Alignment-based semantic translation of geospatial data

Maria Ganzha[*‡], Marcin Paprzycki[*§], Wiesław Pawłowski[†*], Paweł Szmeja[*]
Katarzyna Wasielewska[*]
[*] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
[†] Faculty of Mathematics, Physics, and Informatics, University of Gdańsk, Gdańsk, Poland
[‡] Warsaw University of Technology, Warsaw, Poland
[§] Warsaw Management Academy, Warsaw, Poland

***Abstract*—Recent years are characterized by a renewed interest in semantic technologies. Growing number of data providers and users facilitate production and consumption of semantically-annotated resources. Unfortunately, this process does not go hand-in-hand with "stabilization" (not to talk about standardization) of a set of broadly accepted ontologies. As a matter of fact, the number of ontologies being proposed/developed, often "competing" with each other within the same domain, systematically increases. This, in consequence, stimulates work devoted to, various forms of, ontology "reconciliation", such as: establishing alignments and/or ontology merging.**

In this context, in the Internet of Things (IoT), use of semantic technologies materializes, among others, in the handling of messages exchanged between artifacts that constitute IoT ecosystems. Here, some existing platforms (e.g. Open-IoT, UniversAAL or VITAL) use serializations of RDF to transmit messages, others (e.g. FIWARE or oneM2M) support ontologies, even if they don't use RDF. Thus, one of important unresolved issues remains: how to facilitate understanding of messages exchanged between artifacts founded on different ontologies (i.e. establish semantic interoperability). In this paper, we focus on semantic translations, based on ontology alignments. We discuss the problem of identifying and representing alignments (using geospatial data as the use case example), so that they can be further used in the translation process.

I. Introduction

With the increasing popularity of the idea of the Internet of Things (IoT), and related concepts of Web of Things (WoT), Big Data, and Linked Data, problems related to exchanging, storing and analyzing data originating from heterogeneous (in terms of both syntax and semantics) data sources require urgent actions. Even though, there exist multitude of standards for data representation, there is not a single "leading approach" that would dominate the others, and there does not seem to be any materializing in the near future. As a matter of fact, it is possible that D. Fensel was right, when he claimed[1] that he does not foresee a general "ontology of everything" to materialize. Rather, he expects multitude of small/local/domain specific ontologies to be developed. Obviously, on the other side of this discussion, there is a project CYC[2], which for more than 30 years attempts at developing a single/complete ontology of the world. Let us remain agnostic as to who is right, and be pragmatic. IoT ecosystems of today that have to bring together multiple platforms, applications, middlewares, networks, sensors, etc., cannot wait for standardized ontologies (on any level) to materialize and be agreed on. Therefore, the key aspect of success for IoT deployments, consisting of heterogeneous artifacts, is research that attempts to address the problem of achieving interoperability between existing data representations.

It is worth noting that interoperability, in the Internet of Things, is considered in a number of European projects within the Horizon2020 programme[3]. One of them, INTER-IoT [3] provides context for the work presented in this paper. Its goal is to provide tools and methods to achieve interoperability at different layers of IoT technology stack (devices, network, middleware, applications and services, data and semantics) and across them. The work done within the data and semantics layer includes, among others, designing and implementing an Inter Platform Semantic Mediator (IPSM) component that performs semantic translation of semantically-annotated data, based on *alignments* to and from a deployment specific *central ontology* (for details concerning the IPSM and the central ontology, see [7], [6], [8]). In this context, in [9] a *format* for representing alignments between ontologies has been proposed. In this work, we discuss what information should be included in the alignment, to be able to perform translation of geospatial data. We also show what actual "help" can one expect to obtain in establishing alignments between ontologies, from two of the best tools available for ontology matching that were initially selected in [10].

Use of geospatial information, as practical example of ontology matching, provides context for a more general discussion of issues that have to be confronted in real-world IoT deployments. Geospatial data was selected, because of its common usage to describe observation from different domains, and existence of several popular standards for data representation. However, results presented here naturally generalize beyond the geospatial case. Scenarios, in which different systems collect (geospatial) data and store them in different formats; and where such data later needs to be exchanged (and understood by parties participating in the communication process), can

[1]Keynote presentation and follow-up discussion during "5th Business Information Systems" conference, Poznan, Poland, April, 2002

[2]http://www.cyc.com/

[3]Projects grouped within the IoT-EPI initiative http://iot-epi.eu/

be easily thought of. Even though, the syntax is different in each case, the translation process should ensure that semantics, of exchanged data, is preserved. For example, lets consider a situation when data is send from sensor(s) placed on truck(s) to the IoT platform of a haulier company. Each transmitted observation is described with a device identifier, measured property e.g. temperature, and the actual geographic location of the truck. Another, analytic, IoT system may be subscribed to this data stream, to receive observations from the haulier IoT platform, to analyze truck route patterns. However, the analytic system works with geospatial data represented in a different standard/format. Let us now assume that the analytic platform serves as a data source for other IoT artifacts, e.g. port IoT system that deals with information about positions of trucks (only) within the port area. Because of technical limitations, and already existing formal agreements, the port system is not going to be connected directly to the haulier company IoT system. Here, it is quite possible that the port system uses yet another representation of geospatial information. It should be clear that, in this example, it would be necessary to perform at least two translations of geospatial data (from the haulier company platform to the analytic system and from the analytic platform the the port platform), so that the sensor data would be understood across the ecosystem. While this example may seem somewhat unrealistic, existence of real-world need for translation of geospatial data is real.

A. *Alignments and their representations*

To be able to discuss semantic interoperability, it is necessary to introduce the concepts of ontology alignment, matching, mapping, and translation. These terms are interrelated and sometimes used interchangeably (even if this may not be the right thing to do). To properly distinguish their meaning, we define them as follows (see, also, [5]).

Ontology alignment, refers to the process of *finding correspondences between two, or more, ontologies*. The result of this process consists of, one or more, captured *correspondences* between particular entities, or groups of entities and sub-structures. A correspondence can be either a predicate about similarity, called a *matching*, or a logical axiom – a *mapping*. Typically used mapping axioms are equivalences and subsumptions. In practice, when ontology alignment tools are used to specify alignments, they often state a degree of confidence for every discovered/proposed correspondence. An equivalence axiom, with a degree of confidence, is very close in meaning to a predicate about similarity (a matching). Note that, terms "mapping" and "matching" are often not properly distinguished, in the terminology used by existing alignment tools. Therefore, in the literature, and in practice, a set of correspondences can be called "alignment", "matching", or "mapping". In what follows, we have chosen to use the most widespread term – "alignment".

Ontology translation, or, more precisely, *semantics translation*, is a *process of changing the underlying semantics of a piece of knowledge*, meaning, describing the knowledge in different terms, possibly coming from a different ontology. The fundamental use case is translation of messages that takes place during communication. In general, information described semantically, in terms of a *source ontology*, is transformed into information described in terms of a *target ontology*. As a result, the source semantics (contained in the translated message) is fully replaced with the target semantics. Naturally, good quality translation should preserve as much of the meaning of the translated content as possible (the ideal case being a "lossless translation").

Semantic translation is a natural case for application of ontology alignment(s). Here, the goal is to enable one-way (or two-way) "understanding" between software artifacts that implement differing semantics. This is directly applicable, among others, to the Internet of Things. For instance, IoT ontologies: OpenIoT[4], SAREF[5], and oneM2M[6], have similar scope (see, also [8]). The goal of semantic translation would thus be, for instance, allowing a platform that uses OpenIoT to seamlessly communicate (one-way or bidirectionally) with a platform that represents its data using SAREF.

Existing ontology alignment tools (for a comprehensive summary of existing tools, see [10]) produce output in various formats, including a very popular Alignment API format [2]. Alignment API format can be defined on several levels: 0 – where matched are entities identified by URIs, 1 – in which matched are *sets* of entities identified by URIs (not used[7]), 2 – where matched are more structured entities that may be represented in RDF/XML. In case of level 2 it is necessary to further identify the structure of source and target entities. Results, returned by the researched tools, are usually expressed on level 0, i.e. only simple URI-to-URI mappings are considered. EDOAL [1] (which is on level 2), on the other hand, is capable of expressing complex alignments, i.e. mappings between complex descriptions with multiple restrictions.

Here, it is perhaps worth stressing that the translation procedure is not "symmetric". If one wants to translate messages bidirectionally, two separate translation procedures need to be defined. Hence, a pair of separate one-way alignments enables two-way translation.

B. *Geospatial data representation standards*

As stated earlier, results presented in this paper are based on examples of use of geospatial data in semantic translations. Let us, therefore, briefly introduce selected standards for representation of geospatial data. In what follows, when presenting examples we use Turtlen notation[8] as it seems widely accepted as the most concise and readable serialization of RDF. For completeness, the list of used prefixes can be found in the Appendix section V. Here, let us start by introducing the

[4]https://github.com/OpenIotOrg/openiot

[5]https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology

[6]http://www.onem2m.org/technical/onem2m-ontologies

[7]Based on information provided by the authors of http://alignapi.gforge.inria.fr/format.html

[8]https://www.w3.org/TR/turtle/

most popular standards, available for representing geospatial information:

- **Basic Geo (WGS84 lat/long) Vocabulary**[9] – this is the basic, W3C recommended, vocabulary for describing points on the map, using latitude, longitude, and altitude properties (parts of the WGS84 reference datum specification) and other information about spatially-located "things". The *rdfs:range* of the *geo:lat*, *geo:long* and *geo:alt* properties are the XML Schema float datatypes. Listing 1 shows an instance representing a point with specified coordinates. Such an instance can be used as a value for a measurement value property, for instance as a part of the information about location of a truck, in the above example.

```
[] a geo:Point ;
    geo:lat "55.701" ;
    geo:long "12.552" .
```

Listing 1. Example of geospatial data in Basic Geo Vocabulary

Note: from now on, we shall use the short name **WGS84** to denote the **Basic Geo (WGS84 lat/long) Vocabulary**.

- **GeoSPARQL**[10] – defines a vocabulary for representing geospatial data in RDF, as well as an extension to the SPARQL query language, introduced for processing geospatial data. It allows representing a collection of complex geometries and issue meaningful geospatial queries such as, for example, how many objects are located within 3 km distance of the a selected location? Here, two fundamental concepts are: *Feature* (an entity in the world, with a spatial location) and *Geometry* (a geometric shape, such as a point, polygon, or line; used as a representation of a Feature's spatial location). These two concepts are related, via the *hasGeometry* property. For the *Geometry* instances, coordinates can be specified using the *asWKT* property. Listing 2 shows geospatial data, of a sample instance, representing a Monument object, which is a *Feature*. It has a geometry representing its location, with specific WKT (Well Known Text[11]) coordinates. Obviously, GeoSPARQL can also be used to represent position of any other object (e.g. a truck within port premises).

```
ex:PointOfInterest a owl:Class ;
    rdfs:subClassOf geosparql:Feature .

ex:XYZMonument a ex:PointOfInterest ;
    rdfs:label "XYZ Monument" ;
    geosparql:hasGeometry ex:XYZPoint .

ex:XYZPoint a geosparql:Point ;
    geosparql:asWKT "POINT(-70.12345
        50.875382)"^^geo-sf:WKTLiteral .
```

Listing 2. Example of geospatial data in GeoSPARQL

- **GeoRSS**[12] – Geographically Encoded Objects for RSS Feeds format was designed as a lightweight, community driven, approach to extend existing feeds with geographic information. There are two encodings of the GeoRSS: *Simple* (which supports basic geometries, e.g. points, lines, boxes, polygons, and covers the typical use cases, when encoding locations), and *GML* (which supports a greater range of features, and coordinate reference systems, other than the WGS84 latitude/longitude/altitude). Besides GeoRSS definition as XML Schema, there exists a Geo OWL ontology, which closely matches the GeoRSS feature model, and utilizes the existing GeoRSS vocabulary for geographic properties and classes. Here, the two main concepts are *gml:_Feature* and *gml:_Geometry* (subclasses representing GML objects with corresponding properties). They are related with the *georss:where* property. Subproperties of *georss:where* represent GeoRSS Simple and include, for instance the *georss:Point*. Each of these properties takes a literal list of doubles as their range, but they are equivalent, in definition, to *georss:where* plus the corresponding GeoRSS GML class, and its properties. Properties *georss:lat* and *georss:long* are included as subproperties of *georss:where*, but are treated together as the equivalent of: (i) *georss:where* plus *gml:Point* with *gml:pos*, and (ii) *georss:point*. Listings 3 and 4 show two ways of representing the same information, i.e. location of a Monument object. The first one uses a property *georss:point*, to specify coordinates in a textual format. The second one is similar to the example from Listing 2, where an instance representing a Monument object is declared as a *gml:_Feature*; however, in the case of GeoRSS, it has *georss:where* property, with value being an instance of the *gml:Point* class. The *gml:Point* has specific coordinates, defined with the *gml:pos* property. As above, this format of representing geospatial information could be used also in the "truck + analytics + port" example, introduced above.

```
ex:XYZMonument a georss:_Feature .

ex:XYZMonument dc:title "XYZ Monument" ;
    georss:point "-70.12345 50.875382" .
```

Listing 3. Example of geospatial data in GeoRSS 1

```
ex:PointOfInterest a owl:Class ;
    rdfs:subClassOf gml:_Feature .

ex:XYZMonument a gml:PointOfInterest ;
  georss:where ex:XYZPoint ;
  dc:title "XYZ Monument" .

ex:XYZPoint a gml:Point ;
    gml:pos "-70.12345 50.875382" .
```

Listing 4. Example of geospatial data in GeoRSS 2

Notice that, in this paper, we are not interested in comparison of different syntaxes that can be used in the context of geospatial data representation, such as

[9] https://www.w3.org/2003/01/geo/
[10] http://www.opengeospatial.org/standards/geosparql
[11] http://docs.opengeospatial.org/is/12-063r5/12-063r5.html
[12] http://www.georss.org/

WKT[13], GML[14], KML[15], GeoJSON[16], and so on. We focus exclusively on semantics. Moreover, for the sake of brevity and clarity of examples, we chose the semantics that are both popular and allow us to clearly present our findings. Thus, other popular vocabularies and ontologies that can be used to describe location, such as schema.org[17], DC terms[18], VCard[19], LOCN (ISA Location core vocabulary)[20] and others were not included. Nevertheless, the following discussion, and the findings naturally generalize also to these cases.

Even though the aforementioned geospatial data representation formats are quite straightforward to understand, and mapping them onto each other seems natural (and, thus, easy to achieve), defining formal translation rules is not that obvious. Here, note that, in the IoT domain, it is quite often the case that translation is performed not in the case of "isolated geospatial data", but includes also context information e.g. metadata about the device that sends the geolocation data. Nevertheless, we chose to use geospatial data as an example, to visualize what problems arise for such a simple case of translation. With more complex examples, the procedure gets even more involved.

II. Matching geospatial ontologies using available tools

Let us now show how existing tools, created to establish alignments between ontologies, can be used with geospatial data, represented using the, above summrized, standards. When one would like to create an ontology alignment, to be able to perform semantic translation, the first task is to map concepts from the source ontology to the target ontology. In [10], we have presented the results of our research on available tools, which can be used for ontology matching. Here, we are going to analyze the output produced by two of them: **LogMap** and **AgreementMakerLight**, when applied to the selected geospatial vocabularies. Out of the tools considered in [10] these are the ones that not only actually work, but also are still actively maintained, produce alignments, and accept RDF and OWL as input. Here, our goal was to investigate to what extent these tools can *help* producing an alignment between two selected ontologies. The following cases have been considered: *i*) mapping from WGS84 to GeoSPARQL, *ii*) mapping from GeoSPARQL to GeoRSS *iii*) mapping from GeoRSS to WGS84

Let us note that ontologies, introduced in Section I-B, are more complex than the examples presented in forthcoming Section III-B, where we focus only on translation of instances of points described with coordinates. However, here we will deliberately focus on a rather simplistic use case, when we want to translate only information about the coordinates of a given point, e.g. in case of sharing information about a position observation for a given device (i.e. sensor in a truck). This information is to be shared, among IoT artifacts that are using different ontologies to represent such information, as it was briefly presented in Section I. Let us note, however that, typically, with growing complexity and dissimilarity of ontologies that are to be matched, success of automatic matching decreases. Therefore, results from the simplistic use case, considered here, provide an "optimistic" assessment of actual usefulness of considered tools (in the context of the introduced task).

Ideally, if the tools were able to match human knowledge/expertise, the first case (mapping WGS84 $\mapsto$ GeoSPARQL) should result in a suggestion of mapping between *Point* concepts from both ontologies. Specifically, values of the WGS84 properties *geo:lat* and *geo:long* should be suitably "concatenated", to obtain the value of the *geosparql:asWKT* property. In the second case (mapping GeoSPARQL $\mapsto$ GeoRSS), the concepts of features, geometries and shapes, from both ontologies, should be included in the mapping. The *geosparql:hasGeometry*, *geosparql:asWKT* properties could be indicated as equivalent to the *georss:where* and *gml:pos* properties, respectively. In the latter case the equivalence relation depends on geospatial encoding used, i.e. one should distinguish between WKT and GML coordinates. In the third case (mapping GeoRSS $\mapsto$ WGS84), concepts of a point, and properties representing longitude and latitude, should be indicated as potential match candidates.

LogMap[21] – is an open-source tool that maps ontologies and returns result in OWL, text, or in the OAEI Alignment Format. The Alignment Format is at level 0 (alignments, in which matched entities are identified by their URIs). Tests were done with LogMap that is available at the University of Oxford website [22], with mapping repair option turned on. Listing 5 shows results from LogMap for the WGS84 $\mapsto$ GeoSPARQL mapping. The result contains only one mapping, for the concept of *Point*.

```
<map><Cell>
  <entity1 rdf:resource="http://www.w3.org
      /2003/01/geo/wgs84_pos#Point"/>
  <entity2 rdf:resource="http://www.opengis.net/
      ont/sf#Point"/>
  <measure rdf:datatype="xsd:float">0.78</measure>
  <relation>=</relation>
</Cell></map>
```

Listing 5. LogMap result: WGS84 $\mapsto$ GeoSPARQL

Listing 6 shows results from LogMap for the GeoSPARQL $\mapsto$ GeoRSS mapping. The results contain correct suggestions for concept matches (e.g. *Geometry*, *Feature*, *Point*), however no suggestions for properties (e.g. *hasGeometry*) are returned.

```
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/geosparql#Geometry"/>
```

[13]http://docs.opengeospatial.org/is/12-063r5/12-063r5.html
[14]http://www.opengeospatial.org/standards/gml
[15]http://www.opengeospatial.org/standards/kml/
[16]http://geojson.org/
[17]http://schema.org/geo
[18]purl.org/dc/terms/Location
[19]https://www.w3.org/TR/vcard-rdf/#d4e239
[20]https://www.w3.org/ns/locn
[21]https://www.cs.ox.ac.uk/isg/tools/LogMap/
[22]http://krrwebtools.cs.ox.ac.uk/logmap/

```
  <entity2 rdf:resource="http://www.opengis.net/
      gml/_Geometry"/>
  <measure rdf:datatype="xsd:float">0.8</measure>
  <relation>=</relation>
</Cell></map>
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/geosparql#Feature"/>
  <entity2 rdf:resource="http://www.opengis.net/
      gml/_Feature"/>
  <measure rdf:datatype="xsd:float">0.7</measure>
  <relation>=</relation>
</Cell>
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/gml#Point"/>
  <entity2 rdf:resource="http://www.opengis.net/
      gml/Point"/>
  <measure rdf:datatype="xsd:float">0.62</measure>
  <relation>=</relation>
</Cell></map>
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/sf#Point"/>
  <entity2 rdf:resource="http://www.opengis.net/
      gml/Point"/>
  <measure rdf:datatype="xsd:float">1.0</measure>
  <relation>=</relation>
</Cell></map>
```

Listing 6. LogMap result: GeoSPARQL ↦ GeoRSS

Listing 7 shows results from LogMap for the GeoRSS ↦ WGS84 mapping. Here, again only the concept of a *Point* was mapped.

```
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      gml/Point"/>
  <entity2 rdf:resource="http://www.w3.org
      /2003/01/geo/wgs84_pos#Point"/>
  <measure rdf:datatype="xsd:float">0.5</measure>
  <relation>=</relation>
</Cell></map>
```

Listing 7. LogMap result: GeoRSS ↦ WGS84

In listing 7, concepts are also matched, but the measure of reliability of the result is only 0.5, and properties are not matched. Overall, this is not a surprising result, considering that LogMap matches lexically, inspecting URIs and *rdfs:label* property values. For properties, such as *geosparql:asWKT*, lexical match with other, similar, properties would be surprising, because its name is unique and dissimilar to, for instance, *gml:pos*.

AgreementMakerLight[23] – is open source and extensible matching system, with results produced in Alignment API format at level 0. Tests were performed on AML implementation available in GitHub[24]. Default settings were used, but with WordNet check turned off for the Background Knowledge Matcher (checks against DOID and UBERON were done). This was done because WordNet check returned an execution error and we have decided to not to "dig into this issue" to find the reason why (it was not crucial to this contribution). Matchers turned on by default included: background knowledge matcher, string matcher, word matcher, structural matcher, property matcher, cardinality matcher and coherence matcher.

Listing 8 shows a result from the AML for the WGS84 ↦ GeoSPARQL mapping. It contains a mapping between concepts representing points in both ontologies.

```
<map><Cell>
  <entity1 rdf:resource="http://www.w3.org
      /2003/01/geo/wgs84_pos#Point"/>
  <entity2 rdf:resource="http://www.opengis.net/
      ont/sf#Point"/>
  <measure rdf:datatype="http://www.w3.org/2001/
      XMLSchema#float">0.9801</measure>
  <relation>=</relation>
</Cell></map>
```

Listing 8. AML result: WGS84 ↦ GeoSPARQL

Listing 9 shows a result from the AML for the GeoSPARQL ↦ GeoRSS mapping, with selected mappings between concept in the ontologies. In this case, aforemenioned settings were used but without property matcher that returned an execution error (here, again, we have not investigated the reason for this behavior).

```
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/geosparql#Feature"/>
  <entity2 rdf:resource="http://www.opengis.net/
      gml/_Feature"/>
  <measure rdf:datatype="http://www.w3.org/2001/
      XMLSchema#float">0.99</measure>
  <relation>=</relation>
</Cell></map>
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/sf#Geometry"/>
  <entity2 rdf:resource="http://www.opengis.net/
      gml/_Geometry"/>
  <measure rdf:datatype="http://www.w3.org/2001/
      XMLSchema#float">0.9801</measure>
  <relation>=</relation>
</Cell></map>
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      ont/sf#Point"/>
  <entity2 rdf:resource="http://www.opengis.net/
      gml/Point"/>
  <measure rdf:datatype="http://www.w3.org/2001/
      XMLSchema#float">0.9801</measure>
  <relation>=</relation>
</Cell></map>
```

Listing 9. AML result: GeoSPARQL ↦ GeoRSS

In listing 10 showing the results of GeoRSS ↦ WGS84 mapping, one can see that, once more, properties have not been matched, only concepts matching was suggested. Here, again, the property matcher was turned on.

```
<map><Cell>
  <entity1 rdf:resource="http://www.opengis.net/
      gml/Point"/>
  <entity2 rdf:resource="http://www.w3.org
      /2003/01/geo/wgs84_pos#Point"/>
  <measure rdf:datatype="http://www.w3.org/2001/
      XMLSchema#float">0.99</measure>
  <relation>=</relation>
</Cell></map>
```

Listing 10. AML result: GeoRSS ↦ WGS84

[23]http://alignapi.gforge.inria.fr/
[24]https://github.com/AgreementMakerLight/AML-Jar

In summary, when in need to find suggestions for mappings between ontologies, the selected tools did not turn out to be very useful. This statement is, obviously, based on a rather weak foundation. Nevertheless, it provides a warning for those who plan to use these tools in practice.

III. INTER-IoT approach to alignment representation

Let us now proceed to the question, how can we represent more complicated alignments than the ones that the tools discussed in the previous section were able to construct/produce.

A. INTER-IoT Alignment Format

As mentioned above, in the INTER-IoT project, it was assumed that semantic interoperability will be based on use of ontology alignments [6], [9]. This being the case, in the context of this contribution, it is crucial to specify how alignments will be represented. Results produced by the tools, presented in Section II, are given in the Alignment API format, at level 0. Even in such a simple case, as geospatial data translation, level 0 turns out to be insufficient. There is no property matching, and no rules are/can be defined for coordinates transformation (for instance, concatenating values of two properties into a single string, splitting the property value into two target properties values). If the Alignment API format was to be used, then it should be considered at level 2. However, unfortunately, at this level it is very extensive and no tools that can parse and execute translations, specified using it, exist (to the best of our knowledge). Therefore, we have developed our own format, with an XML representation (defined in XSD), based on the Alignment API [4] and inspired, to some extent, by the EDOAL [1].

Listing 11, presents the general structure of the INTER-IoT alignment format (also called IPSM alignment format). The format is used by the, mentioned above, IPSM component that performs semantic translations. In principle, the alignment element describes a uni-directional set of translation rules comprised of independent mapping cells, each of which has an "input" and "output" entity descriptions. Elements `<onto1>` and `<onto2>` describe the "source" and "target" ontologies of the alignment, by giving their URIs and specifying the formalism used for their definition (e.g., OWL 2.0). The sample structure of an alignment is presented in listing 11.

```
<Alignment id="align_id" version="align_version"
    creator="align_creator" description="
    align_desc">
<onto1> { source ontology info } </onto1>
<onto2> { target ontology info } </onto2>
<steps>
  <step order="1" cell="cell_k"/>
  { more steps }
</steps>
<map>
  <Cell id="cell_id">
    <entity1> { source RDF pattern } </entity1>
    <entity2> { target RDF pattern } </entity2>
    <transformation>
      { functional constraints }
    </transformation>
    <filters> { datatype constraints } </filters>
    <typings> { typing info } </typings>
  </Cell>
  { more Cells }
</map>
</Alignment>
```

Listing 11. INTER-IoT alignment format – general structure

In Listing 11, the `<steps>` element defines the (default) `order`, in which *cells* of the alignment will be subsequently applied in the message transformation process (each `<step>` refers to a cell identifier, as given by the `id` attribute of the `<Cell>` element). Each cell represents a match from `<entity1>` into `<entity2>`. Both entities should be valid RDF graphs (presented in the RDF/XML serialization), possibly containing special-purpose nodes "variables", which are to be bound and referenced within the transformation.

Variables from `<entity1>` can be used in `<entity2>`, which denotes making a simple copy of whatever value, or entity, is stored in the variable at the runtime. To give a "deeper" meaning to the "variable" elements, it is necessary to add some *constraints*, which would define them in terms of values of other variables, or simple values (e.g. as a result of concatenation). This is the role of the cell's (optional) `<transformation>` element. The content of this element is a sequence of *functional constraints* (given by `<function>` elements). Each constraint is of the form

$$\mathtt{fun}(\mathtt{arg1}, \ldots, \mathtt{argN}) = \mathtt{res}$$

where `fun` is a SPARQL *function*, referenced by the `about` attribute of the `<function>` element, `arg1`, ..., `argN` are *arguments* (described by the `<param>` elements), and `res` (given by the `<return>` element) is the *result* of executing the function with given arguments. Both these arguments, and the result, might refer to the "variable" elements.

The (optional) `<filters>` and `<typings>` elements, add datatype information to the variable elements, from the source and the target RDF graph patterns, respectively.

B. Alignments between geospatial data standards

Let us now present how alignments, between simple coordinate representations, in different standards (summarized above), look like. Note that these alignments were prepared by hand.

In Listing 12 an alignment, in the IPSM format that can be used for translating, from the WGS84 to the GeoSPARQL, is presented.

```
<Cell id="cell_1">
  <entity1>
    <sripas:node_CTX>
      <geo:lat>
        <sripas:node_x/>
      </geo:lat>
      <geo:long>
        <sripas:node_y/>
      </geo:long>
    </sripas:node_CTX>
  </entity1>
  <entity2>
    <sripas:node_CTX>
      <geosparql:asWKT>
```

```
        <sripas:node_z/>
      </geosparql:asWKT>
    </sripas:node_CTX>
  </entity2>
  <relation>=</relation>
  <transformation>
    <function about="str">
      <param order="1" about="&sripas;node_x"/>
      <return about="&sripas;node_sx"/>
    </function>
    <function about="str">
      <param order="1" about="&sripas;node_y"/>
      <return about="&sripas;node_sy"/>
    </function>
    <function about="concat">
      <param order="1" val="Point("/>
      <param order="2" about="&sripas;node_sx"/>
      <param order="3" val=" "/>
      <param order="4" about="&sripas;node_sy"/>
      <param order="5" val=")"/>
      <return about="&sripas;node_z"/>
    </function>
  </transformation>
  <filters>
    <filter about="&sripas;node_x" datatype="&xsd;
        float"/>
    <filter about="&sripas;node_y" datatype="&xsd;
        float"/>
    <filter about="&sripas;node_sx" datatype="&xsd
        ;string"/>
    <filter about="&sripas;node_sy" datatype="&xsd
        ;string"/>
    <filter about="&sripas;node_z" datatype="&geo-
        sf;wktLiteral"/>
  </filters>
  <typings>
    <typing about="&sripas;node_z" datatype="&geo-
        sf;wktLiteral"/>
  </typings>
</Cell>
```

Listing 12. Alignment: WGS84 ↦ GeoSPARQL

In the listing, in `<entity1>` and `<entity2>`, source and target structures, presented in RDF/XML, are defined. Note that variable nodes are used, to indicate subjects and objects of RDF triples. In this case, all instances that have properties *geo:lat* and *geo:long* (with values denoted by variables `<node_x>` and *node_y* bound to float data type) will be translated. The translated RDF graph will contain the same instance (preserved URI) with the *geosparql:asWKT* property, with value constructed from the concatenated source properties' values. The datatype of the resultant text should be *wktLiteral*, and this is indicated in the `<typing>` element. Observe also that the possibility to nest functions is represented. Specifically, the STR function is called first, to cast float values to strings and, as the result of the call, the "intermediate" variables `<node_sx>` and *node_sy* are defined. They should be, as any variable, and have datatype indicated in the `<filters>`.

Moving ahead, in listing 13, an alignment translating data from the GeoSPARQL to the GeoRSS is presented.

```
<Cell id="cell_1">
  <entity1>
    <sripas:node_CTX>
      <geosparql:asWKT>
        <sripas:node_x/>
      </geosparql:asWKT>
    </sripas:node_CTX>
  </entity1>
  <entity2>
    <sripas:node_CTX>
      <georss:Point>
        <sripas:node_z/>
      </georss:Point>
    </sripas:node_CTX>
  </entity2>
  <relation>=</relation>
  <transformation>
    <function about="str">
      <param order="1" about="&sripas;node_x"/>
      <return about="&sripas;node_sx"/>
    </function>
    <function about="replace">
      <param order="1" about="&sripas;node_sx"/>
      <param order="2" val="Point\\((\\d+\\.\\d+)
          \\s+\\d+\\.\\d+\\)"/>
      <param order="3" val="$1"/>
      <param order="4" val="i"/>
      <return about="&sripas;node_rx"/>
    </function>
    <function about="replace">
      <param order="1" about="&sripas;node_sx"/>
      <param order="2" val="^Point\\(\\d+\\.\\d+\\
          s+(\\d+\\.\\d+)\\)$"/>
      <param order="3" val="$1"/>
      <return about="&sripas;node_ry"/>
    </function>
    <function about="concat">
      <param order="1" about="&sripas;node_rx"/>
      <param order="2" val=" "/>
      <param order="3" about="&sripas;node_ry"/>
      <return about="&sripas;node_z"/>
    </function>
  </transformation>
  <filters>
    <filter about="&sripas;node_x" datatype="http:
        //www.opengis.net/def/sf/wktLiteral"/>
    <filter about="&sripas;node_sx" datatype="&xsd
        ;string"/>
    <filter about="&sripas;node_y" datatype="&xsd;
        string"/>
    <filter about="&sripas;node_rx" datatype="&xsd
        ;string"/>
    <filter about="&sripas;node_ry" datatype="&xsd
        ;string"/>
  </filters>
</Cell>
```

Listing 13. Alignment: GeoSPARQL ↦ GeoRSS

Here, `<entity1>` defines the pattern for the source RDF graph, where all instances with the *geosparql:asWKT* property are selected (naturally, they will be instances of the *geosparql:Geometry* class). Once more, using nested function calls, the values of coordinates are parsed from the *wktLiteral*, and concatenated, once more, to produce value for variable *node_z* that will become an instance of the *georss:Point* property value.

Let us notice that, even though the geospatial data is quite a trivial example for semantic translation, defining alignment cells requires some knowledge, e.g. about SPARQL functions. Overall, this process should be supported by a well documented procedure e.g. due to the complexity of defined transformations.

IV. Concluding remarks

Even though there are tools available that simplify "understanding" of ontologies and help to identify their concepts that can be mapped onto each other, the analysis and design of rules, required to facilitate semantic translation is still a complex task. The crucial step, here, is to formalize the translation as an alignment. The alignment format specified in the INTER-IoT project allows to express simple RDF graph transformations, as well as more complex ones that can involve calling a function on defined input variables. Unfortunately, as a side effect of this generic method of alignment representation, even in such simple cases as translation of geospatial data, the alignment is quite extensive. On the other hand, the great advantage of the INTER-IoT alignment format is the fact that it is read and executed by the IPSM component. This cannot be claimed about EDOAL that is very expressive, however, we have not found any tool that is able to "produce", or "consume" and execute defined translations. Note that, the original level 0 Alignment API format can be easily translated to ITER-IoT alignment format, because of the similarity in structure.

The examples we have presented expose the truth about automatic alignment methods. We conclude that they are not developed to the extent that is needed to produce alignments usable in our use case. Instead, their outputs may be considered as a first (and very rough) approximation, or a guideline for making a more advanced alignment that can be applied in actual semantic translation. In a broader sense, we cannot rely on currently existing automated methods to find the alignments that would be useful in real-world practice. The simple example of geolocation (i.e. *Point* information in different ontologies) shows that automatically generated alignments are not sufficient when one needs to establish interoperability between different semantics. What follows, is that the dream of fully interoperable Linked Data that makes use of varied ontologies to describe overlapping domains will still, for some time, be a challenge.

V. Prefixes

The declarations for prefixes used in the paper:

```
@prefix geo: <http://www.w3.org/2003/01/geo/
    wgs84_pos#>.
@prefix geosparql: <http://www.opengis.net/ont/OGC
    -GeoSPARQL/1.0/>.
@prefix geo-sf: <http://www.opengis.net/def/
    dataType/OGC-SF/1.0/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix ex: <http://example.org/PointOfInterest#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix gml <http://www.opengis.net/gml/>.
@prefix georss: <http://www.georss.org/georss/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
    syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-
    schema#>.
@prefix sripas: <http://www.inter-iot.eu/sripas#>.
```

Listing 14. Prefixes used

Acknowledgment

This research was partially supported by the European Union's "Horizon 2020" research and innovation programme as part of the "Interoperability of Heterogeneous IoT Platforms" (INTER-IoT) project under Grant Agreement No. 687283.

References

[1] EDOAL: Expressive and declarative ontology alignment language. http://alignapi.gforge.inria.fr/edoal.html.

[2] A format for ontology alignment. http://alignapi.gforge.inria.fr/format.html.

[3] INTER-IoT Project. http://www.inter-iot-project.eu.

[4] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The Alignment API 4.0. *Semantic Web*, 2(1):3–10, jan 2011.

[5] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2 edition, 2013.

[6] Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Semantic interoperability in the Internet of Things: an overview from the INTER-IoT perspective (in press). *Journal of Network and Computer Applications*, 2016.

[7] Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. *Towards semantic interoperability between Internet of Things platforms (submitted for publication)*. Springer, 2016.

[8] Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Towards common vocabulary for IoT ecosystems—preliminary considerations. In *Intelligent Information and Database Systems, 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3-5, 2017, Proceedings, Part I*, volume 10191 of *LNCS*, pages 35–45. Springer, 2017.

[9] Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Katarzyna Wasielewska. Streaming semantic translations. In *21st International Conference on System Theory, Control and Computing ICSTCC, Proceedings*, in press.

[10] Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, Katarzyna Wasielewska, and Giancarlo Fortino. Tools for ontology matching—practical considerations from INTER-IoT perspective. In *Proc. of the 8th Int. Conference on Internet and Distributed Computing Systems*, volume 9864 of *LNCS*, pages 296–307. Springer, 2016.