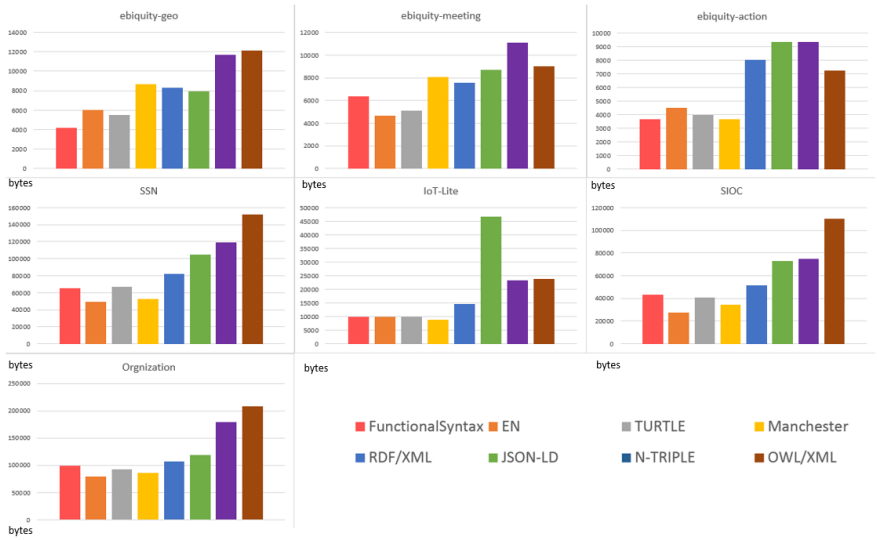

Fig. 1. Comparison of the lengths of different OWL 2 syntaxes.

the smallest among all the sets. The average ratio between the biggest format and the smallest one is 3.3 times.

In general, EN Schema, Turtle, and Manchester syntax are more compact than the other syntaxes. Especially, EN Schema is the most compact syntax for four ontologies, including COBRA-ONT ebiquity-meeting, SIOC, SSN, and the Organization Ontology. JSON-LD, N-Triple and OWL/XML appear to be the most verbose format. The RDF/XML is among the medium size. OWL/XML makes the biggest size ontologies in four out of eight ontologies. According to the results of this comparison, EN Schema, Turtle and Manchester syntax are considered to be optimized syntaxes when IoT system developers require to minimize the communication load for transferring ontologies. However, some syntaxes may have benefits of transferring certain structures in ontologies, for example, Turtle minimizes the lengths of representations of complex classes in most ontologies. Therefore, when certain structures are dominant in a ontology, it is often easier to select an optimized syntax.

3.2 Comparison of building ontology models with OWL 2 syntaxes

The second experiment focuses on building Jena models with different OWL syntaxes. The experiment is executed under a simulated IoT environment. Figure 2 presents a general architecture of the IoT system with edge nodes. The system consists of a IoT node layer, edge computing devices, and a Cloud server. The IoT node layer consists of end-point device such as sensors and single-chip devices. These devices are regraded as IoT nodes which can detect the environmental situation and generate real-time data. But the capacities of these device

are limited. Therefore, the edge devices and the Cloud server are required for complex data processing tasks. The edge computing devices are often physically closed to the IoT nodes. with the capabilities of semantic modelling and processing, edge computing devices, such as smart mobile phones, can receive small ontologies and perform fast reasoning tasks. In regard of heavy reasoning task, edge computing devices send the data to Cloud server to process. The Cloud server is assumed to have unlimited computing power, but physically far from the IoT nodes. The response time is often slow because of the network latency. The IoT system enables transferring part of an ontolgogy to support the semantic reasoning tasks. The distribution of reasoning tasks utilizes the resources of an IoT system to its best and offers predictable response time for users.

In this experiment, we utilize LG Nexus 5X phones with Android OS version 6.0.1 for edge computing devices. The Nexus 5X has six CPU cores, which consists of four Quad-core 1.44 GHz Cortex-A53 processors and two dual-core 1.82 GHz Cortex-A57 processors, running on Qualcomm MSM8992 Snapdragon 808 chip-set. It has 2GB RAM and 32GB storage. We utilize Amazon M4 Deca Extra Large Cloud as Cloud server, which has 160 GB memory with 124.5 EC2 Compute Units.

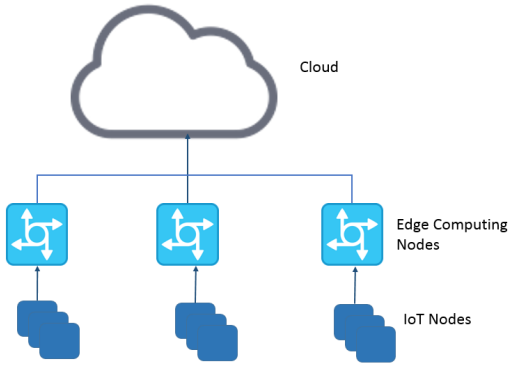

Fig. 2. A general architecture for IoT systems with edge computing nodes.

In our experiment, we evaluate the performance of different OWL 2 syntaxes by comparing the required time of building ontology models with Jena framework in an edge computing device. We again use the previously mentioned seven ontologies with different syntaxes and we include sample ontology in OWL 2 primer in this test. Figure 3 presents the Jena *OntModel* building time (in millisecond) among different ontologies with five selectd syntaxes, including RDF/XML, JSON-LD, Turtle, EN Schema, and N-triple. EN Schema is not directly supported by Jena, so we add one step to transfer EN Schema to Turtle and the transferring time is around 20 milliseconds. In the Jena framework, a RDF graph is built as a model and is implemented with the "Model" interface.

The Jena framework includes object classes to represent graphs, resources, properties, and literals. The interfaces representing resources, properties, and literals are called Resource, Property, and Literal respectively. For ontology, Jena use *OntModel*, which is an extension to the *InfModel* interface. Jena wraps an underlying model with this ontology interface, that presents a convenient syntax for accessing the language elements. We utilize OWL API to transfer syntaxes. To keep the minimum size of the ontology, all the transformed ontologies only contain the OWL axioms and annotation, while the comments are removed. We keep space characters as it is necessary for structure and separating elements.

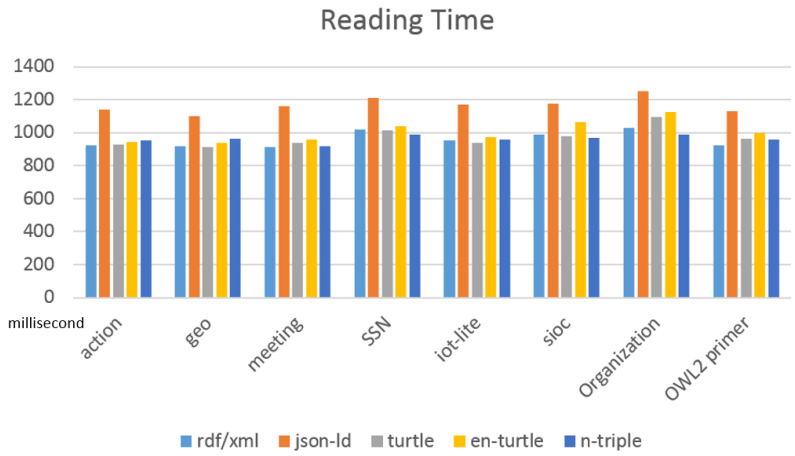

Fig. 3. Comparision of building ontology models with Jena framework with OWL2 syntaxes.

We find that for all ontologies, JSON-LD requires longer time than the other four syntaxes and the other four syntaxes consume comparable amount of time. RDF/XML, Turtle, and N-Triple appear to be the shortest format. In ebiquity-meeting Ontology, the JSON-LD format reading time is 27% longer than the shortest RDF/XML one. JSON-LD reading time is 23% longer than the smallest one for all the test cases in average.

Figure 4 presents the different *OntModel* building time on the Cloud with the increase of ontology size. The X-axis of Figure 4 represents the size of the above mentioned ontologies. The Y-axis represents the modelling time on the Cloud (in millisecond). We notice a linear increment with the incremental size of ontologies. However, with larger amount of ontology data, Figure 4 shows the processing time is increasing but considerably slower than the increasing of the data size. For example, the largest ontology is 23 times than the smallest one but the required modelling time is only 20 % with Turtle syntax. From this experiment, we consider loading a large ontology, which often enables rich functions, but does not introduce too much modelling time for a Cloud server.

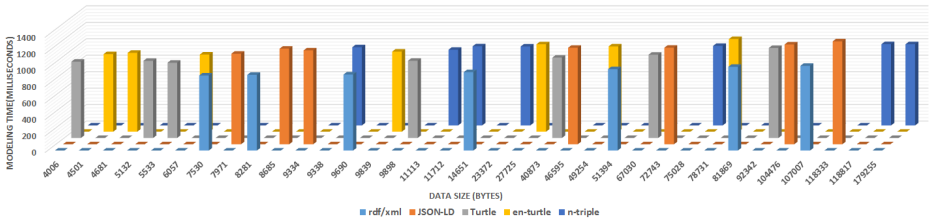

Fig. 4. Jena model building time by length.

4 Conclusions and Future Work

Transferring ontologies to edge computing devices enables intelligent functions for IoT systems, such as developing semantic models and performing reasoning. This paper presents different OWL 2 syntaxes for transferring ontologies to edge computing devices for developing knowledge models. Our main contribution is a comparison of the length of different syntaxes and their computation effort of building models in edge computing devices and Cloud in terms of processing time. Our experiments show that implementing semantic functions and services are practical solutions in edge computing devices.

However, there are challenges for introducing semantic functions and services in edge computing devices of IoT systems. First, scalability is a challenge as the tasks need to be deployed and performed on different devices based on real capabilities of these devices and communication networks. We need to minimize the overall cost in order to find an efficient deployment of tasks on to edge computing devices with different capabilities. Second, dynamic extensibility challenge of IoT systems should be addressed. When new IoT devices are sending data to edge nodes, how to balance the semantic functions with different edge computing devices to utilize their resources to their best? A flexible and extensible design is important to address this challenge. Third, data reliability and veracity challenges rise from the sensing and communication of data in IoT systems. Some solutions to handle unreliable conditions should be considered.

As future research, we will address these challenge of dynamic linking data from data sources to edge computing devices to achieve reasoning efficiency of whole IoT systems. Moreover, we will tackle the challenge of data veracity and variety to support intelligent decision making in IoT systems with edge computing devices.

Acknowledgments. The first author would like to thank Jorma Ollila Grant of Nokia foundation for funding this research.

Appendix

Example of SSN ontology with RDF/XML syntax:

```
<owl:Class rdf:about="http://purl.oclc.org/NET/ssnx/ssn#Input">
<rdfs:label>Input</rdfs:label>
<owl:disjointWith rdf:resource="http://purl.oclc.org/NET/
ssnx/ssn#Output"/>
<dc:source>http://marinemetadata.org/community/teams/ontdevices
</dc:source>
<rdfs:seeAlso>http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#Process
</rdfs:seeAlso>
<rdfs:isDefinedBy>http://purl.oclc.org/NET/ssnx/ssn</rdfs:isDefinedBy>
<rdfs:comment>Any information that is provided to a process for its use.
</rdfs:comment>
</owl:Class>
<owl:Class rdf:about="http://purl.oclc.org/NET/ssnx/ssn#Output">
<rdfs:label>Output</rdfs:label>
<rdfs:comment>Any information that is reported from a process.
</rdfs:comment>
<dc:source>http://marinemetadata.org/community/teams/ontdevices
</dc:source>
<rdfs:isDefinedBy>http://purl.oclc.org/NET/ssnx/ssn</rdfs:isDefinedBy>
<rdfs:seeAlso>http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#Process
</rdfs:seeAlso>
</owl:Class>
```

Example of SSN ontology with Turtle syntax:

```
<http://purl.oclc.org/NET/ssnx/ssn#Input>
  a owl:Class;
  rdfs:label "Input";
  owl:disjointWith <http://purl.oclc.org/NET/ssnx/ssn#Output>;
  dc11:source "http://marinemetadata.org/community/teams/ontdevices";
  rdfs:seeAlso "http://www.w3.org/2005/Incubator/ssn/wiki/
  SSN_Model#Process";
  rdfs:isDefinedBy "http://purl.oclc.org/NET/ssnx/ssn";
  rdfs:comment "Any information that is provided to a process for its use.".
<http://purl.oclc.org/NET/ssnx/ssn#Output>
  a owl:Class;
  rdfs:label "Output";
  rdfs:comment "Any information that is reported from a process.";
  dc11:source "http://marinemetadata.org/community/teams/ontdevices";
  rdfs:isDefinedBy "http://purl.oclc.org/NET/ssnx/ssn";
  rdfs:seeAlso "http://www.w3.org/2005/Incubator/ssn/wiki/
  SSN_Model#Process".
```

Example of SSN ontology with functional syntax:

```
Ontology:
AnnotationProperty: dc11:source
AnnotationProperty: rdfs:comment
AnnotationProperty: rdfs:isDefinedBy
AnnotationProperty: rdfs:label
```

```
AnnotationProperty: rdfs:seeAlso
Datatype: rdf:PlainLiteral

Class: <http://purl.oclc.org/NET/ssnx/ssn#Input>
Annotations:
  rdfs:isDefinedBy "http://purl.oclc.org/NET/ssnx/ssn",
rdfs:label "Input",
rdfs:seeAlso "http://www.w3.org/2005/Incubator/ssn/wiki/
SSN_Model#Process",
dc11:source "http://marinemetadata.org/community/teams/ontdevices",
rdfs:comment "Any information that is provided to a process
for its use."
DisjointWith: <http://purl.oclc.org/NET/ssnx/ssn#Output>
Class: <http://purl.oclc.org/NET/ssnx/ssn#Output>
Annotations:
  rdfs:isDefinedBy "http://purl.oclc.org/NET/ssnx/ssn",
rdfs:seeAlso "http://www.w3.org/2005/Incubator/ssn/
wiki/SSN_Model#Process",
dc11:source "http://marinemetadata.org/community/teams/ontdevices",
rdfs:comment "Any information that is reported from a process",
rdfs:label "Output".
DisjointWith: <http://purl.oclc.org/NET/ssnx/ssn#Input>
```

Example of SSN ontology with OWL/XML syntax:

```
<Ontology
<Declaration><Class IRI="http://purl.oclc.org/NET/ssnx/ssn#Output"/>
</Declaration>
<Declaration><Class IRI="http://purl.oclc.org/NET/ssnx/ssn#Input"/>
</Declaration>
<Declaration><AnnotationProperty abbreviatedIRI="dc11:source"/>
</Declaration>
<DisjointClasses>
<Class IRI="http://purl.oclc.org/NET/ssnx/ssn#Input"/>
<Class IRI="http://purl.oclc.org/NET/ssnx/ssn#Output"/>
</DisjointClasses>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="dc11:source"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Input</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
http://marinemetadata.org/community/teams/ontdevices</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:comment"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Input</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
Any information that is provided to a process for its use.</Literal>
</AnnotationAssertion>
```

```
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:isDefinedBy"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Input</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
http://purl.oclc.org/NET/ssnx/ssn</Literal></AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Input</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">Input</Literal></AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:seeAlso"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Input</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#Process</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="dc11:source"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Output</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
http://marinemetadata.org/community/teams/ontdevices</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:comment"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Output</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
Any information that is reported from a process.</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:isDefinedBy"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Output</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
http://purl.oclc.org/NET/ssnx/ssn</Literal></AnnotationAssertion>
<AnnotationAssertion>
<AnnotationProperty abbreviatedIRI="rdfs:label"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Output</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">Output</Literal></AnnotationAssertion>
<AnnotationAssertion><AnnotationProperty abbreviatedIRI=rdfs:seeAlso"/>
<IRI>http://purl.oclc.org/NET/ssnx/ssn#Output</IRI>
<Literal datatypeIRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#
PlainLiteral">
http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#Process
</Literal></AnnotationAssertion>
</Ontology>
```

Example of SSN ontology with Manchester syntax:

```
Ontology:
AnnotationProperty: dc11:source
AnnotationProperty: rdfs:comment
AnnotationProperty: rdfs:isDefinedBy
AnnotationProperty: rdfs:label
AnnotationProperty: rdfs:seeAlso
Datatype: rdf:PlainLiteral
Class: <http://purl.oclc.org/NET/ssnx/ssn#Input>
Annotations:
  rdfs:isDefinedBy "http://purl.oclc.org/NET/ssnx/ssn",
  rdfs:label "Input",
  rdfs:seeAlso "http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model
  #Process",
  dc11:source "http://marinemetadata.org/community/teams/ontdevices",
  rdfs:comment "Any information that is provided to a process for its use."
  DisjointWith: <http://purl.oclc.org/NET/ssnx/ssn#Output>
  Class: <http://purl.oclc.org/NET/ssnx/ssn#Output>
Annotations:
  rdfs:isDefinedBy "http://purl.oclc.org/NET/ssnx/ssn",
  rdfs:seeAlso "http://www.w3.org/2005/Incubator/ssn/wiki/
  SSN_Model#Process",
  dc11:source "http://marinemetadata.org/community/teams/ontdevices",
  rdfs:comment "Any information that is reported from a process.",
  rdfs:label "Output"
  DisjointWith:<http://purl.oclc.org/NET/ssnx/ssn#Input>
```

Example of SSN ontology with JSON-LD:

```
[{"@id":"http://purl.oclc.org/NET/ssnx/ssn#Input",
  "@type":
  ["http://www.w3.org/2002/07/owl#Class"],
  "http://www.w3.org/2000/01/rdf-schema#label":
  [{"@value":"Input"}],
  "http://www.w3.org/2002/07/owl#disjointWith":
[{"@id":"http://purl.oclc.org/NET/ssnx/ssn#Output"}],
"http://purl.org/dc/elements/1.1/source":
[{"@value":"http://marinemetadata.org/community/teams/ontdevices"}],
"http://www.w3.org/2000/01/rdf-schema#seeAlso":
[{"@value":"http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#
Process"}],
"http://www.w3.org/2000/01/rdf-schema#isDefinedBy":
[{"@value":"http://purl.oclc.org/NET/ssnx/ssn"}],
"http://www.w3.org/2000/01/rdf-schema#comment":
[{"@value":"Any information that is provided to a process for
its use."}]},
{"@id":"http://purl.oclc.org/NET/ssnx/ssn#Output",
"@type":["http://www.w3.org/2002/07/owl#Class"],
"http://www.w3.org/2000/01/rdf-schema#label":
```

```
[{"@value":"Output"}],
"http://www.w3.org/2000/01/rdf-schema#comment":
[{"@value":"Any information that is reported from a process."}],
"http://purl.org/dc/elements/1.1/source":
[{"@value":"http://marinemetadata.org/community/teams/ontdevices"}],
"http://www.w3.org/2000/01/rdf-schema#isDefinedBy":
[{"@value":"http://purl.oclc.org/NET/ssnx/ssn"}],
"http://www.w3.org/2000/01/rdf-schema#seeAlso":
[{"@value":"http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#
Process"}]},
{"@id":"http://www.w3.org/2002/07/owl#Class"}]
```

Example of SSN ontology with EN Schema:

```
<owl:Class http://purl.oclc.org/NET/ssnx/ssn#Output
a owl:Class
rdfs:isDefinedBy http://purl.oclc.org/NET/ssnx/ssn
rdfs:seeAlso http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#Process
rdfs:label "Output"
rdfs:comment "Any information that is reported from a process."
dc11:source http://marinemetadata.org/community/teams/ontdevices>
<owl:Class http://purl.oclc.org/NET/ssnx/ssn#Input
a owl:Class
rdfs:isDefinedBy http://purl.oclc.org/NET/ssnx/ssn
rdfs:seeAlso http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Model#Process
rdfs:label "Input"
rdfs:comment "Any information that is provided to a process for its use."
owl:disjointWith http://purl.oclc.org/NET/ssnx/ssn#Output
dc11:source http://marinemetadata.org/community/teams/ontdevices>
```

References

1. Shi, W., Cao, J., Zhang, Q., Li Y., Xu L.: Edge Computing: Vision and Challenges. IEEE Internet of Things Journal 3(5), 637–646 (2016)
2. Su X., Li P., Li Y., Flores H., Riekki J., Prehofer P.: Towards Semantic Reasoning on the Edge of IoT Systems. in Proceedings of the 6th International Conference on the Internet of Things, pp. 171–172, ACM Press, Stuttgart, Germany (2016)
3. Su, X., Riekki, J., Nurminen, JK., Nieminen J., Koskimies, M.: Adding semantics to internet of things. Concurrency and Computation: Practice and Experience 27(8), 1844–1860 (2015)
4. Maarala, AI., Su, X., Jukka, R.: Semantic reasoning for advanced Internet of Things applications. IEEE Internet of Things Journal 2(4), 1–13 (2016)
5. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, PF., Rudolph, S.: OWL 2 Web Ontology Language Primer (Second Edition), `https://www.w3.org/TR/owl2-primer/`
6. Semantic sensor network ontology, `http://purl.oclc.org/NET/ssnx/ssn`
7. Woods, WA., Schmolze, JG.: The KL-ONE family. Computers & Mathematics with Applications 23(2-5), 133–177 (1992)

8. Kifer, M., Lausen, G.: F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In: Proceedings of the 1989 ACM SIGMOD international conference on Management of data, pp. 134–146. ACM press, Portland, Oregon (1989)
9. Heflin, J., Hendler, Luke J., SHOE S.: A Knowledge Representation Language for Internet Applications. Technical report CS-TR-4078, Department of Computer Science, University of Maryland (1999)
10. McGuinness, DL., Fikes, R., Stein, LA., Hendler, J.: DAML-ONT: an ontology language for the Semantic Web. In: Fensel, F., Hendler, J., Lieberman, H., Wahlster, W. (ed) Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press (2002)
11. Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, DL., Patel-Schneider, PF.: OIL: an ontology infrastructure for the Semantic Web. IEEE Intelligent Systems 16(2), 38–45 (2001)
12. McGuinness, DL., Fikes, R., Hendler, J., Stein, LA.: DAML+OIL: an ontology language for the Semantic Web. IEEE Intelligent Systems 17(5), 72–80 (2002)
13. Horrocks, I.: Reasoning with expressive description logics: theory and practice. In: Proceeding of the 19th International Conference on Automated Deduction, pp. 1–15. Springer, Copenhagen, Denmark (2002)
14. Motik, B., Grau, BC., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web ontology language profiles (second edition), http://www.w3.org/TR/owl2-profiles/
15. Yu, LY.: A deveoper's guide to the Semantic Web. Springer-Verlag, Berlin Heidelberg (2014)
16. Horridge, M., Patel-Schneider, PF.: OWL 2 Web Ontology Language Manchester Syntax (Second Edition), https://www.w3.org/TR/owl2-manchester-syntax/
17. SON for Linking Data, http://json-ld.org/
18. Su, X.: Lightweight Data and Knowledge Exchange for Pervasive Environments. Acta Universitatis Ouluensis series C581 (2016)
19. Fuchs, NE., Kaljurand, K., Kuhn, T.: Attempto controlled English for knowledge representation. In: Baroglio, C., Bonatti, PA., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (ed) Reasoning Web. pp. 104–124. Springer, Berlin Heidelberg (2008)
20. Cregan, A., Schwitter, R., Meyer, T.: Sydney OWL syntax - towards a controlled natural language syntax for OWL 1.1. In: Proceedings of the OWLED 2007 Workshop on OWL: Experience and Directions. CEUR-WS, Innsbruck, Austria (2007)
21. Hart, G., Johnson, M., Dolbear, C.: Rabbit: developing a control natural language for authoring ontologies. In: Proceedings of the 5th European Semantic Web Conference, pp. 348–360. Springer, Berlin Heidelberg (2008)
22. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A comparison of three controlled natural languages for OWL 1.1. In: Proceedings of the 4th OWL Experiences and Directions Workshop. Washington, USA (2008)
23. Chen, H., Finin, T., Joshi, A.: An ontology for context-aware pervasive computing environments. The Knowledge Engineering Review 18(3), 197–207 (2003)
24. Bermudez-Edo, M., Barnaghi, P., Elsaleh, T.: iot-lite Ontology, http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite
25. Berrueta, D., Brickley, D., Decker, S., Fernández, S., Görn, C., Harth, A., Heath, T., Idehen, K., Kjernsmo, K., Miles, A., Passant, A., Polleres, A., Polo, L., Sintek, M.: SIOC Core Ontology Specification, https://www.w3.org/Submission/sioc-spec/
26. Reynolds, D.: The Organization Ontology, https://www.w3.org/TR/vocab-org/