Formal Model for Ontology Mapping Creation*

Adrian Mocan, Emilia Cimpian, Mick Kerrigan

Digital Enterprise Research Institute, University of Innsbruck, Austria
{adrian.mocan, emilia.cimpian, mick.kerrigan}@deri.org

Abstract. In a semantic environment data is described by ontologies and heterogeneity problems have to be solved at the ontological level. This means that alignments between ontologies have to be created, most probably during design-time, and used in various run-time processes. Such alignments describe a set of mappings between the source and target ontologies, where the mappings show how instance data from one ontology can be expressed in terms of another ontology. We propose a formal model for mapping creation. Starting from this model we explore how such a model maps onto a design-time graphical tool that can be used in creating alignments between ontologies. We also investigate how such a model helps in expressing the mappings in a logical language, based on the semantic relationships identified using the graphical tool.

1 Introduction

Ontology mapping is becoming a crucial aspect in solving heterogeneity problems between semantically described data. The benefits of using ontologies, especially in heterogenous environments where more than one ontology is used, can only be realized if this process is effective. The trend is to provide graphical tools capable of creating alignments during design-time in a (semi-)automatic manner [2,10,9]. These alignments consist of mapping rules, frequently described as *statements* in a logical language. One of the main challenges is to fully isolate the domain expert (who is indispensable if 100% accuracy is required) from the burdens of logics using a graphical tool, and in the same time to be able to create complex, complete and correct mappings between the ontologies.

It is absolutely necessary to formally describe the mapping creation process and to link it with the instruments available in a graphical tool and with a mapping representation formalism that can be used later during run-time. This allows the actions performed by the user to be captured in a meaningful way with respect to the visualized ontology structure and to associate the results of these actions (mappings) with concrete statements in a mapping language (mapping rules).

* Work funded by the European Commission under the projects ASG, DIP, enIRaF, InfraWebs, Knowledge Web, Musing, Salero, SEKT, Seemp, SemanticGOV, Super, SWING and TripCom; by Science Foundation Ireland under the DERI-Líon Grant No.SFI/02/CE1/I13; by the FFG (Österreichische Forschungsfrderungsgeselleschaft mbH) under the projects Grisino, RW^2, SemNetMan, SeNSE, TSC, OnTourism.

The document structure is as follows: the next section presents the context and motivation for the work. Section 3 introduces the model we propose expressed using First-Order Logic [4]. Section 4 describes how this model can be applied to WSMO [3] ontologies, while Section 5 presents the creation of mapping rules; the prototype that implements and applies the proposed formal model is described in Section 6. Following, related work and conclusions are presented.

2 Context and Motivation

The work described in this paper has been carried out in the Web Service Execution Environment (WSMX) working group, whose scope is to build a framework that enables discovery, selection, mediation, invocation and interoperation of Semantic Web Services [6]. Web Services are semantically described using ontologies, but as they are generally developed in isolation, heterogeneity problems appear between the underlying ontologies. Without resolving these problems the communication (data exchanged) between Web Services cannot take place. The data mediation process in WSMX includes two phases: a *design-time* and a *run-time* phase. The mismatches between the ontologies are resolved at design-time, while these findings are used at run-time to transform the data passing through the system. The run-time phase can be completely automated, while the design-time phase remains semi-automatic, requiring the inputs of a domain expert.

For the design-time a semi-automatic ontology mapping tool was developed that allows the user to create alignments between ontologies and to make these alignments available for the run-time process. There has been much research in the area of graphical mapping tools, e.g. [9,10], however we believe there are many challenges still to be addressed. In particular, our focus has been on defining strategies that hide the burden of logical languages, that are generally used to express ontology alignments, from the domain expert. The mapping process must remain simple to use but simultaneously allow the creation of complex mappings between two ontologies.

Table 1. Ontology Fragment

```
concept person
    name ofType xsd:string
    age ofType xsd:integer
    hasGender ofType gender
    hasChild ofType person
    marriedTo ofType person

concept gender
     value ofType xsd:string

instance male memberOf gender
     value hasValue "male"
instance female memberOf gender
     value hasValue "female"
```

As described in [8], we noticed that the graphical point of view adopted to visualize the source and target ontologies makes it easier to identify certain types of mappings. The ontology fragment in Table 1 can be visualized using different viewpoints by shifting the focus from one ontology element to another (see Table 2). We call such a viewpoint a *perspective* and argue that only by switching between combinations of these perspectives on the source and target

ontologies, can certain types of mappings be created using only one simple operation, *map*, combined with mechanisms for ontology traversal and contextualized visualization strategies.

A formal model that describes the general principles of the perspectives allows a better understanding of the human user actions in the graphical tool and of the effects of these actions on the ontology alignment (i.e. mapping rules) that is being created. This model defines the main principles that support the graphical instruments (e.g. perspectives) and how they fit with the underlying logical mechanism (e.g. decomposition, context updates). The same model is also used to describe how the inputs placed through these graphical instruments by the domain expert effects the generated mappings. Having this formal model as a link between the graphical elements and the mappings, defines precisely the process of hiding from the domain expert the complexity of the underlying logical languages; it also allows some of the mapping properties such as (in)completeness or (in)consistency to be reflected back into the graphical tool. Additionally, such a model allows experts to become more familiar with the tools and to create extensions that are more suited to capturing certain types of mismatches.

Table 2. *PartOf*, *InstanceOf* and *RelatedBy* Perspectives

• string
• integer
• person
  ⊢ *name* → string
  ⊢ *age* → integer
  ⊢ *hasGender* → gender
  ⊢ *hasChild* → person
  └ *marriedTo* → person
• gender
  └ *value* → string
	• string
	• integer
	• person
	  ⊢ *hasGender* → male:gender
	  └ *hasGender* → female:gender
	• gender
		• name
		  ⊢ *hasDomain* → person
		  └ *hasRange* → string
		• hasGender
		  ⊢ *hasDomain* → person
		  └ *hasRange* → gender
		• marriedTo
		  ⊢ *hasDomain* → person
		  └ *hasRange* → person
		...
		• value
		  ⊢ *hasDomain* → gender
		  └ *hasRange* → string

3 A Model for Mapping Creation

This section defines a model to be used in the creation of mappings between ontologies. The roles that appear in the graphical user interface, and which will be later associated with ontological entities, are defined here. First-Order Logic [4] is used as a formalism to represent this model.

3.1 Perspectives

In our approach the ontologies are presented to the user using *perspectives*. A perspective can be seen as a vertical projection of the ontology and it will be used by the domain expert to visualize and browse the ontologies and to define mappings. We can define several perspectives on an ontology as presented in Section 4, all of them characterized by a set of common elements.

Table 3. Types of items for the perspectives in Table 2

	PartOf	*InstanceOf*	*RelatedBy*
ci	person, gender	person	name, hasGender, marriedTo, value
pi	string, integer	string, integer, gender	-
di	name, age, hasChild, marriedTo	hasGender	hasDomain, hasRange
si	string, integer, gender, person	male, female	person, string, gender

We identify four types of such elements (items): *compound*, *primitive*, *description* and *successor*. We use the following unary relations to denote each of them, $ci(x)$ where x is a compound item, $pi(x)$ where x is a primitive item, $di(x)$ where x is a description item and $si(x)$ where x is a successor item. Both primitive and compound items represent first-class citizens of a perspective while description and successor items link the compound and the primitive items in a graph-based structure. In addition we define a set of general relationships between these items that hold for all perspectives:

- Each compound item is described by at least one description item:

$$\forall x.(ci(x) <=> \exists y.(di(y) \wedge describes(y, x))) \tag{1}$$

  where *describes* is a binary relation that holds between a compound item and one of its description items. The participants in this relation are always a compound item and a description item:

$$\forall x.\forall y.(describes(x, y) => ci(y) \wedge di(x)) \tag{2}$$

- Each description item points to at least one successor item:

$$\forall x.(di(x) <=> \exists y.(si(y) \wedge successor(y, x))) \tag{3}$$

  where *successor* is a binary relation that holds between a description item and one of its successor items. The participants in this relation are always a description item and a successor item:

$$\forall x.\forall y.(successor(x, y) => di(y) \wedge si(x)) \tag{4}$$

- The successor items are either primitive or compound items:

$$\forall x.(si(x) => pi(x) \vee ci(x))) \tag{5}$$

- The compound, primitive and description items are mutually exclusive for the same perspective:

$$\forall x.(\neg((ci(x) \wedge pi(x)) \vee (ci(x) \wedge di(x)) \vee (di(x) \wedge pi(x)))) \tag{6}$$

This is a set of minimal descriptions for our model, but by inference other useful consequences can be inferred. For example, note that sentences 1 and 6 imply that primitive items have no description items. Table 4 shows examples of relationships for the perspectives in Table 2.

Table 4. Relations between items in the perspectives depicted in Table 2

	successor(·, ·)	*describes(·, ·)*
PartOf	(string, name), (gender, hasGender) (person, marriedTo)	(name, person), (hasGender, person) (marriedTo, person)
InstanceOf	(male, hasGender), (female, hasGender)	(hasGender, person)
RelatedBy	(person, hasDomain), (string, hasRange) (gender, hasRange)	(hasDomain, name), (hasRange, name) (hasRange, hasGender)

As a consequence we can define a perspective as being a set $\phi = \{x_1, x_2, ..., x_n\}$ for which we have:

$$\forall x.\forall \phi.(member^1(x, \phi) => pi(x) \vee ci(x) \vee di(x)) \tag{7}$$

In addition, for any perspective the following sentences hold:

$$\forall x.\forall y.\forall \phi.(describes(y, x) => (member(x, \phi) <=> member(y, \phi))) \tag{8a}$$
$$\forall x.\forall y.\forall \phi.(successor(y, x) => (member(x, \phi) <=> member(y, \phi))) \tag{8b}$$

Sentences 8a and 8b together with 2 and 4 state that the description of a compound item appears in the perspective *iff* the compound item appears in the perspective as well. Similarly, a successor of a description item appears in a perspective *iff* the description item appears in the perspective too.

3.2 Contexts

Not all of the information modeled in the ontology is useful in all stages of the mapping process. The previous section shows that a perspective represents only a subset of an ontology, but we can go further and define the notion of *context*. A context is a subset of a perspective that contains only those ontological entities, from that perspective, relevant to a concrete operation. We can say that γ_ϕ is a context of the perspective ϕ if:

$$\forall x.(member(x, \gamma_\phi) => member(x, \phi)) \tag{9}$$

For a context from formulas 8a and 8b only 8a holds, such that:

$$\forall x.\forall y.\forall \gamma_\phi.(describes(y, x) => (member(x, \gamma_\phi) <=> member(y, \gamma_\phi))) \tag{10}$$

As a consequence we can say that all perspectives are contexts but not all contexts are perspectives.

A notion tightly related with contexts is the process of *decomposition*. A context can be created from another context (this operation is called *context update*) by applying decomposition on an item from a perspective or a context. Let $decomposition(x, \phi)$ be a binary function which has as value a new context obtained by decomposing x in respect with the context γ_ϕ. We can define the following axioms:

$$\forall x.\forall y.\forall \gamma_\phi.(member(x, \gamma_\phi) \wedge pi(x) => (member(y, decomposition(x, \gamma_\phi)) <=> member(y, \gamma_\phi))) \tag{11}$$

[1] *member* is a relationships expressing the membership of an element to a list.

$$\forall x.\forall y.\forall \gamma_\phi.(member(x,\gamma_\phi) \wedge ci(x) => (member(y, decomposition(x,\gamma_\phi)) <=> y = x \vee describes(y,x))) \quad (12)$$

$$\forall x.\forall y.\forall z.\forall \gamma_\phi.(member(x,\gamma_\phi) \wedge di(x) \wedge successor(z,x) \wedge (pi(z) \vee (ci(z) \wedge member(z,\gamma_\phi))) => (member(y, decomposition(x,\gamma_\phi)) <=> member(y,\gamma_\phi))) \quad (13)$$

$$\forall x.\forall y.\forall z.\forall \gamma_\phi.(member(x,\gamma_\phi) \wedge di(x) \wedge successor(z,x) \wedge ci(z) \wedge \neg(member(z,\gamma_\phi)) => (member(y, decomposition(x,\gamma_\phi)) <=> member(y, decomposition(z,\phi)))) \quad (14)$$

Intuitively, formula 11 specifies that the decomposition of a primitive concept does not update the current context (the context remains unchanged). Also, decomposition applied on a description item that has a primitive successor (formula 13) leaves the current context unchanged. The same formula also does not allow the decomposition of those description items that have as successor a compound item already contained by the current context (recursive structures).

Table 5 presents some examples of decompositions and context updates: each column shows how the context changes by decomposing any of the marked items in the top row. The decomposition can be applied simultaneously on multiple items, and the result of decomposing each item is contributing to the new context. Note as described over for column 1 no change occurs as all of the marked items cannot trigger decomposition conforming to formulae 11 and 13.

Table 5. Decomposition and context updates

Original Context		
• **string** • **integer** • person ⊢ *name* → string ⊢ *age* → integer ⊢ *hasGender* → gender ⊢ *hasChild* → person └ *marriedTo* → person • gender └ *value* → string	• string • integer • **person** ⊢ *name* → string ⊢ *age* → integer ⊢ *hasGender* → gender ⊢ *hasChild* → person └ *marriedTo* → person • gender └ *value* → string	• string • integer • person ⊢ *name* → string ⊢ *age* → integer ⊢ **hasGender** → gender ⊢ *hasChild* → person └ *marriedTo* → person • **gender** └ *value* → string
New Context		
• string • integer • person ⊢ *name* → string ⊢ *age* → integer ⊢ *hasGender* → gender ⊢ *hasChild* → person └ *marriedTo* → person • gender └ *value* → string	• person ⊢ *name* → string ⊢ *age* → integer ⊢ *hasGender* → gender ⊢ *hasChild* → person └ *marriedTo* → person	• gender └ *value* → string

3.3 Mappings

To create mappings between ontologies, a source and target perspective is used to represent the source and target ontologies. We refer to this approach as interactive mapping creation. It means that the mapping creation process relies upon

the domain expert, who has the role of choosing an item from the source perspective and one from the target perspective (or contexts) and explicitly marking them as mapped items. We call this action *map* and using this the domain expert states that there is a semantic relationship between the mapped items. Choosing the right pair of items to be mapped is not necessarily a manual task: a semi-automatic solution can offer suggestions that are eventually validated by the domain expert [8].

We define a *mapping context* as a quadruple $Mc =< \phi_S, \gamma_{\phi_S}, \phi_T, \gamma_{\phi_T} >$ where ϕ_S and ϕ_T are the source and target perspectives associated to the source and target ontologies. γ_{ϕ_S} and γ_{ϕ_T} are the current contexts derived out of the two perspectives ϕ_S and ϕ_T. Initially, $\gamma_{\phi_S} \equiv \phi_S$ and $\gamma_{\phi_T} \equiv \phi_T$.

We also define $map_{Mc}(x, y)$ the action of marking the two items x and y as being semantically related with respect to the mapping context Mc. Thus, we have the following axiom:

$$\begin{aligned} \forall x.\forall y.\forall \phi_S.\forall \phi_T.\forall \gamma_{\phi_S}.\forall \gamma_{\phi_T} map_{Mc}(x,y) \wedge Mc =< \phi_S, \gamma_{\phi_S}, \phi_T, \gamma_{\phi_T} > \wedge \\ ((ci(x) \vee pi(x)) \wedge (ci(y) \vee pi(y))) \vee (di(x) \wedge di(y)) \;\; => \\ member(x, \gamma_{\phi_S}) \wedge member(y, \gamma_{\phi_T}) \end{aligned} \tag{15}$$

Formula 15 defines the allowed types of mapping. Thus we can have mappings between primitive and/or compound items and between description items. As described in [8] the set of the allowed mappings can be extended or restricted by a particular, concrete perspective.

Each time a *map* action occurs the mapping context is updated; we denote the updates using: $Mc \twoheadrightarrow Mc^{'}$ meaning that at least one element of the quadruple defining Mc has changed and the new mapping context is $Mc^{'}$. The mapping context updates occur as defined in axiom 16:

$$\begin{aligned} \forall x.\forall y. map_{Mc}(x,y) \wedge Mc =< \phi_S, \gamma_{\phi_S}, \phi_T, \gamma_{\phi_T} > \; => \\ Mc' =< \phi_S, decomposition(x, \gamma_{\phi_S}), \phi_T, decompositin(y, \gamma_{\phi_T}) > \wedge Mc \twoheadrightarrow Mc' \end{aligned} \tag{16}$$

There are cases when Mc and Mc' are identical; such situations occur when the source and target context remain unchanged, e.g. when creating mappings between primitive items.

4 Grounding the Model to Ontologies

This section explores the way in which the model presented above can be applied to a real ontological model and how we can use it to define concrete perspectives that could be used to create meaningful mappings between ontologies. We first introduce the main aspects of WSMO ontologies and a mechanism to link these ontologies with our model and then we will present the three types of concrete perspectives we identified as being useful in the mapping process.

The Web Service Modeling Ontology (WSMO) defines the main aspects related to Semantic Web Services: Ontologies, Web Services, Goals and Mediators [3], from these only Ontologies are interesting in this work. We will focus only on concepts, attributes and instances in this paper, however we intend to address other ontological elements in the future. WSMO ontologies are expressed

using the Web Service Modeling Language (WSML) which is based on different logical formalisms namely, Description Logics, First-Order Logic and Logic Programming [5].

Table 1 presents an example of concepts and their attributes, and some instances of these concepts. The concept *person* is modeled as having 5 attributes, each of them having a type (i.e. a range) that is either another concept or a data type. For the concept *gender* there are two instances defined (i.e. *male* and *female*) that have attributes pointing to values of the corresponding types.

4.1 PartOf Perspective

The *PartOf* perspective is the most common perspective that can be used to display an ontology, focusing on the concepts, attributes and attributes' types hierarchies. To link this perspective with our model we define the unary relations $ci_{PartOf}(x)$, $pi_{PartOf}(x)$ and $di_{PartOf}(x)$ such that:

$$ci(x) \textbf{ iff } ci_{PartOf}(x) \qquad pi(x) \textbf{ iff } pi_{PartOf}(x) \qquad di(x) \textbf{ iff } di_{PartOf}(x) \tag{17}$$

$ci_{PartOf}(x)$, $pi_{PartOf}(x)$ and $di_{PartOf}(x)$ have to be defined in the logical language used to represent the ontologies to be aligned, in our case WSML[2] as can be seen in 18. In the *PartOf* perspective the role of compound items is taken by those concepts that have at least one attribute - we call them *compound concepts*. Naturally, the description items are in this case attributes, as stated in 19. Primitive items are data types or those concepts that have no attributes, as expressed by axiom 20 where x **subconceptOf** *true* holds iff x is a concept and *naf* stands for negation as failure. Finally we link the *describes* and *successor* relations with the WSML ontologies in 21. The ontology fragment presented in Table 1 can be visualized using the *PartOf* perspective as in Table 2.

$$\textbf{axiom } ci_{PartOf} \textbf{ definedBy } ci_{PartOf}(x) \textbf{ equivalent exists } ?y, ?z(?x[?y \textbf{ ofType } ?z]) \tag{18}$$

$$\textbf{axiom } di_{PartOf} \textbf{ definedBy } di_{PartOf}(y) \textbf{ equivalent exists } ?x, ?z(?x[?y \textbf{ ofType } ?z]) \tag{19}$$

$$\textbf{axiom } pi_{PartOf} \textbf{ definedBy } \; pi_{PartOf}(x) \textbf{ :- } ?x \textbf{ subconceptOf } true \text{ and } \textbf{naf } ci_{PartOf}(x) \tag{20}$$

$$describes(y, x) \wedge successor(z, y) \textbf{ iff } ?x[?y \textbf{ ofType } ?z] \tag{21}$$

4.2 InstanceOf Perspective

The *InstanceOf* perspective can be used to create conditional mappings based on predefined values and instances. To link this perspective with our model we define $ci_{InstanceOf}(x)$, $pi_{InstanceOf}(x)$ and $di_{InstanceOf}(y, w)$ such that:

$$ci(x) \textbf{ iff } ci_{InstanceOf}(x) \quad pi(x) \textbf{ iff } pi_{InstanceOf}(x) \quad di(< y, w >) \textbf{ iff } di_{InstanceOf}(y, w) \tag{22}$$

[2] In WSML $\alpha[\beta$ **ofType** $\gamma]$ is an atomic formulas called *molecule*; in here both α and γ identifies concepts while β identifies an attribute and '?' is used to denote variables. An example of a molecule for the ontology fragment in Table 1 is $person[name$ **ofType** $string]$

The description items are tuples $< y, w >$ where **y** is an attribute matching the above conditions and **w** is an instance member of y's type explicitly defined in the ontology or an anonymous id representing a potential instance of the y's type. In the same way as above, $ci_{InstanceOf}(x)$, $pi_{InstanceOf}(x)$ and $di_{InstanceOf}(x)$ are defined using WSML; also the *describes* and *successor* relations can be linked with the WSML ontologies in a similar manner as presented in the previous section. From space reasons, they are omitted from this paper. The fragment of ontology presented in Table 1 can be visualized using the *InstanceOf* perspective as in Table 2.

4.3 RelatedBy Perspective

The *RelatedBy* perspective focuses on the attributes of the ontology, and describes them from their domain and type perspective.

$$ci(x) \textbf{ iff } ci_{RelatedBy}(x) \quad pi(x) \textbf{ iff } pi_{RelatedBy}(x) \quad di(x) \textbf{ iff } di_{RelatedBy}(x) \tag{23}$$

In the same way as above, $ci_{RelatedBy}(x)$, $pi_{RelatedBy}(x)$ and $di_{RelatedBy}(x)$ are defined using WSML; also the *describes* and *successor* relations can be linked with the WSML ontologies in a similar manner as presented in Section 4.1. From space reasons, they are omitted from this paper. The fragment of ontology presented in Table 1 can be visualized using the *RelatedBy* perspective as in Table 2.

5 Linking the Model to a Mapping Language

In this section we specify the allowed mappings for each of the perspectives described in Section 4. We start from the following premise $map_{Mc}(x_S, y_T) \wedge Mc =< \phi_S, \gamma_{\phi_S}, \phi_T, \delta_{\phi_T} >$ which means that the elements x_S and y_T from the source and target ontology, respectively, are to be mapped in the mapping context Mc. In the following subsection we will discuss the situations that can occur for a pair of perspectives (due to space reasons we address only those cases when the source and target perspectives are of the same type). The types of mapping that can be created will be analyzed with respect to the Abstract Mapping Language proposed in [1], briefly described in 5.1.

5.1 Abstract Mapping Language

We chose to express the mappings in the abstract mapping language proposed in [1] because it does not commit to any existing ontology representation language. Later, a formal semantic has to be associated with it and to ground the mappings to a concrete language (such a grounding can be found in [8]). We provide only a brief listing of some of the abstract mapping language statements:

- *classMapping* - By using this statement, mappings between classes in the source and the target ontologies are specified. Such a statement can be conditioned by class conditions (*attributeValueConditions*, *attribuiteTypeConditions*, *attributeOccurenceConditions*).

– *attributeMapping* - Specifies mappings between attributes. Such statements usually appear together with classMappings and can be conditioned by attribute conditions (*valueConditions*, *typeConditions*).
– *classAttributeMapping* - It specifies mappings between a class and an attribute (or the other way around) and it can be conditioned by both class conditions and attribute conditions.
– *instanceMapping* - It states a mapping between two individuals, one from the source and the other from the target.

In the next sections we illustrate how these mapping language statements are generated during design time by using a particular combination of perspectives.

5.2 *PartOf* to *PartOf* Mappings

When using the *PartOf* perspective to create mappings for both the source and target ontologies we have the following allowed cases (derived from axiom 15):

– $pi_{PartOf}(x_S) \wedge pi_{PartOf}(x_T)$ In this case, the mapping will generate a *classMapping* statement in the mapping language and leaves the mapping context unchanged (axioms 11 and 16).
– $ci_{PartOf}(x_S) \wedge ci_{PartOf}(x_T)$ Generates a *classMapping* statement and updates the context for the source and target perspectives (axioms 12 and 16).
– $di_{PartOf}(x_S) \wedge di_{PartOf}(x_T)$ In this case $successor(y_S, x_S) \wedge successor(y_T, x_T)$ holds and we can distinguish the following situations:
  • $pi_{PartOf}(y_S) \wedge pi_{PartOf}(y_T)$ An *attributeMapping* is generated between x_S and x_T followed by a *classMapping* between y_S and y_T. Conforming to the axioms 13 and 16, the mapping context remains unchanged.
  • $ci_{PartOf}(y_S) \wedge ci_{PartOf}(y_T)$ An *attributeMapping* is generated having as participants x_S and x_T. The mapping context is updated conform to the axioms 13, 14 and 16.
  • $pi_{PartOf}(y_S) \wedge ci_{PartOf}(y_T)$ Generates a *classAttributeMapping* between z_S and the x_T, where $describes(x_S, z_S)$. The new mapping context keeps the source context unchanged while decomposing the target context over y_T.
  • $ci_{PartOf}(y_S) \wedge pi_{PartOf}(y_T)$ This case is symmetric with the one presented above and it generates a *classAttributeMapping* between x_S and the z_T where $describes(x_T, z_T)$.
– $ci_{PartOf}(x_S) \wedge pi_{PartOf}(x_T)$ It is not allowed for this combination of perspectives. To take an example, such a case would involve a mapping between $ci_{PartOf}(person)$ and $pi_{PartOf}(string)$ where $describes(hasName, person) \wedge successor(string, hasName)$, which does not have any semantic meaning. A correct solution would be a mapping between $ci_{PartOf}(person)$ and $ci_{PartOf}(u_T)$ such as $\exists v_T.(describes(v_T, u_T) \wedge successor(string, v_T)$.
– $pi_{PartOf}(x_S) \wedge ci_{PartOf}(x_T)$ The same explanation applies as above.

5.3 *InstanceOf* to *InstanceOf* Mappings

When using the *InstanceOf* perspectives we can create similar mappings to those created with the *PartOf* perspectives, the difference being that conditions are added to the mappings, and by this, the mappings hold only if the conditions are fulfilled. The mappings between two primitive items or between two compound items in the *InstanceOf* perspective are identical with the ones from the *PartOf* perspective. For the remaining cases we have:

- $di_{InstanceOf}(x_S, w_S) \wedge di_{InstanceOf}(x_T, w_T)$ In this case, we have $successor(< x_S, w_S >, y_S) \wedge successor(< x_T, w_T >, y_T)$ and we can distinguish the following situations:
  - $pi_{InstanceOf}(y_S) \wedge pi_{InstanceOf}(y_T)$ An *attributeMapping* is generated between x_S and x_T conditioned by two *attributeValueCondition*s imposing the presence of w_S and w_T in the mediated data. Also a *classMapping* between y_S and y_T is generated. Conforming to the axioms 13 and 16 the mapping context remains unchanged.
  - $ci_{InstanceOf}(y_S) \wedge ci_{InstanceOf}(y_T)$ An *attributeMapping* is generated having as participants x_S and x_T conditioned by two *typeConditions*. The mapping context is updated conforming to the axioms 13, 14 and 16.
  - $pi_{InstanceOf}(y_S) \wedge ci_{InstanceOf}(y_T)$ This case generates a *classAttributeMapping* between z_S and the x_T, where $describes(x_S, z_S)$. A *typeCondition* is added for x_T attribute. The new mapping context keeps the source context unchanged while decomposing the target context over y_T.
  - $ci_{InstanceOf}(y_S) \wedge pi_{InstanceOf}(y_T)$ This case is symmetric with the one presented above and it generates a *classAttributeMapping* between x_S and the z_T where $describes(x_T, z_T)$. A *typeCondition* is added for x_S.
- $di_{InstanceOf}(x_S, w_S) \wedge pi_{InstanceOf}(x_T)$ *InstanceOf* extends the set of allowed mappings as defined in 15. For z_S such that $describes(< x_S, w_S >, z_S)$, a *classMapping* between z_S and x_T is generated, conditioned by an *attributeValueCondition* on the attribute x_S and value w_S.
- $pi_{InstanceOf}(x_S) \wedge di_{InstanceOf}(x_T, w_T)$ Similar with the above case.
- $ci_{InstanceOf}(x_S) \wedge pi_{InstanceOf}(x_T)$ It is not directly allowed for this combination of perspectives, but the intended mapping can be created as described by previous case.
- $pi_{InstanceOf}(x_S) \wedge ci_{InstanceOf}(x_T)$ The same explanation applies as above.

5.4 *RelatedBy* to *RelatedBy* Mappings

In the *RelatedBy* perspective attributes are seen as root elements, having only two descriptions: their domain and their type. We identify the following cases:

- $pi_{RelatedBy}(x_S) \wedge pi_{RelatedBy}(x_T)$ This case does not appear as we do not have primitive items in the *RelatedBy* perspective.
- $ci_{RelatedBy}(x_S) \wedge ci_{RelatedBy}(x_T)$ The mapping will generate an *attributeMapping* statement in the mapping language having as participants x_S and x_T.
- $di_{RelatedBy}(x_S) \wedge di_{RelatedBy}(x_T)$ The source and the target perspectives are changed from *RelatedBy* to *PartOf* and the context is obtained by decomposing the perspectives over z_S and z_T, where $successor(z_S, x_S) \wedge successor(z_T, x_T)$

5.5 Mapping Examples

Table 6 shows examples of mappings in the abstract mapping language and how these mappings look like when grounded to WSML when mapping the person concept in the the source ontology with human (and man) in the target ontology. When evaluated, the WSML mapping rules will generate instances of *man* if the *gender* condition is met, or of *human* otherwise. The construct $mediated(X, C)$ represents the identifier of the newly created target instance, where X is the source instance that is transformed, and C is the target concept we map to.

Table 6. Decomposition and context updates

Abstract Mapping Language	Mapping Rules in WSML

Mapping(o1#persono2#man
  classMapping(*one-way*
  person man))
Mapping(o1#ageo2#age
  attributeMapping(one-way
  [(person)age=>integer]
  [(human)age=>integer]))
Mapping(o1#nameo2#name
  attributeMapping(one-way
  [(person)name => string]
  [(human)name => string]))
Mapping(o1#hasGendero2#man
  attributeClassMapping(one-way
  [(person)hasGender => gender] man))
  valueCondition(
  [(person)hasGender => gender] male)
	axiom mapping001 definedBy
	  mediated(X_1, o2#man) memberOf o2#man:-
	  X_1 memberOf o1#person.
	axiom mapping001 definedBy
	  mediated(X_2, o2#human) memberOf o2#human:-
	  X_2 memberOf o1#person.
	axiom mapping005 definedBy
	  mediated(X_5, o2#human)[o2#age hasValue Y_6]:-
	  X_5[o1#age hasValue Y_6]:o1#person.
	axiom mapping006 definedBy
	  mediated(X_7, o2#human)[o2#name hasValue Y_8]:-
	  X_7[o1#name hasValue Y_8]:o1#person.
	axiom mapping007 definedBy
	  mediated(Y_11, o2#man)[A_9 hasValue AR_10]:-
	  mediated(Y_11, o2#human)[A_9 hasValue AR_10],
	  Y_11[o1#hasGender hasValue o1#male].

6 Implementation and Prototype

The ideas and methods presented in this paper are used in the mediation component of the WSMX architecture. The WSMX Data Mediation component is designed to support data transformation, which means to transform the source ontology instances entering the system into instances expressed in terms of the target ontology. As described above, in order to make this possible the data mediation process consists of a design-time and a run-time phase. Each of these two phases has its own implementations: the *Ontology Mapping Tool* and the *Run-time Data Mediator*.

The *Ontology Mapping Tool* is implemented as an Eclipse plug-in, part of the Web Service Modeling Toolkit (WSMT)[3] [7] an integrated environment for ontology creation, visualization and mapping. The Ontology Mapping Tool is currently compatible with WSMO ontologies (but by providing the appropriate wrappers different ontology languages could be supported); it offers different ways of browsing the ontologies using perspectives and allows the domain expert to create mappings between two ontologies (source and target) and to store them in a persistent mapping storage. Currently only the *PartOf* and *InstanceOf* perspectives are implemented while decomposition and context principles are fully supported. These principles, together with the suggestion mechanisms make the prototype a truly semi-automatic ontology mapping tool.

[3] Open Source Project available at http://sourceforge.net/projects/wsmt

The *Run-time Data Mediator* plays the role of the data mediation component in WSMX (available together with the WSMX system[4]. It uses the abstract mappings created during design-time, grounds them to WSML and uses a reasoner to evaluate them against the incoming source instances. The storage used is a relational data base. The Run-time Data Mediator is also available as a stand alone application.

7 Related Work

MAFRA [10] proposes a Semantic Bridge Ontology to represent the mappings. This ontology has as central concept, the so called "Semantic bridge" which is the equivalent of our mapping language statements. The main difference to our approach is that MAFRA does not define any explicit relation between the graphical representation of the ontologies in their tool and the generation of these Semantic Bridges or between the user's actions and the particular bridges to be used. The formal abstract model we propose links the graphical elements of the user interface with the mapping representation language, ensuring a clear correspondence between user actions and the generated mappings.

PROMPT[9] is an interactive and semi-automatic algorithm for ontology merging. The user is asked to apply a set of given operations to a set of possible matches, based on which, the algorithm recomputes the set of suggestions and signals the potential inconsistencies. The fundamental difference in our approach is that instead of defining several operations we have only one operation (*map*) which will take two ontology elements as arguments, and multiple *perspectives* to graphically represent the ontologies in the user interface. Based on the particular types of perspectives used and on the roles of the *map* action arguments in that perspective the tool is able to determine the type of mapping to be created. Such that, by switching between perspectives, different ontology mismatches can be addressed by using a single *map* action. An interesting aspect is that PROMPT defines the term *local context* which perfectly matches our *context* definition: the set of descriptions attached to an item together with the items these descriptions point to. While PROMPT uses the local context in decision-making when computing the suggestions, we also use the context when displaying the ontology.

Instead of allowing browsing on multiple hierarchical layers, as PROMPT and MAFRA do, we adopt a context based browsing that allows the identification of the domain experts intentions and generates mappings.

8 Conclusion and Further Work

In this paper we define a formal model for mapping creation. This model sits between the graphical elements used to represent the ontologies and the result of the mapping process, i.e. the ontology alignment. By defining both the graphical

[4] Open Source Project available at http://sourceforge.net/projects/wsmx

instruments and the mapping creation strategies in terms of this model we assure a direct and complete correspondence between human user action and the effect on the generated ontology alignment. In addition we propose a set of different graphical perspectives that can be linked with the same model, each of them offering a different viewpoint on the displayed ontology. By combining this types of perspectives different types of mismatches can be addressed in an identical way from one pair of views to the other.

As future work, we plan to focus in identifying more relevant perspectives and to investigate the possible combination of these perspectives in respect with the types of mappings to be created. These would lead in the end to defining a set of mapping patterns in terms of our model, which will significantly improve the mappings finding mechanism. Another point to be investigated is the mapping with multiple participants from the source and from the target ontology. In this paper we investigated only the cases when exactly one element from the source and exactly one element from the target can be selected at a time to be mapped. We plan to also address transformation functions from the perspective of our model. Such transformation functions (e.g. string concatenation) would allow the creation of new target data based on a combination of given source data.

References

1. J. de Bruijn, D. Foxvog, and K. Zimmerman. Ontology mediation patterns library. SEKT Project Deliverable D4.3.1, Digital Enterprise Research Institute, University of Innsbruck, 2004.
2. M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with APFEL. *Fourth International Semantic Web Conference (ISWC-2005)*, 2005.
3. C. Feier, A. Polleres, R. Dumitru, J. Domingue, M. Stollberg, and D. Fensel. Towards intelligent web services: The web service modeling ontology (WSMO). *International Conference on Intelligent Computing (ICIC)*, 2005.
4. M. R. Genesereth and N. J. Nilson. *Logical Foundations of Artificial Inteligence*. Morgan-Kaufmann, 1988.
5. A. Polleres H. Lausen, J. de Bruijn and D. Fensel. WSML - A Language Framework for Semantic Web Services. *W3C Workshop on Rule Languages for Interoperability*, April 2005.
6. A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-Oriented Architecture. *International Conference on Web Services (ICWS 2005)*, July 2005.
7. M. Kerrigan. WSMOViz: An Ontology Visualization Approach for WSMO. *10th International Conference on Information Visualization*, 2006.
8. A. Mocan and E. Cimpian. Mapping creation using a view based approach. *1st International Workshop on Mediation in Semantic Web Services (Mediate 2005)*, December 2005.
9. N.F. Noy and M. A. Munsen. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 6(59), 2003.
10. N. Silva and J. Rocha. Semantic web complex ontology mapping. *Proceedings of the IEEE Web Intelligence (WI2003)*, page 82, 2003.