Mappings Representation in Ontologies

Olga Kovalenko[1], Christophe Debruyne[2], Estefanía Serral[1], and Stefan Biffl[1]

[1]Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology
Favoritenstrasse 9-11/E188, A-1040 Vienna
`firstname.lastname@tuwien.ac.at`
[2]Semantics Technology and Applications Research Laboratory
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussel
`firstname.lastname@vub.ac.be`

Abstract. Ontology mapping is needed to explicitly represent the relations between several ontologies, which is an essential task for applications such as semantic integration and semantic interoperability, data exchange and data transformation. Currently, there is no standard for representing mappings. However, there are a number of technologies that support the representation of mappings between Semantic Web ontologies. In this paper we investigate the existing technologies regarding the support provided to represent complex mappings between different ontologies required in the data integration scenario. We introduce a set of mappings categories that were identified based on requirements for the data integration projects of an industry partner. An evaluation of available technologies for mappings representation regarding the support for introduced mappings categories have been performed. The results of the evaluation show that the SPARQL Inference Notation (SPIN) would fit the best in the requirements.

1 Introduction

Ontologies are widely used to obtain semantic integration and semantic interoperability among systems and applications. For the majority of applications it is not enough to have one ontology, but instead a set of ontologies related with each other in a certain way must be handled. For example, in the semantic integration scenario, a set of local ontologies must be related with a global ontology. Another example is when different people modeled the same domain differently and the correspondences must be defined between the entities of the resulting ontologies. For such applications it is necessary to explicitly specify the relations and correspondences between different ontologies. The common technique here is to define a set of mappings that bind a set of entities in one ontology to the entities in another ontology. Based on these mappings various further tasks can be performed, e.g., query rewriting, data transformation from one ontology to another, or ontology merging.

To our knowledge there is no standard or conventional technique on how to represent mappings in ontologies. The Ontology Web Language[1] (OWL), a W3C recommendation for building ontologies in the Semantic Web, provides some support to express mappings between the entities of different ontologies [11]. These mapping are stored as a part of the ontology and are coupled with this ontology [25]. However, the expressivity of mappings in OWL is restricted to rather simple mappings, such as one-to-one mapping for ontology concepts and properties. Differences in granularity or understanding of the domain semantics require support for more sophisticated mappings and cannot be expressed in pure OWL [11]. In order to support defining more complex mappings between ontologies various technologies have been developed, e.g., the Datalog language, the Semantic Web Rule Language (SWRL)[2], SPARQL CONSTRUCT queries[3], and the SPARQL Inference Notation (SPIN)[4]. The degree of support for defining mappings between ontologies provided by these technologies vary in many aspects, e.g., by the expressivity for mappings representation and on how mappings are stored (inside the ontology or apart from the ontology), which must be taken into account for choosing the appropriate technology for a specific task. It is also important to note that the selection of an optimal technique for mappings representation will strongly depend on the application at hand, e.g., the best fitting technology for ontology merging or for semantic integration can be different [26].

In this paper we investigate which categories of complex mappings between ontologies are required in a typical data integration scenario in the application context of multi-disciplinary engineering [**?**], where the data from local ontologies must be transformed into the top level ontology, to derive a benchmark for comparing the abilities of technologies to express these complex mappings. The contribution of this paper is twofold: firstly, we categorize mappings that are required in the projects of an industry partner to fulfill effective and efficient data integration; and, secondly, we evaluate the available technologies for mappings representation regarding the support provided for introduced mappings categories. Based on performed evaluation a recommendation is given on what technology will fit better in the described application scenario.

In this paper we focus on mapping representation, without considering a problem of finding similarities between ontologies, but instead investigating what are the requirements regarding the expressivity of ontology mappings and which of current technologies are able to satisfy them. We also leave beyond the investigation the problem of mappings definition between ontologies, which are described using different representational mechanisms, considering it as a direction for future work.

The rest of the paper is organized as follows: Section 2 describes the related work in the field of ontology mapping and also explicitly explains the understand-

[1] OWL overview: http://www.w3.org/TR/owl-features/
[2] SWRL specification: http://www.daml.org/2003/11/swrl/
[3] SPARQL specification: http://www.w3.org/TR/rdf-sparql-query/
[4] SPIN overview: http://www.w3.org/Submission/spin-overview/

ing of important terms, such as mapping and alignment, which is used in the paper in order to prevent ambiguity and misunderstandings with the terminology. In Section 3 we present the application scenario from our industry partners that is used across the paper to illustrate introduced mappings categories and motivates the choice of optimal technology for mappings representation. Section 4 determines a set of mappings categories that may be needed in the described use case scenario. Section 5 discusses the technologies that are currently available to represent mappings in OWL ontologies. Finally, before concluding the paper, we evaluate existing mappings technologies regarding the support for defined mappings categories and give a recommendation which technology would be the better choice in the presented use case scenario in Section 6.

2 Ontology Mapping: State-of-the-art and Terminology Disambiguation

There is a surprising variety, close to confusion, in the terminology used in literature regarding ontology mapping [6,9]. Different disciplines have arisen, studying slightly different or overlapping aspects of the ontology mapping process, e.g., ontology alignment, ontology merging and ontology matching. Because of the interlinked nature of these disciplines, considerable ambiguity in the terminology exists in the literature: the definitions of foundational notions, such as ontology alignment, mapping, etc. given by different authors seem confusing, inconsistent, or even contradicting [6]. The goal of this section is to bring an insight on the state-of-the-art in the field of ontology mapping and to explicitly specify the understanding of important notions in this area, which will be used in this paper. While determining the meaning of basic terms, we tried to follow the most common and used by the majority of authors understanding of a certain term.

2.1 Ontology Mapping: Foundational Definitions

We define our understanding of ontology alignment similarly to Flouris et al: ***Aligning*** *two ontologies corresponds to the process of finding for an every entity (concept, property or instance) in one ontology a corresponding entity, which expresses the same semantics, in another ontology.* An ***alignment*** therefore *is one-to-one correspondence between specific entities of two ontologies.* It might happen that for some entities no corresponding entity can be found [6].

While alignment only helps to identify which concept in two ontologies are related, mappings specify the connection between entities in enough details that it will be possible to apply or execute them for a certain task [6]. Mapping captures the semantic and structural relationships between the entities of two ontologies.

We will follow the definition of ontology mapping applied in [20] and [30]: *A* ***mapping*** *between two ontologies O_1 and O_2 is a 5-tuple $\langle id, E_1, E_2, n, R\rangle$, where: id is a unique identifier for a given mapping; E_1, E_2 are a set of entities, e.g., classes and properties of the first and the second ontology, respectively; R is* *a relation holding between the sets of entities E_1 and E_2; and n is a confidence measure in the [0;1] range that a relation R holds.*

The set of all mappings between two ontologies is a (declarative) specification of the semantic overlap between them [25].

2.2 Ontology Mapping: State-of-the-art

According to Noy et al three dimensions can be distinguished in ontology mapping (OM) research: a) mappings discovery; b) mappings representation and c) reasoning with mappings [23].

Most of the work in the area of OM is dedicated to mappings discovery, which is a challenging and complex problem. Mappings discovery considers the process of finding similarities between two ontologies by identifying the entities that represent the same semantics in both ontologies. In the past few years various tools and techniques were created to support the process of mappings discovery in ontologies. Tool and algorithm examples developed by academia include PROMPT [24], Chimaera [21], ONION [22], FCA-Merge [29], GLUE [5], QOM [7], Coma++ [1], Falcon [12], Rimom [18] and Asmov [14]. Interesting work was reported by Mascardi et al. that investigate the usage of upper ontologies, such as SUMO-OWL, OpenCyc, and DOLCE, for mappings discovery problem [20]. Although much research has been spent on developing tools and algorithms for the mappings discovery in ontologies, the process of mappings finding remains highly iterative and cannot be fully automated in majority of cases [2]. User participation is also required by many tools to conduct the process and to verify candidate mappings identified during the mapping discovery process. It is also interesting to notice that there is no "standard" or adopted by majority tool for this purpose [8].

Another important issue within the ontology mapping research is the application of mappings, i.e. what are the mappings applied for, what kind of reasoning is involved? [23] This question must be considered even before choosing the representation mechanism for mappings, because the mapping representation will depend on the specific application [26].

Most of the applications described in the literature are intended to address the problem of semantic heterogeneity. Following applications are addressed most often by the ontology mapping community:

- **Ontology Alignment** deals with the task of finding the similarities or relations between two different ontologies. The main aim is to bring the ontologies into mutual agreement [15].
- **Ontology Articulation** refers to the process of construction of the intermediate ontology with the explicitly identified relations between the initial ontologies and the intermediate ontology [9].
- **Ontology Merging** and **Ontology Integration** both refer to the process of creating a new ontology from several ontologies, in such a way that new ontology will unify and/or replace original ontologies [9, 15]. However a

slight difference can be identified between these terms: while ontology merging focuses on combining information from different ontologies in the same domain, ontology integration deals with ontologies that can originate from similar or even different domain (depending on the task at hand). Therefore, the domain of discourse of the resulting ontology in case of ontology integration often is more general than in the initial ontologies [9].

- **Ontology Mediation** refers to the process of reconciling differences between heterogeneous ontologies in order to achieve semantic interoperability between different services or applications [16].
- **Ontology Transformation** relates to the process of changing the semantics of an ontology (most often only slightly; the representation may also change) in order to adjust it for purposes other than original one [15].
- **Ontology Translation** refers to the task of changing the representational format of an ontology, while semantics is preserved. Therefore it mostly concerns data translation, but may also relate to syntax, e.g., while performing the translation from RDF(S)[5] into OWL [15, 16].
- **Ontology Versioning** is a process of managing the ontology development by creating different versions of it [16]. Very close stays the notion of **Ontology Evolution** that refers to the process of modifying an ontology because of the change in the domain of conceptualization [9]. The applications of ontology versioning and ontology evolution are highly interlinked in the literature, the first one is often considered to be a stronger version of a second one.

Another dimension of the ontology mapping research, which will be the main focus of this paper, is the representation of mappings. Noy at al. distinguish three main ways on how mappings can be represented: a) as first-order logic axioms serving as semantic bridges between ontologies; b) using views that link global ontology to local ontologies; and c) mappings themselves are instances in a mappings ontology [23]. An approach exploiting the Semantic Bridges between entities of two ontologies was presented by Maedche et al. In this approach each Semantic Bridge is a rule according to which a specific set of instances from the source ontology can be translated into the instances in the target ontology [19]. A language to specify mappings between ontologies and requirements for such language are described by Scharffe and Jos de Bruijn in [25]. The general problem is that there are many systems developed by different vendors, which represent mappings in their own formats, different from each other. A standard language for this purpose would highly facilitate the reusability and interoperability of the results of such systems, however there is no such single standard established yet [25]. Up to date there is also a lack of work investigating the abilities and appropriateness of current technologies for mappings representation in different application scenarios. We believe that this problem has high practical importance as the specific application will influence the choice of optimal mappings representation technique for the task at hand [26]. Focused on the data integration and data transformation applications, this paper provides a state of the

[5] `http://www.w3.org/TR/rdf-schema/`

art on existing technologies for mappings representation between ontologies, and evaluates their support for a set of essential mappings categories.

3 Use Case Scenario

In large-scale automation systems engineering projects, e.g., hydro power plants engineering, different participants, coming from various engineering disciplines, e.g. electrical engineering and software engineering, need to cooperate effectively and efficiently in order to deliver the end-product in time. The challenge here is that participants use different terminologies and tools with heterogeneous data formats, which are well-suited to support their specific discipline but are not adapted to cooperate with other engineering disciplines, their tools and data representations. As a running example for the paper we present a data integration use case scenario from an industry partner, a hydro power plant systems integrator.

Figure 1 illustrates a data integration scenario, in which ontologies are applied to integrate data from all disciplines involved in the project engineering. The data model of each discipline is represented by its local ontology. The common data model, which includes only concepts that are relevant to at least two disciplines on the project level (so called common concepts), is built on the top of local ontologies. For simplicity reasons we limit the set of concepts and attributes shown to the minimal set necessary to illustrate all the identified mappings categories introduced further in the paper.

On the top of the figure 1, the local ontologies representing data models from three different domains (software engineering (SE), mechanical engineering (ME) and project management (PM)) are shown. From the SE data model, the concept *PLC* describes the controllers' behavior of the devices. From the ME data model, the concept *PhysicalComponent* describes the components of the plant, and the concept *Connection* captures direct connections between the devices. From the PM data model, the concept *Project* describes the important characteristics of the project under development.

The figure 1 also shows relations between each local model and the common model represented by mappings. Different mapping categories are identified as M1-M7. These mapping categories are discussed in detail in the next section.

4 Mappings description

In the following we will describe various types of mappings that can arise in the context of real-world applications, especially when it comes to industry applications, but are not easy to express by means of pure RDF or OWL.

4.1 Value processing (M1-M4)

A widespread case is that a relation between entities in two ontology is not "exactly-the-same", but can be represented by some function that takes a value

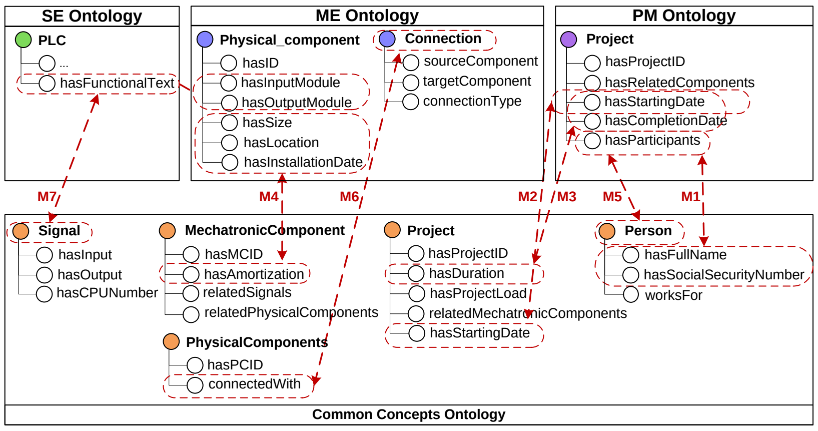

Fig. 1. Data integration use case - Mappings categories: M1 - string processing; M2 - data type transformation; M3 - math functions; M4 - user-defined functions; M5 - structural differences → granularity; M6 - structural differences → schematic differences; M7 - conditional mappings. Mappings M1-M2 could be also considered as bidirectional mappings (M8).

of a property in source ontology as an input and returns a value of a property in the target ontology as an output, i.e. a certain processing in needed to map the entities. The complexity of this processing varies strongly from simple string operation to sophisticated mathematical transformations. Below several types of suchlike mappings are described.

String processing (M1). All employers that work in a specific project are represented by attribute *hasParticipants* of the *Project* concept. The value of this attribute is a list of strings, each string contains information about employees full name and social security number in the following format: "name/surname/social security number". In the common model each employee is represented by common concept *Person* with a full name and social security number as its attributes. Thus, to transform data from the PM local ontology into common concepts ontology each string representing a certain employee must be split into 3 parts, which then will be used as values for the *hasFullName* and the *hasSocialSecurityNumber* attributes.

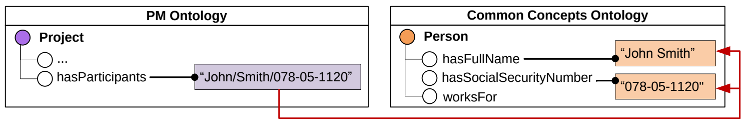

Fig. 2. Mappings category M1: string processing.

Data type transformation (M2). It can happen that in a proprietary data model the data type of a certain concept was not modeled in an optimal way, e.g. the date can be represented in the format of string, except of Date or DateType formats. For instance, all values of the *hasStartingDate* attribute in the PM ontology may be strings in the following format "DD/MM/YYYY". If the data type of corresponding attribute in the common ontology is *Date* then the data type transformation mapping must be defined between these 2 attributes. We consider two types of data type transformations: xsd data type transformations (i.e. "casting") and transforming values into other values.

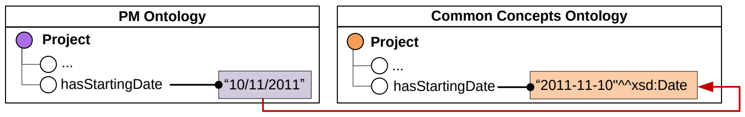

Fig. 3. Mappings category M2: data type transformation.

Math functions (M3). Here the relation between the entities in two ontologies represents some mathematical or physical formula. A sample of this mappings complexity type is the mapping between the *hasStartingDate* and *hasEndingDate* attributes of *Project* concept in PM ontology and the *hasDuration* attribute of the CC ontology. Corresponding formula will present a simple subtraction of ending date and starting date in order to obtain the value of project duration.

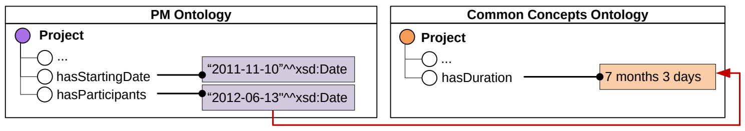

Fig. 4. Mappings category M3: math functions.

User-defined functions (M4). The common concept *MechatronicComponent* captures information regarding complex composite components, which consists of many physical components and basically can represent a plant itself. The anticipated amortization value can be an important characteristic on the project level. The exact value will depend on the specific location and size of the mechatronic component and also on its installation date.

4.2 Structural differences (M5-M6)

Granularity (M5). As an example of mapping that relates concepts on different granularity level, we can consider the mapping between the *hasParticipants*

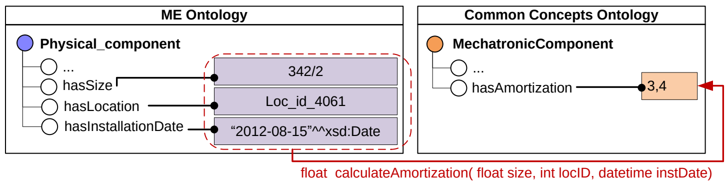

Fig. 5. Mappings category M4: user-defined functions.

attribute of the *Project* concept in PM ontology and the common concept *Person*. In the local PM ontology each employee that works for a specific project is represented by a string value of the hasParticipants attribute, while in the common model the concept *Person* serves the same purpose.

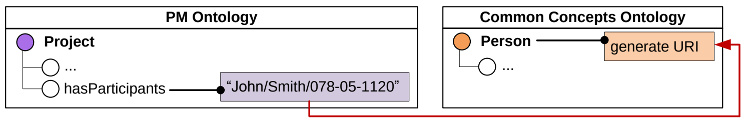

Fig. 6. Mappings category M5: structural differences: granularity.

Schematic differences (M6). This type of mappings can arise, for an instance, when two ontologies on a similar domain were created separately by different knowledge engineers and thus, the same semantics was modeled very differently. As an example of such mappings we will consider the mapping denoted as M6 in the described above use case scenario. The connection between physical devices is represented as a concept *Connection* with the *sourceComponent* and *targetComponent* as its attributes, while in the CC ontology the same semantics is expressed with the *connectedWith* attribute of the *PhysicalComponent* concept. In other words, the *Connection* of the ME ontology is a reification of the relation *connectedWith* of the common ontology.

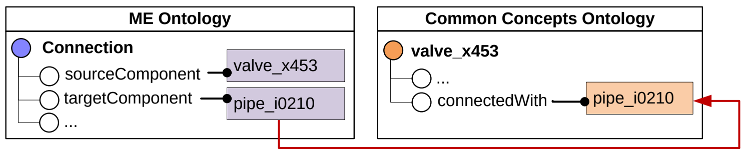

Fig. 7. Mappings category M6: structural differences: granularity: schematic differences.

4.3 Conditional mappings (M7)

The OWL ontology language provides some support for describing mappings between imported ontologies on the level of the schema (class-equivalence, property-equivalence, subClassOf) and at the level of instances (the “same as” relation). OWL, however, does not provide means for describing more complex mappings. For an instance, in the industry, “signal” is an important common concept in the automation systems engineering domain. It comprises information regarding the connection between physical devices and their controllers and corresponding PLC code. In the presented use case scenario value of the *hasFunctionalText* attribute of the *PLC* concept from the SE ontology is a string in the following format: ”CPU number/input module/output module”. Corresponding mapping states that for any pair of PLC and physical component, such that their values of *hasFunctionalText* and *hasInputModule* and *hasOutputModule* attributes have certain conformity an instance of *Signal* concept must be created in the CC ontology and appropriate values must be assigned to its *hasInput*, *hasOutput* and *hasCPUNumber* attributes. Even though the concepts of *PLC*, *Physical_component* and *Signal* are related, they are not the same concept. One of the mapping requirements we wish to examine is thus to what extent technologies allow for the transformation of descriptions of one type of thing into descriptions of another type.

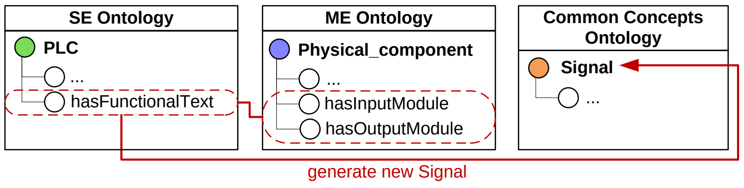

Fig. 8. Mappings category M7: conditional mappings.

4.4 Bidirectional mappings (M8)

An important characteristic of mappings is that they are directional [10], i.e. usually they are specified from a source ontology to a target ontology and the knowledge/data flow can’t occur in the opposite direction. However, for some applications, such as for an instance data transformation, it could be beneficial to have the opportunity to define bidirectional mappings between the entities of different ontologies. It would help to reduce total amount of mappings thus facilitating their maintenance.

5 Mechanisms for Representing Mappings

This section provides a description of technologies that can be applied to create mappings between ontologies. Some of the considered technologies allow mappings between RDF[6] graphs. However, RDF is merely the data model in which ontologies and instances respectively created and described with RDF(S) or OWL are stored.

Datalog Datalog is a declarative logic programming language, which is becoming increasingly popular for data integration tasks [13]. In Datalog, some mappings can be implemented as clauses:

```
area(X,Area) :- width(X,Width), height(X,Height), multiply(X,Y,Area).
```

Allowing one to retrieve all areas of things with a width and a height.

Web Ontology Language. The Web Ontology Language (OWL)[7], of which OWL 2.0 [8] became recently a W3C recommendation, envisaged an ontology language that should allow users to provide an explicit, formal conceptualization of a domain of discourse. The motivations of the development of OWL were: a well-defined syntax, with formal semantics and efficient reasoning support. OWL DL, however, permits efficient reasoning support and is the most expressive decidable OWL sublanguage.
Note that OWL itself does not either support the representation of mappings between ontologies or provide means for translating instances from one ontology into another. OWL reasoners allow for inferring additional triples from an OWL file, and thus could thus be used in combination of SPARQL CONSTRUCT queries to create a "reasoner-enabled" mapping.

Using rules. Where OWL and DL allow for classifying instances under concepts based on statements describing these concepts by inferring additional triples, (business) rules can be declared to infer additional triples that are not handled by OWL reasoners. Below are some examples of rule languages and engines:

Jena Rules. Apache Jena[9] is a Java framework for building Semantic Web applications and includes a rule-based inference engine for reasoning with RDF and OWL data sources. These rules are in terms of the source RDF file and will generate new triples. One problem with Jena rules is its implementation; some of the built-in primitives are not completely implemented. It is for an instance possible to compute the sum $?c$, if $?a$ and $?b$ in $sum(?a, ?b, ?c)$ are bound, but one can not obtain $?b$, if $?a$ and $?c$ are bound. Jena provides a Datalog implementation to reason with these rules.

SPARQL CONSTRUCT. The CONSTRUCT query form returns an RDF graph created with a template for generating RDF triples based on the

[6] `http://www.w3.org/TR/rdf-primer/`
[7] `http://www.w3.org/TR/owl-ref/`
[8] `http://www.w3.org/TR/owl2-overview/`
[9] `http://jena.apache.org/`

results of matching the graph pattern of the query. To use this construct, one thus needs to specify how patterns in one RDF graph are translated into another graph. The outcome of a SPARQL CONSTRUCT query depends on the reasoner and rule engine used. A SPARQL endpoint not backed by an OWL reasoner will only do simple graph matching for returning triples. A software agent that needs to compute these inferences will therefore have to consume all the necessary triples and perform this computation itself. The same holds for inferring additional information via business rules. SPARQL CONSTRUCT, however, is *not* a rule language and "merely" allows one to make a transformation from one graph match to another graph; i.e., a one-step transformation.

SPARQL Inference Notation (SPIN). The SPARQL Inference Notation (SPIN) [17] is currently submitted to W3C and provides means to link class definitions with SPARQL – the W3C standard for querying RDF – queries by describing the constraints and rules in these queries. These constraints and rules are in terms of SPARQL ASK and CONSTRUCT queries and thus provide means for translating instances of one RDF source into a different set of triples. This implies that the source and target data formats can be translated from and to RDF.

The Semantic Web Rule Language, (SWRL) is a W3C recommendation for a Semantic Web rules language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog). SWRL is based on OWL DL: all rules are expressed in terms of OWL concepts (classes, properties, individuals, literals, etc.). Rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. It is important to notice that SWRL rules are not means for creating mappings, but to infer additional information from a knowledge base. Therefore, they should be used in combination with e.g. SPARQL CONSTRUCT queries to create a target RDF graph out of a source graph.

6 Technologies Evaluation over Required Mappings Categories

In this section we provide an examination how well above mentioned technologies cover the examples in Fig. 1. This comparison is depicted in Table 1. Where necessary, we provide some extra information on certain caveats of the technologies. Datalog is implicitly included in the table below as Jena is a Datalog implementation. An example of the implementation of a particular mapping in the technologies considered is shown in Appendix A.

String processing (M1). The built-in string functions for Jena-Rules are rather limited, there exists for instance no function for obtaining substrings or string

Table 1. Evaluation of Technologies for Different Mapping Types.

Legend: Y = yes P = partial N = no * = depends on provider	**String**	**Data Type**	**Math**	**User-defined**	**Granularity**	**Schematic**	**Conditional**	**Bi-directional**	
	M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**Standard**
Jena-Rules	P	P	P	*	Y	Y	Y	P	-
SPIN	Y	Y	Y	Y*	Y	Y	Y	P	W3C Sub.
SWRL	Y	Y	Y	*	Y	Y	Y	P	W3C Rec.
SPARQL CONSTRUCT	P	Y	P	*	Y	Y	Y	N	W3C Rec.

replacement. SPARQL supports some additional basic string functions. We therefore consider the string processing coverage of Jena-Rules and SPARQL partial. SWRL and SPIN offer additional string functions. Note however, that some frameworks allows one to provide custom functions (cfr. M4). When one uses Jena to query RDF, for example, missing string functions can be added by using both Jena Rules and SPARQL.

Data type transformation (M2). There are two aspects to be analyzed: xsd data type transformations (i.e. "casting") and transforming values into other values. SPARQL 1.1 allows one to cast the value of a variable to another xsd data type. As SPIN allows one to create rules based on SPARQL queries, this is also covered by SPIN. The Jena-Rules and SWRL data type transformations are limited. For instance, Jena-Rules has a built-in primitive for creating a URI out of a set list of strings. However, SWRL and Jena-Rules are used to infer additional triples to query, the SPARQL queries built on top of these rules can do the casting. As for the second type of transformation: the combined limitations of M1 to M4 will hold.

Math functions (M3). All technologies support arithmetic, albeit Jena-Rules implementation has some limitations as mentioned in the previous Section. SWRL and SPIN even provide some functions beyond (e.g., *sin*, *cos*, etc.). These additional functions are not defined in the SPARQL specification and their existence can depend on the provider. Again, when necessary, some function can be provided as a user-defined function (cfr. M4).

User-defined functions (M4). The implementation of user-defined functions is interesting to look at. Some simple functions can be built via rules (i.e., declaratively), others might need to be implemented or already exist (e.g., legacy code that can be wrapped or procedurally). Defining functions can only be done in SPIN. For Jena Rules, SWRL and SPARQL, however, one can include custom functions if the software allows so. Jena, for instance, allows one to create custom functions in java which can be imported via a special namespace. Jena thus covers the inclusion of custom functions – albeit via code, and not in a separate knowledge artifact. This is why we denoted this possibility in the table with an asterisk. Since SPIN uses the Jena library, this possibility is also offered using that technology.

Granularity (M5) and Schematic differences (M6) tackled the problem of structural differences, and more particularly the different granularity and schematic differences respectively. All technologies provide means for transforming information into different representations. The limits actually depend on the specific requirements of a domain. In the presented use case scenario, one could use the social security number to create a URI for that person. Otherwise, blank nodes have to be introduced. The application domain of the knowledge bases could also have business rules that tell us how to construct URIs; which can be accomplished with aforementioned mapping techniques.

Conditional mappings (M7). Unlike OWL, which only allows relations between classes and properties via class- and property relations, both rules and SPARQL CONSTRUCT allow the transformation of descriptions of one type of concept into descriptions of other types of (related) concepts.

Bidirectional mappings (M8). Ideally, mappings are bidirectional. Meaning that the transformation from one model to the other and vice versa would be maintained in one artifact. However, if a mapping in one direction requires processing, the inverse can only be guaranteed if the process is a bijective function. Take, for example, the mapping from one entity containing with a height and width as an attribute to another entity with a surface. One can easily compute the surface by multiplying the height and width, but not vice verse. In fact, this mapping would not even be possible if no sufficient additional information is provided. None of the studied technologies allow for bidirectional rules. However, we can examine the problem from a different angle and see to what extent we can “simulate” bidirectionality by storing both mappings in one artifact. When possible rules from one ontology to another and vice verse can be declared with Jena-Rules, SWRL and SPIN. Moreover, that knowledge can even be stored in a separate artifact (i.e., file) as to not clutter the ontologies. This is not possible with SPARQL CONSTRUCT queries, as a SPARQL CONSTRUCT queries allows one to populate a graph based on the result set of a query.

The selection of one or other technology depends on the application requirements. For our running use case scenario, SPIN and SPARQL are the most suitable candidates to represent the mappings. On the one hand side we have SPARQL CONSTRUCT queries, allowing one to have step-wise transformation from one model to the other. The advantage is that SPARQL is a W3C recommendation. As SPARQL has been around for a while, different software solutions supporting SPARQL exist, of which some allow one to define their own functions (e.g., Jena). The downside is that mappings that depend on other mappings have to be executed in a certain order and are actually not a part of the knowledge base. This has been solved by SPIN, in which such rules even in terms of SPARQL CONSTRUCT queries are part of the knowledge base. SPIN even has a notion of function that can be reused in different parts. The downside of

SPIN is that it is still in consideration by W3C. An implementation of SPIN is available from TopBraid[10] that depends on Jena. Given the fact that it is more desirable to have the mappings as part of the knowledge base, SPIN proves to be a more desirable candidate. This will bring us to one of the future directions mentioned in the next section: "how can we support a community of stakeholders in collaboratively describing the different mappings".

7 Conclusion and Future Work

In this paper we examined available technologies that support mappings representation between ontologies regarding the ability to represent complex mappings required in the data integration scenario. We presented the set of mappings categories that were identified based on requirements for the data integration projects of an industry partner. We also evaluate existing technologies for mappings representation regarding the support provided for introduced mappings categories and discussed which technology would fit the best for the described use case scenario. Based on the results of the performed evaluation we can conclude that in the introduced use case scenario the SPARQL Inference Notation (SPIN) would be the optimal choice to define mappings between the ontologies.

We consider the following research directions for the future work: a) Investigation on how different applications scenarios, e.g. query rewriting, will influence the choice of optimal mappings representation technology; b) Investigation of the problem of mappings representation between ontologies described/defined in different languages and/or using different representational mechanisms, e.g. OWL ontology and F-Logic ontology [**?**]. Brockmans et al., for instance, provided a formalism-independent specification for ontology mappings [3] by using the Meta-Object Facility[11] for relating UML models with OWL DL ontologies.; c) Investigation of the problem of translation between mappings expressed using different representational techniques/mechanisms; d) Evaluation of existing software systems allowing mappings definition between different ontologies, additionally to technologies.

Acknowledgements

This work has been supported by the Christian Doppler Forschungsgesellschaft and by the Erasmus Mundus Action 2 Programme of the European Union.

References

1. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. pp. 906–908. ACM (2005)

[10] `http://www.topquadrant.com/`
[11] `http://www.omg.org/mof/`

2. Bernstein, P.A., Melnik, S.: Model management 2.0: manipulating richer mappings. In: Proceedings of the 2007 ACM SIGMOD international conference on Management of data. pp. 1–12. ACM (2007)
3. Brockmans, S., Haase, P., Stuckenschmidt, H.: Formalism-independent specification of ontology mappings - a metamodeling approach. In: Meersman, R., Tari, Z. (eds.) OTM Conferences (1). Lecture Notes in Computer Science, vol. 4275, pp. 901–908. Springer (2006)
4. Debruyne, C., Meersman, R.: Gospl: A method and tool for fact-oriented hybrid ontology engineering. In: Morzy, T., Härder, T., Wrembel, R. (eds.) ADBIS. Lecture Notes in Computer Science, vol. 7503, pp. 153–166. Springer (2012)
5. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.: Learning to match ontologies on the semantic web. The VLDB JournalThe International Journal on Very Large Data Bases 12(4), 303–319 (2003)
6. Ehrig, M.: Ontology alignment: bridging the semantic gap, vol. 4. Springer Science+ Business Media (2007)
7. Ehrig, M., Staab, S.: Qom–quick ontology mapping. In: The Semantic Web–ISWC 2004, pp. 683–697. Springer (2004)
8. Falconer, S.M., Noy, N.F., Storey, M.A.: Ontology mapping-a user survey. In: Proceedings of the Workshop on Ontology Matching (OM2007) at ISWC/ASWC2007, Busan, South Korea. pp. 113–125 (2007)
9. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. The Knowledge Engineering Review 23(2), 117–152 (2008)
10. Ghidini, C., Serafini, L., Tessaris, S.: On relating heterogeneous elements from different ontologies. In: Modeling and Using Context, pp. 234–247. Springer (2007)
11. Haase, P., Motik, B.: A mapping system for the integration of owl-dl ontologies. In: Proceedings of the first international workshop on Interoperability of heterogeneous information systems. pp. 9–16. ACM (2005)
12. Hu, W., Qu, Y.: Falcon-ao: A practical ontology matching system. Web Semantics: Science, Services and Agents on the World Wide Web 6(3), 237–239 (2008)
13. Huang, S.S., Green, T.J., Loo, B.T.: Datalog and emerging applications: an interactive tutorial. In: Sellis, T.K., Miller, R.J., Kementsietsidis, A., Velegrakis, Y. (eds.) SIGMOD Conference. pp. 1213–1216. ACM (2011)
14. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology matching with semantic verification. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 235–251 (2009)
15. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In: IJCAI-2001 Workshop on ontologies and information sharing. pp. 53–62. Citeseer (2001)
16. Klein, M., Fensel, D.: Ontology versioning on the semantic web. In: Proceedings of the International Semantic Web Working Symposium (SWWS). pp. 75–91 (2001)
17. Knublauch, H., Handler, J.A., Idehen, K.: SPIN - Overview and Motivation. W3C Member Submission, W3C (Feburary 2011), `http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/`
18. Li, J., Tang, J., Li, Y., Luo, Q.: Rimom: A dynamic multistrategy ontology alignment framework. Knowledge and Data Engineering, IEEE Transactions on 21(8), 1218–1232 (2009)
19. Maedche, A., Motik, B., Silva, N., Volz, R.: Mafraa mapping framework for distributed ontologies. In: Knowledge engineering and knowledge management: ontologies and the semantic web, pp. 235–250. Springer (2002)

20. Mascardi, V., Locoro, A., Rosso, P.: Automatic ontology matching via upper ontologies: A systematic evaluation. Knowledge and Data Engineering, IEEE Transactions on 22(5), 609–623 (2010)
21. McGuinness, D.L., Fikes, R., Rice, J., Wilder, S.: The chimaera ontology environment. In: Proceedings of the National Conference on Artificial Intelligence. pp. 1123–1124. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2000)
22. Mitra, P., Wiederhold, G.: Resolving terminological heterogeneity in ontologies. In: Proceedings of the ECAI workshop on Ontologies and Semantic Interoperability (2002)
23. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record 33(4), 65–70 (2004)
24. Noy, N.F., Musen, M.A.: The prompt suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies 59(6), 983–1024 (2003)
25. Scharffe, F., de Bruijn, J.: A language to specify mappings between ontologies. In: Proc. of the Internet Based Systems IEEE Conference (SITIS05). Citeseer (2005)
26. Scharffe, F., Fensel, D.: Correspondence patterns for ontology alignment. In: Knowledge Engineering: Practice and Patterns, pp. 83–92. Springer (2008)
27. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. In: Meersman, R., Tari, Z. (eds.) OTM Conferences (2). Lecture Notes in Computer Science, vol. 5332, pp. 1164–1182. Springer (2008)
28. Stuckenschmidt, H., Uschold, M.: Representation of semantic mappings. In: Kalfoglou, Y., Schorlemmer, W.M., Sheth, A.P., Staab, S., Uschold, M. (eds.) Semantic Interoperability and Integration. Dagstuhl Seminar Proceedings, vol. 04391. IBFI, Schloss Dagstuhl, Germany (2005)
29. Stumme, G., Maedche, A.: Fca-merge: Bottom-up merging of ontologies. In: International joint conference on artificial intelligence. vol. 17, pp. 225–234. LAWRENCE ERLBAUM ASSOCIATES LTD (2001)
30. Zhdanova, A.V., Shvaiko, P.: Community-driven ontology matching. In: The Semantic Web: Research and Applications, pp. 34–49. Springer (2006)

A Mappings in the Different Technologies

In this section, we demonstrate the implementation of one particular mapping (M3) in each of aforementioned technologies.

- Jena-Rules

```
[rule1: (?x <http://.../PM.owl#hasParticipants> ?p)
        regex(?p, '(.*)/(.*)/(.*)', ?fn, ?ln, ?sn)
        uriConcat('http://.../persons/', ?sn, ?uri)
        -> (?uri a <http://.../CC.owl#Person>)]
```

- SWRL. The following snippet is the stored SWRL rule created with Protégé. Note, however, that Protégé does not yet support SWRL built-ins. A more human-readable presentation of that rule is depicted in the XML comment above the snippet. This is how the rule was entered using the editor.

```
<!-- hasParticipants(?x, ?y),
     stringConcat("http://.../persons/", ?a, ?uri),
     substringAfter(?y, "/", ?z),
     substringAfter(?z, "/", ?a) -> Person(?uri). -->
<DLSafeRule>
  <Body>
    <DataPropertyAtom>
      <DataProperty abbreviatedIRI="pm:hasParticipants"/>
      <Variable IRI="urn:swrl#x"/>
      <Variable IRI="urn:swrl#y"/>
    </DataPropertyAtom>
    <BuiltInAtom abbreviatedIRI="swrlb:stringConcat">
      <Literal datatypeIRI="&rdf;PlainLiteral">http://.../persons/</Literal>
      <Variable IRI="urn:swrl#a"/>
      <Variable IRI="urn:swrl#uri"/>
    </BuiltInAtom>
    <BuiltInAtom abbreviatedIRI="swrlb:substringAfter">
      <Variable IRI="urn:swrl#y"/>
      <Literal datatypeIRI="&rdf;PlainLiteral">/</Literal>
      <Variable IRI="urn:swrl#z"/>
    </BuiltInAtom>
    <BuiltInAtom abbreviatedIRI="swrlb:substringAfter">
      <Variable IRI="urn:swrl#z"/>
      <Literal datatypeIRI="&rdf;PlainLiteral">/</Literal>
      <Variable IRI="urn:wrl#a"/>
    </BuiltInAtom>
  </Body>
  <Head>
    <ClassAtom>
      <Class IRI="http://.../Person"/>
      <Variable IRI="urn:swrl#uri"/>
    </ClassAtom>
  </Head>
</DLSafeRule>
```

– SPIN

```
@prefix sp:   <http://spinrdf.org/sp#> .
@prefix spin: <http://spinrdf.org/spin#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
<http://.../PM.owl#Project>
  spin:rule
    [ a sp:Construct ;
      sp:text """ PREFIX  o2:   <http://.../CC.owl#>
                  PREFIX  o1:   <http://.../PM.owl#>
                  CONSTRUCT { ?uri rdf:type o2:Person .}
                  WHERE { ?this rdf:type o1:Project .
                          ?this o1:hasParticipants ?p
                          BIND(strafter(?p, "/") AS ?rest)
                          BIND(strafter(?rest, "/") AS ?sn)
                          BIND(uri(concat("http://.../persons/", ?sn)) AS ?uri) } """ ] .
```

– SPARQL CONSTRUCT. The rule below states that when one encounters a project, one can infer some additional properties from that project by manipulating its properties. In SPIN, `?this` is reserved to refer the instance of that class being manipulated.

```
PREFIX  o2:   <http://.../CC.owl#>
PREFIX  o1:   <http://.../PM.owl#>
CONSTRUCT { ?uri <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> o2:Person .}
WHERE { ?x <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> o1:Project .
        ?x o1:hasParticipants ?p
        BIND(strafter(?p, "/") AS ?rest)
        BIND(strafter(?rest, "/") AS ?sn)
        BIND(uri(concat("http://.../persons/", ?sn)) AS ?uri) }
```