# 4
# Complex schema matching

We address in this chapter the problem of matching two schemas that belong to an expressive OWL dialect. We adopt an instance-based approach and, therefore, assume that a set of instances from each schema is available.

First, we decompose the problem of OWL schema matching into the problem of vocabulary matching and the problem of concept mapping. We first describe a vocabulary matching technique based on the notion of similarity. Then, we evaluate the precision of the technique using data available on the Web. Finally, we also introduce sufficient conditions guaranteeing that a vocabulary matching induces a correct concept mapping.

Unlike any of the previous instance-based techniques presented in section 1.2, the matching process we describe uses similarity functions to induce vocabulary matchings in a non-trivial way, coping with an expressive OWL dialect. We also illustrate, through a set of examples, that the structure of OWL schemas may lead to incorrect concept mappings and indicate how to avoid such pitfalls.

## 4.1.
## OWL Extralite

We will work with an OWL dialect, that we call *OWL Extralite*. It supports named classes, datatype and object properties, subclasses, and individuals. The *domain* of a datatype or object property is a class, the *range* of a datatype property is an XML schema type, whereas the *range* of an object property is a class. As *property restrictions*, the dialect admits *minCardinality* and *maxCardinality*, with the usual meaning. As *property characteristic*, it allows just the *inverseFunctional* property, which captures simple keys. We note that only OWL Full supports the inverseFunctional property for datatype properties.

An *OWL schema* (more often called an *OWL ontology*) is a collection of RDF triples that use the OWL vocabulary. A *concept* of an OWL schema is a class, datatype property or object property defined in the schema. The *vocabulary* of the schema is the set of concepts defined in the schema (a set of URIrefs). The scope of a property name is global to the OWL schema, and not local to the class indicated as its domain.

In the rest of the thesis, when we refer to a *schema* we mean an OWL Extralite schema.

Figure 9 and Figure 10 show OWL schemas for fragments of the Amazon and the eBay databases, using a simplified and unofficial notation to save space and improve readability. Consistently with XML usage, from this point on, we will use the namespace prefixes `am:` and `eb:` to refer to the vocabularies of the Amazon and the eBay OWL schemas, and qualified names of the form `V:T` to indicate that `T` is a term of the vocabulary `V`.

In Figure 9, for example, `am:title` is defined as a datatype property with domain `am:Product` and range `string` (an XML Schema data type), `am:Book` is declared as a subclass of `am:Product`, and `am:publisher` is defined as an object property with domain `am:Book` and range `am:Publ`. Note that the scope of `am:title` and `am:publisher` is the schema, and not the classes defined as their domains.

```
Product
  title         range string
  listPrice     range decimal
  currency      range string
Book is-a Product
  author        range string
  edition       range integer
  isbn          range string
  ean           range string
  detailPageURL range anyURI
  publisher     range Publ
Publ
  name          range string
  address       range string
Music is-a Product
Video is-a Product
PCHardware is-a Product
```

Figure 9. An OWL schema for the Amazon Database.

Furthermore, although not indicated in Figure 9, we assume that all properties, except `am:author`, have maxCardinality equal to 1, and that `am:isbn` is inverseFunctional. This means that all properties are single-valued, except `am:author`, which is multi-valued, and that `am:isbn` is a key of `am:Book`. Likewise, although not shown in Figure 10, all properties, except `eb:author`, have maxCardinality equal to 1, and `eb:isbn-10` and `eb:isbn-13` are inverseFunctional.

```
Seller
  name                  range string
  redistrationDate      range dateTime
  offers                range Offer
Offer
  quantity              range integer
  startPrice            range double
  currency              range string
  seller                range Seller
  product               range Product
Product
  title                 range string
  condition             range string
  returnPolicyDetails   range string
  offers                range Offer
Book is-a Product
  author                range string
  edition               range integer
  publicationYear       range integer
  isbn-10               range integer
  isbn-13               range integer
  publisher             range string
  binding               range string
  condition             range string
Music is-a Product
DVDMovies is-a Product
ComputerNetworking is-a Product
```

Figure 10. An OWL schema for the eBay Database.

Finally, to express mappings, we adopt the Semantic Web Rule Language (SWRL) (Horrocks et al. 2004). However, we also opted for a simplified, datalog-like syntax to improve readability and save space.

An example of an SWRL rule in our simplified syntax would be:

```
eb:publisher(b,n) ← am:publisher(b,p), am:name(p,n)
```

which says that, if `b` and `p` are related by `am:publisher`, and `p` and `n` by `am:name`, then `b` and `n` are related by `eb:publisher`.

## 4.2. Vocabulary matching

### 4.2.1. Formal definition of vocabulary matching

Let *S* and *T* be two (OWL Extralite) schemas, and $V_S$ and $V_T$ be their vocabularies, respectively. Let $C_S$ and $C_T$ be the sets of classes, and $P_S$ and $P_T$ be the sets of datatype or object properties in $V_S$ and $V_T$, respectively.

A *contextualized vocabulary matching* between *S* and *T* is a finite set $\mu_V$ of quadruples $(v_1,e_1,v_2,e_2)$ such that

(i) if $(v_1,v_2)\in C_S\times C_T$, then $e_1$ and $e_2$ are the top class `T`

(ii) if $(v_1,v_2)\in P_S\times P_T$, then $e_1$ and $e_2$ are classes in $C_S$ and $C_T$ that must be subclasses of the domains of $v_1$ and $v_2$, respectively

(iii) and these are the only possible quadruples in $\mu_V$

If $(v_1,e_1,v_2,e_2)\in \mu_V$, we say that $\mu_V$ *matches* $v_1$ *with* $v_2$ *in the context of* $e_1$ *and* $e_2$, that $e_i$ *is the context of* $v_i$ and that $(v_i,e_i)$ is a *contextualized concept*, for $i=1,2$. A *contextualized property (or class) matching* is a matching defined only for properties (or classes).

Intuitively, a vocabulary matching expresses equivalences between properties and classes in a given context. The context of a property `P` in a vocabulary matching is an RDF class that specifies the `rdf:type` of subjects of existing triples of the form `(?subject P ?object)` for which the matchings holds. The context of a class is always the top class `T` (i.e., this notion is not used for class matchings).

Table 19 is a fragment of a vocabulary matching between the Amazon and the eBay schemas. The first line of the table indicates that property `am:title`, associated with instances of class `am:Book`, is equivalent to property `eb:Book`, when associated with instances of class `eb:Book`, and the fourth line of the table indicates that classes `am:Book` and `eb:Book` are equivalent.

Table 19: Fragment of a vocabulary matching between Amazon and eBay schemas.

| Amazon Database | | eBay Database | |
|---|---|---|---|
| Concept | Context | Concept | Context |
| `am:title` | `am:Book` | `eb:title` | `eb:Book` |
| `am:title` | `am:Music` | `eb:title` | `eb:Music` |
| `am:name` | `am:Publ` | `eb:publisher` | `eb:Book` |
| `am:Book` | `T` | `eb:Book` | `T` |
| `am:Music` | `T` | `eb:Music` | `T` |

## 4.2.2. Instance-based vocabulary matching

In this section, we describe an instance-based process to create contextualized vocabulary matchings.

Recall from Section 3.3 that a property *A* of a catalogue *C* may be represented by the set of values of *A* that occur in the current extension ***C*** of *C*, or by the set of pairs *(i,v)* such that *v* is the value of *A* for an instance *i* that occurs in ***C***. If the domain of *A* is a set of strings, the set of values is replaced by a set of tokens, and the property representations are reinterpreted accordingly. Similarity models are then applied to such property representations to generate property matchings between two catalogue schemas.

We also recall that the instance matching technique of Bilke and Naumann (Bilke and Naumann 2005) represents each database tuple as a character string and uses k-mean clustering algorithms to find duplicate tuples. However, we note that the representations of the same object in distinct databases may differ in the list of properties and in the property values. As a consequence, we may end up with dissimilar tuples that represent the same object.

For example, suppose that we apply the Bilke and Naumann technique to match instances that represent the book “The Tragedy of Romeo and Juliet”. Table 20 shows their lists of property-value pairs. If we measure the similarity between the sets of tokens extracted from all property values of each instance, we obtain a score of 43% of common tokens (Figure 11). By contrast, if we consider only the values of the properties that match, the similarity increases to 64% (Figure 12).

However, note that, to extract tokens from the values only of properties that match, we have to know that `am:book` matches `eb:book`, and have access to the set of matching properties of such classes.

Combining these observations, we propose a four-step schema matching process, as follows:

1) Generate a preliminary property matching using similarity functions.
2) Use the property matching obtained in Step (1) to generate: (a) a class matching; and (b) a class instance matching.
3) Use the class matching and the class instance matching obtained in Step (ii) to generate a refined contextualized property matching.
4) The final vocabulary matching is the result of the union of the property and class matchings obtained in Steps (2) and (3).

Table 20. Example the same book instance representation in eBay and Amazon.

| eBay | Amazon |
|---|---|
| `isbn-10 = “039577537X”` | `isbn = “039577537X”` |
| `isbn-13 = 9780395775370` | `ean = 9780395775370` |
| `title = “The Tragedy of Romeo and Juliet”` | `title = “Tragedy of Romeo and Juliet: And Related Readings (Literature Connections)”` |
| `author = “William Shakespeare”` | `author = “William Shakespeare”` |
| `publisher = “Houghton Mifflin”` | `name = “Houghton Mifflin Company”` |
| `returnPolicyDetails = “NO RETURNS ARE ACCEPTED”` | - |
| `condition = “Like New”` | - |
| `binding = “Hardcover”` | - |
| - | `listPrice = 18.92` |
| - | `currency = “USD”` |

Step (1) generates preliminary property matchings based on the intuition that *“two properties match iff they have many values in common and few values not in common”*. Step (2) creates class matchings that reflect the intuition that *“two classes match iff they have many matching properties”*. However, to work correctly, Step (2) requires that Step (1) generates preliminary property matchings only for highly similar properties.

For example, in the experiments described in Section 4.2.3, with data from the eBay and the Amazon databases, if we use a threshold $\tau$=0.12, then

Instance representation

i) eBay

{**039577537x**, **9780395775370**, **tragedy**, **romeo**, **juliet**, **william**, **shakespeare**, **houghton**, **mifflin**, returns, accepted, like, new, hardcover}

ii) Amazon

{**039577537x**, **9780395775370**, **tragedy**, **romeo**, **juliet**, related, readings, literature, connections, **william**, **shakespeare**, **houghton**, **mifflin**, company, 18.92, usd}

Similarity

9 common tokens (in bold face) in a total of 21 tokens from both instances ⇒ 43% of commonalities

Figure 11. Similarity of the instances in Table 20 based on the set of tokens representation, where tokens were extracted from all properties.

`eb:level` with context `eb:Seller` matches `am:color` with context `am:PCHardware`, and `eb:title` with context `eb:Music` matches `am:title` with content `am:Video`. These property matchings may cause classes `eb:Seller` and `am:PCHardware` to match, as well as `eb:Music` and `am:Video`, depending on the threshold and the total amount of common properties among the classes (as discussed below, class matching depends on the similarity between sets of properties). If we increase the threshold to 0.13, the previous property matchings do not hold and we may avoid the above unwanted class matchings.

In what follows, let $S$ and $T$ be two schemas, $V_S$ and $V_T$ be their vocabularies, $P_S$ and $P_T$ be their sets of properties, and $C_S$ and $C_T$ be their sets of

Instance representation

i) eBay

{**039577537x**, **9780395775370**, **tragedy**, **romeo**, **juliet**, **william**, **shakespeare**, **houghton**, **mifflin**}

ii) Amazon

{**039577537x**, **9780395775370**, **tragedy**, **romeo**, **juliet**, related, readings, literature, connections, **william**, **shakespeare**, **houghton**, **mifflin**, company}

Similarity

9 common tokens (in bold face) in a total of 14 tokens from both instances ⇒ 64% of commonalities

Figure 12. Similarity of the instances in Table 20 based on the set of tokens representation, where tokens were extracted from matching properties.

classes, respectively. Let $\boldsymbol{U_S}$ and $\boldsymbol{U_T}$ be fixed sets of instances of $S$ and $T$, respectively, used to compute the vocabulary matchings.

Let $\mathcal{U}$ be the universe of all tokens extracted from literals and all URIrefs. Consider a similarity function $\sigma : 2^{\mathcal{U}}\times 2^{\mathcal{U}}\rightarrow[0,1]$, a *similarity threshold* $\tau\in[0,1]$ and a *related similarity threshold* $\tau'\in[0,1]$ such that $\tau' < \tau$.

For each property $P\in P_S$, for each class $C\in C_S$ such that $C$ is the domain of $P$ or a subclass of the domain of $P$, consider the contextualized property $P^C=(P,C)$ and construct the set $o[U_S,P^C]$ of all values $v$ such that there are triples of the form $(I,P,v)$ and $(I,$`rdf:type`$,C')$ in $U_S$, where $C'=C$ or $C'$ is a subclass of $C$, and likewise for a property in $P_T$. We call $o[U_S,P^C]$ the *observed-value representation* for $P^C$ in $U_S$. This construction explores the fact that $P$ is inherited by all subclasses of its domain.

The basis of the matching process are the *matchings directly induced by* $\sigma$ *and* $\tau$, defined as follows.

The *contextualized property matching between S and T induced by* $\sigma$ *and* $\tau$, and based on the observed-value representation for properties, is the relation $\mu_P$ such that

$$(P,C,Q,D)\in\mu_P \text{ iff } \sigma(o[U_S,P^C],o[U_T,Q^D])\geq\tau$$

For each class $C$ in $C_S$, let $props[S,C]$ be the set of properties in $P_S$ whose domain is $C$ or that $C$ inherits from its superclasses, and likewise for classes in $C_T$. We call $props[S,C]$ the *representation* of $C$ in $U_S$.

The *contextualized class matching between S and T induced by* $\sigma$, $\tau$ and $\mu_P$ is the relation $\mu_C\subseteq C_S\times C_T$ such that (recall that $\top$ is the top class)

$$(C,\top,D,\top)\in\mu_C \textit{ iff } \sigma(props[S,C],relprops[S,C,T,D]))\geq\tau$$

where $relprops[S,C,T,D]$ denotes the set of properties $P$ of class $C$ of $S$ such that there is a property $Q$ of class $D$ of $T$ such that $(P,C,Q,D)\in\mu_P$. Note that it does not make sense to directly compute $\sigma(props[S,C],props[T,D])$, since $props[S,C]$ and $props[T,D]$ are sets of URIrefs from different vocabularies. To avoid this problem, we replaced $props[T,D]$ by $relprops[S,C,T,D]$.

From the matchings directly induced by $\sigma$ and $\tau$, the process then derives a

```
INSTANCE-MATCHING(S,T,$\mu_C$)
$\mu_I$ = { }
for each pair of classes (C,D) in S and T such that
$\mu_C$ matches C with D
  for each pair of instances (I,J) of C and D in $\boldsymbol{U_S} \times \boldsymbol{U_T}$
    if $\sigma$(t[$\boldsymbol{U_S}$,C](I),t[$\boldsymbol{U_T}$,D](J)) $\geq \tau$ then
      $\mu_I$ = $\mu_I \cup$ {(I,C,J,D)}
return $\mu_I$
```

Figure 13. The class instance matching algorithm.

class instance matching and a refined contextualized property matching, as follows.

Figure 13 shows the class instance matching algorithm. It receives as input *S* and *T*, and the class matching $\mu_C$ induced by $\sigma$, $\tau$ and $\mu_P$. It also implicitly receives as input $\boldsymbol{U_S}$ and $\boldsymbol{U_T}$. It outputs a class instance matching $\mu_I$ between class instances in $\boldsymbol{U_S}$ and $\boldsymbol{U_T}$. In Figure 13, if *C* is a class in $C_S$, and *I* is an instance of *C* in $U_S$, then $t[\boldsymbol{U_S},C](I)$ denotes the set of tokens extracted from all values *v* such that, for some property $P \in P_S$, for some property *Q* in $P_T$, for some class $D \in C_T$, there is a triple $(I,P,v)$ in $U_S$, and there is a tuple $(P,C,Q,D)$ in $\mu_P$, and likewise for $t[\boldsymbol{U_T},D](J)$.

For each $(P,C,Q,D) \in \mu_P$ such that $(C,\top,D,\top) \in \mu_C$, construct the set *q* of triples $(I,u,v)$ such that there are triples of the form $(I,P,u)$ and *(I,*`rdf:type`*,C)* in $U_S$, there are triples of the form $(J,Q,v)$ and *(J,*`rdf:type`*,D)* in $U_T$, and $(I,C,J,D) \in \mu_I$ (where $\mu_I$ is the class instance matching of Figure 13). Define $iv[P,C,Q,D]=(s,t)$ such that $s=\{(I,u)/(\exists v)(I,u,v) \in q\}$ and $t=\{(I,v)/(\exists u)(I,u,v) \in q\}$. We call *s* the *instance-value representation* of $P^C$ in $\boldsymbol{U_S}$ (and likewise for *t*). This second representation for properties is useful since it helps distinguish properties with similar sets of values, but which refer to distinct instances, matched by $\mu_I$.

Figure 14 shows the refined contextualized property matching algorithm. It has the same input as the algorithm in Figure 13, including the class matching $\mu_C$ induced by $\sigma$, $\tau$ and $\mu_P$, and outputs a contextualized property matching $\mu_P$ only between properties whose domains are classes directly or indirectly matched by $\mu_C$. The algorithm uses the maximum of the similarity values computed using the observed-value and the instance-value representations for a pair of properties *P* and *Q*, and the more relaxed similarity threshold. Although not shown in Figure 14, object properties receive a special treatment, since their representations are sets of URIrefs that are compared with help of the class instance matching $\mu_I$ (computed by the algorithm in Figure 13).

The final vocabulary matching $\mu_V$ is the union of the class matching $\mu_C$ induced by $\sigma$, $\tau$ and $\mu_P$ and the contextualized property matching $\mu_P$ computed by the algorithm in Figure 14. However, $\mu_V$ may have to be adjusted, by dropping matchings, until it becomes structurally correct (see Section 4.3.4).

```
CONTEXTUALIZED-PROPERTY-MATCHING(S,T,$\mu_C$)
$\mu_P$ = { }
for each pair of classes (C,D) in S and T such that
$\mu_C$ matches C with D
or C′ dominates C and $\mu_C$ matches C′ with D
or D′ dominates D and $\mu_C$ matches C with D′
  for each pair (P,Q) of properties of C and D
    X = $\sigma$(o[$U_S$,$P^C$],o[$U_T$,$Q^D$])
    if (C matches D) then
      (s,t)=iv[P,C,Q,D]
      Y = $\sigma$(s,t)
    else
      Y = 0
    if max(X,Y) $\geq$ $\tau'$ then
      $\mu_P$ = $\mu_P \cup$ {(P,C,Q,D)}
return $\mu_P$
```

Figure 14. The contextualized property matching algorithm.

## 4.2.3. Experimental results

We conducted an experiment to assess the performance of the vocabulary matching process of Section 4.2.3, using data about products obtained from Amazon and eBay.

We tested the process with data downloaded from the Web, rather than with the benchmark proposed in (Duchateau et al. 2007), since the benchmark does not include instances and is therefore unsuitable to test our process.

We first defined a set of terms, which were used to query the databases. From the query results, we extracted the less frequent terms common to both databases. We then used these terms to once more query the databases. This pre-processing step enhanced the probability of retrieving duplicate objects from the databases, which is essential to evaluate any instance-based schema matching technique. We extracted a total of 116,201 records: 16,410 from Amazon and 99,791 from eBay.

We adopted as similarity functions the contrast model (Leme et al. 2008b), for property matchings, and the cosine distance with TF/IDF, for instance matchings. The experiments lead us to conclude that the contrast model has a better performance when we want to emphasize the difference between two sets of values. This follows because the contrast model has room for calibrating several parameters.

The configuration of the similarity models in this experiment was as follows (see the Appendix for more details on how to setup and calibrate a similarity model):

- Configuration for the temporary property matching:
  *multiset*
  *similarity based on the contrast model*
  *threshold = max similarity - 20%*
- Configuration for the instance matching:
  *multiset*
  *cosine with TF-IDF*
  *threshold = 0,8*
- Configuration for the refinement of the property matching:
  *does not use multiset*
  *similarity based on the contrast model*
  *threshold = max similarity - 30%*

Table 21 shows the vocabulary matching obtained. The headings indicate that $e_1$ is the context of $v_1$, and $e_2$ that of $v_2$. Also, “`B`” abbreviates classes `eb:Book` and `am:Book`, and similarly for the other classes.

Note that both properties, `eb:format` and `eb:biding`, in the context of Books matches the property `am:format`. This is because the databases hold similar information in the first two properties.

The rightmost column of Table 21 classifies the matchings as follows: *tp* for true positive, *fp* for false positive and *fn* for false negative. Since the number of true positives is 25, that of false positives is 4 and that of false negatives is 10, the performance measures are:

$$precision = \frac{tp}{tp + fp} = 86\%$$

$$recall = \frac{tp}{tp + fn} = 71\%$$

$$f = 2 * \frac{precision \cdot recall}{precision + recall} = 78\%$$

The highlighted lines in Table 21 refer to matchings that would have been considered false negatives, if the algorithm in Figure 14 ignored the instance-value representation for properties. In this case, the performance measures would drop to:

$$precision = 82\%, recall = 51\%, f = 63\%$$

Lastly, if we consider only the class matchings in Table 21, the performance measures are:

$$precision = 80\%, recall = 80\%, f = 80\%$$

Table 21. Automatically obtained vocabulary matching from eBay into Amazon

| eBay | | Amazon | | |
|---|---|---|---|---|
| $v_1$ | $e_1$ | $v_2$ | $e_2$ | |
| Books | | Books | | tp |
| author | B | author | B | tp |
| binding | B | biding | B | tp |
| edition | B | edition | B | tp |
| format | B | biding | B | tp |
| isbn-10 | B | isbn | B | tp |
| isbn-13 | B | ean | B | tp |
| publisher | B | name | P | tp |
| title | B | title | B | tp |
| ComputerNetworking | | PCHardware | | tp |
| operatingSys | CN | platform | PC | tp |
| operatingSystem | CN | operSyst | PC | tp |
| processorConfig | CN | cpuType | PC | tp |
| processorType | CN | cpuType | PC | tp |
| processorType | CN | cpuManufact | PC | tp |
| title | CN | title | PC | tp |
| DVDMovies | | Video | | tp |
| director | DVD | director | V | tp |
| leadingRole | DVD | actor | V | tp |
| title | DVD | title | V | tp |
| upc | DVD | upc | V | tp |
| Music | | Music | | tp |
| artist | M | artist | M | tp |
| format | M | biding | M | tp |
| title | M | title | M | tp |
| editionDesc | B | format | B | *fp* |
| Offer | | Books | | *fp* |
| currency | O | currencyCode | B | *fp* |
| startPrice | O | listPrice | B | *fp* |
| brand | CN | brand | PC | *fn* |
| hardDriveCap | CN | hardDiskSize | PC | *fn* |
| memoryRam | CN | systemMemory | PC | *fn* |
| processorSpeed | CN | cpuSpeed | PC | *fn* |
| screenSize | CN | displaySize | PC | *fn* |
| format | DVD | biding | V | *fn* |
| releaseDate | DVD | releaseDate | V | *fn* |
| ReleaseDate | M | releaseDate | M | *fn* |
| upc | M | upc | M | *fn* |
| Product | | Product | | *fn* |

## 4.3. Concept mapping

### 4.3.1. Informal definition of concept mapping rules

In this section, we informally introduce the notions of *vocabulary matching* and *concept mapping*, formally defined in sections 4.2 and 4.3, respectively.

A *concept mapping from a source schema S to a target schema T* is a set of transformation rules that express concepts of the target schema *T* in terms of concepts of the source *S* such that it is possible to translate queries over *T* to queries over *S*.

In the context of this thesis we consider queries defined in the SPARQL Query Language for RDF (Prud'hommeaux and Seaborne 2008). In this language a query consists of two parts: the `SELECT` clause identifies the variables to appear in the query results, and the `WHERE` clause provides the basic graph pattern to match against the data graph.

A *basic graph pattern* is a set of triple patterns. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. A basic graph pattern *matches* a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is an RDF graph equivalent to the subgraph.

For example, the query of Figure 15.a returns titles of book instances from the Amazon database. The variable `?b` in lines 5 and 6 means that the same instances that have property `am:title` must be instances of the class `am:Book`. If line 5 were omitted, the query would return titles of instances of the classes `am:Book`, `am:Music`, `am:Video` and `am:PCHardware`. Figure 15.b returns titles and authors of books but, in this case, there is no need to restrict the type of instances, since all instances that have the property `am:author` must be books (Figure 9). Figure 15.c extends the second query by adding to the result set the publisher of the books. Line 6 of Figure 15.c is analogous to an equijoin in the relational model that joins the class `am:Book` with the class `am:Publisher` of the Amazon schema (Figure 9).

The `SELECT` keyword may be replaced by `CONSTRUCT` in order to return an RDF graph instead of a relation. Figure 16 shows a SPARQL code which returns triples of publisher names from Amazon as triples of eBay.

```
PREFIX am:<...>
PREFIX rdf:<...>
SELECT ?title
WHERE
  {?b rdf:type am:Book.
   ?b am:title ?title}
```

a) Simple SPARQL query over the Amazon database which returns titles of books

```
PREFIX am:<...>
SELECT ?title ?author
WHERE
  {?b am:author ?author.
   ?b am:title ?title}
```

b) Simple SPARQL query over the Amazon database which returns title and author of books

```
PREFIX am:<...>
SELECT ?title ?author ?pub
WHERE
  {?b am:author ?author.
   ?b am:title ?title.
   ?b am:publisher ?p.
   ?p am:name ?pub}
```

c) Simple SPARQL query over the Amazon database which returns title, author and publisher of books

Figure 15. Simple SPARQL queries over the Amazon database with schema in Figure 9

After this brief overview of SPARQL, we return to the problem of concept mapping. From Figure 9 and Figure 10 and an analysis of the database schemas, one may infer that the properties named `title` from both schemas are likely to

```
PREFIX am:<...>
PREFIX eb:<...>
CONSTRUCT {?b eb:publisher ?pub}
WHERE
  {?b am:publisher ?p.
   ?p am:name ?pub}
```

Figure 16. Simple SPARQL query for returning a RDF graph

be equivalent because their names are syntactically equal and the experiments, which we will describe later in this chapter, showed that the set of values of both properties are very similar. A rule in SWRL which expresses this mapping can be written as follows:

`eb:title(p,t) ← am:title(p,t)`

This rule means that the concept `eb:title` of eBay can be translated to the concept `am:title` of Amazon, i.e., triple patterns of the form `(?s eb:title ?o)` can be translated to `(?s am:title ?o)`. Figure 17 shows an eBay query on the left and its translation to the Amazon schema on the right.

```
PREFIX eb:<...>
SELECT ?title
WHERE {?s eb:title ?title}
```

```
PREFIX am:<...>
SELECT ?title
WHERE {?s am:title ?title}
```

Figure 17. Equivalent queries over the eBay and the Amazon schemas that return titles.

However, a deeper analysis might not confirm the equivalence for all classes which inherits the property `title` in both databases. For example, consider Table 19 which says that the property `am:title` of instances of `am:Book` is equivalent to the property `eb:title` of instances of `eb:Book` and, likewise, for property `title` of `Music` in both databases. In addition, the table matches `am:name` with `eb:publisher`, `am:Book` with `eb:Book` and `am:Music` with `eb:Music`. In view of this, the previous rule should be replaced by the following two rules.

`eb:title(p,t) ← am:title(p,t), am:Book(p)`

`eb:title(p,t) ← am:title(p,t), am:Music(p)`

Note that Table 19 omit pairs of the *title* property in the context of movies and computers. This means that the equivalence between the property `eb:title` and `am:title` does not hold in these contexts.

More precisely, these new rules state that, although `eb:title` refers to titles of all types of eBay products (subclasses of `eb:Product`), this property has a narrower meaning in the Amazon database: `eb:title` can only be considered equivalent to `am:title` for instances of type `am:Book` and `am:Music`, although `eb:title` is a valid property for other classes in the eBay schema.

```
PREFIX eb:<...>
SELECT ?title
WHERE {?s eb:title ?title}
```

```
PREFIX am:<...>
PREFIX rdf:<...>
SELECT ?title
WHERE
  {{?s am:title ?title.
    ?s rdf:type am:Book}
   union
   {?s am:title ?title.
    ?s rdf:type am:Music}}
```

Figure 18. Equivalent queries over eBay and Amazon databases that return titles when only instances of titles of books and music are equivalent

The translation of the concept `eb:title` of queries over the eBay schema to the Amazon schema is as exemplified in Figure 18.

Recall that in a SPARQL query, the triple pattern `(?s rdf:type C)` indicates that `C` is a class and the triple pattern `(?s P ?o)` expresses that `P` is a property. Therefore, we provide rules for translating only these two types of triples. The approach we propose does not cover triple patterns with variables in the property position.

The third line in Table 19 matches the property `am:name` in the context of `am:Publ` with the property `eb:publisher` in the context of `eb:Book`. Consistently with the previous discussion, we would generate the following mapping rule:

```
eb:publisher(b,n)← am:name(b,n), am:Publ(b)
```

The query translation shown in Figure 19.a illustrates the use of this rule.

However, the above rule is a *wrong* because:

- the domain of `am:name` is `am:Publ` and the domain of `eb:publisher` is `eb:Book`
- Table 19 does not match the `am:Publ` and the `eb:Book` classes

The correct mapping rule would be as follows:

```
eb:publisher(b,n)←
      am:publisher(b,p), am:name(p,n), Publ(p)
```

Note that this rule can be optimized by omitting `Publ(p)` because `Publ` is the domain of the property name.

```
eb:publisher(b,n)← am:publisher(b,p), am:name(p,n)
```

The query translation shown in Figure 19.b illustrates the use of this second rule. This translation can be regarded as a *raw translation*, automatically generated by the mapping rules. However, note that the second part of the union

```
PREFIX eb:<...>
SELECT ?title, ?pub
WHERE
  {?s eb:title ?title.
   ?s eb:publisher ?pub}
```

```
PREFIX am:<...>
PREFIX rdf:<...>
SELECT ?title, ?pub
WHERE
  {{?s am:title ?title.
    ?s am:name ?pub.
    ?s rdf:type am:Book}
   union
   {?s am:title ?title.
    ?s am:name ?pub.
    ?s rdf:type am:Music}}
```

a) Wrong translation of eBay query

```
PREFIX eb:<...>
SELECT ?title, ?pub
WHERE
  {?s eb:title ?title.
   ?s eb:publisher ?pub}
```

```
PREFIX am:<...>
PREFIX rdf:<...>
SELECT ?title, ?pub
WHERE
  {{?s am:title ?title.
    ?s am:publisher ?p.
    ?p am:name ?pub.
    ?p rdf:type am:Publ.
    ?s rdf:type am:Book}
   union
   {?s am:title ?title.
    ?s am:publisher ?p.
    ?p am:name ?pub.
    ?p rdf:type am:Publ.
    ?s rdf:type am:Music}}
```

b) Correct translation of eBay query

Figure 19. Equivalent queries over eBay and Amazon databases that return titles when only instances of titles of books and music are equivalent

asks for publishers of instances of the class `am:Music`. But, according to Figure 9, `am:publisher` is defined only for instances of type `am:Book`. Hence, this part of the union would return an empty set. Therefore, the query can be optimized in a subsequent processes to improve performance, a problem we do not address in this thesis.

In general, a triple pattern of the target schema has to be translated to a graph pattern of the source schema that connects objects of the same type. In the previous example, `{?s eb:publisher ?pub}` connects an `eb:Book` instance, denoted by the variable `?s`, to an `xsd:string` literal, denoted by the variable `?pub`, and so does the graph pattern `{?s am:publisher ?p. ?p am:name ?pub}`.

However, we note that only part of the previous rule can be directly derived from the vocabulary matching, for the following reasons:

1. The rule provides a translation for `eb:publisher`, a concept of the target schema (eBay) .
2. The vocabulary matching of Table 19 matches `am:name`, a concept of the source schema (Amazon), with `eb:publisher`, i.e., the matching indicates that `am:name` and `eb:publisher` have similar values.
3. However, Table 19 does not match `am:Publ`, the context (which, in this case, is the domain) of `am:name`, with `eb:Book`, the context (which, in this case, is also the domain) of `eb:publisher`.
4. The body of the rule has to be generated from the schemas and the vocabulary matching. It reflects a *path* from `am:Book`, the class that matches the context `eb:Book` of `eb:publisher`, to the class `am:Publ`, the context of `am:name`.

If `am:Book` and `eb:Book` did not match or there were no path from `am:Book` to `am:Publ`, we would say that the vocabulary matching were *inconsistent* . In this case, we would have to remove rows from the vocabulary matching until it became consistent.

Because of the above considerations, we say that the concept mapping, expressed by the above rules, is *derived from* the vocabulary matching indicated in Table 19, or conversely the vocabulary matching of Table 19 *induces* the above concept mapping.

Note that the process of concept mapping generation is not symmetric. In our running example, if we make eBay schema the source and the Amazon schema the target, the property `am:name` will not have any translation, since there will not be a concept in the eBay schema which is equivalent to `am:Publ`.

The last type of rule aims at translating class concepts. Table 19 indicates that that `am:Book` class is equivalent to `eb:Book`. The corresponding mapping rule is as follows:

```
eb:Book(b)← am:Book(b)
```

The rule means that each triple pattern of the form `(?s rdf:type eb:Book)` should be translated to the pattern `(?s rdf:type am:Book)`.

However, consider the following query over the eBay schema:

```
PREFIX eb:<http://ebay.com>
SELECT ?s
WHERE
  {?s rdf:type eb:Product}
```

The result set contains instances of `eb:Book`, `eb:Music`, `eb:DVDMovies` and `eb:ComputerNetworking`, which are subsets of `eb:Product`. Since Table 19 only indicates that `am:Book` matches with `eb:Book` and `am:Music` matches with `eb:Music`, it is expected that an equivalent query over the Amazon schema returns instances of `am:Book` and `am:Music`. Indeed, the equivalent query over the Amazon schema would be:

```
PREFIX am:<http://amazon.com>
SELECT ?s
WHERE
  {{?s rdf:type am:Book} union
   {?s rdf:type am:Music}}
```

This translation can be achieved by adding to the concept mappings the following two rules:

```
eb:Product(p)← am:Book(p)
eb:Product(p)← am:Music(p)
```

### 4.3.2. Formal definition of concept mapping rules

Let *S* be an OWL Extralite schema in what follows.

We say that *S* is *well-formed* iff

- for any property *p* of *S*, the domain of *p* is a class of *S*
- for any object property *p* of *S*, the range of *p* is a class of *S*
- for any class *c* of *S*, if *s* is defined as a superclass of *c* in *S*, then *s* is also a class of *S*

We understand *S* as a theory *T[S]=(A[S],C[S])* in $\mathcal{ALCQI}$ (Chomicki and Saake 1998), a dialect of Description Logics, such that

- the concepts and roles of the alphabet *A[S]* are the classes and properties of *S*
- the axioms of *C[S]* are the constraints of *S*, denoted in $\mathcal{ALCQI}$ as follows:
  - a property *p* has domain *d* and range *r*: $\top \sqsubseteq \forall p.r \sqcap \forall p^{-}.d$
  - a property *p*, with range *r*, is inverse functional: $r \sqsubseteq (\leq 1\, p^{-})$
  - a property *p*, with domain *d*, has minCardinality *k*: $d \sqsubseteq (\geq k\, p)$
  - a property *p*, with domain *d*, has maxCardinality *k*: $d \sqsubseteq (\leq k\, p)$
  - a class *s* is defined as a superclass of *c*: $c \sqsubseteq s$

In what follows, we will also use from $\mathcal{ALCQI}$ the *intersection* of two concepts, denoted $c \sqcap d$, and the *subsumption* of two concepts, denoted $c \sqsubseteq d$.

Let ***V*** be the set of *variables*, which is assumed to be disjoint from the set of concepts of *S*. A *class literal* is an expression of the form *c(x)*, where *c* is a class and *x* is a variable; a *property literal* is an expression of the form *p(x,y)*, where *p* is a property and *x* and *y* are variables; a *literal* is a class literal or a property literal. A *conjunction* is a list of literals separated by commas. A *disjunction* is a list of conjunctions separated by semi-colons. (This notation should be familiar to Prolog programmers).

A *rule* is an expression of one of the forms:

- $c(x) \leftarrow B[x]$, where *c(x)* is a class literal and *B[x]* is a disjunction where the variable *x* occurs in each conjunction

- $p(x,y) \leftarrow B[x,y]$, where $p(x,y)$ is a property literal and $B[x,y]$ is a disjunction where the variables $x$ and $y$ occur in each conjunction

The literal is the *head* and the disjunction is the *body* of the rule. We use the notation $B[x]$ and $B[x,y]$ to stress which variables must occur in the body.

Let $I$ be a set of triples of $S$. The *universe* of $I$ is the set $U[I]$ of all URIrefs and literals that occur in triples of $I$.

Consistently with the notion of interpretation of Description Logics, given a class $c$ of $S$, the *interpretation of c in I* is the set

$$c^I = \{ i \in U[I] \,/\, (i, \texttt{:type}, c) \in I \}$$

and, given a property $p$ of $S$, the *interpretation of p in I* is the binary relation

$$p^I = \{ (i,o) \in U[I] \times U[I] \,/\, (i,p,o) \in I \}$$

The *interpretation* of the intersection of two concepts $c \sqcap d$ is the set

$$(c \sqcap d)^I \ = \ c^I \cap d^I$$

We say that the subsumption of two concepts $c \sqsubseteq d$ is *true in I*, denoted $I \vDash c \sqsubseteq d$, iff $c^I \subseteq d^I$.

Rather than resorting to the formalization in $\mathcal{ALCQI}$, we directly define when a constraint $\sigma$ of $S$ is *true in I*, denoted $I \vDash \sigma$:

- if $\sigma$ declares that a property $p$ has domain $d$ and range $r$, then $I \vDash \sigma$ iff $p^I \subseteq d^I \times r^I$
- if $\sigma$ declares that a property $p$, with range $r$, is inverse functional, then $I \vDash \sigma$ iff, for any $b \subseteq r^I$, $card(\{ a \in U[I] \,/\, (a,b) \in p^I \}) \leq 1$
- if $\sigma$ declares that a property $p$, with domain $d$, has minCardinality $k$, then $I \vDash \sigma$ iff, for any $a \subseteq d^I$, $card(\{ b \in U[I] \,/\, (a,b) \in p^I \}) \geq k$
- if $\sigma$ declares that a property $p$, with domain $d$, has maxCardinality $k$, then

  $I \vDash \sigma$ iff, for any $a \subseteq d^I$, $card(\{ b \in U[I] \,/\, (a,b) \in p^I \}) \leq k$
- if $\sigma$ declares that a class $s$ is a superclass of $c$, then

$$I \vDash \sigma \text{ iff } c^I \subseteq s^I$$

We now turn to the semantics of rules. A *valuation* for the set of variables $\boldsymbol{V}$ in $I$ is a function $v$ that maps the variables in $\boldsymbol{V}$ into elements of $U[I]$.

We extend the notion of interpretation to rule bodies as follows. We first define when a rule body $B$ is *true in I for v*, denoted $I,v \vDash B$, inductively as follows:

- if $B$ is of the form $c(x)$ then $I,v \vDash B$ iff $v(x) \in c^I$
- if $B$ is of the form $p(x,y)$ then $I,v \vDash B$ iff $(v(x),v(y)) \in p^I$
- if $B$ is of the form $C,D$ then $I,v \vDash B$ iff $I,v \vDash C$ and $I,v \vDash D$
- if $B$ is of the form $C;D$ then $I,v \vDash B$ iff $I,v \vDash C$ or $I,v \vDash D$

The *interpretation* of a rule body of the form $B[x]$ in $I$ is the set

$$B[x]^I = \{\, a \in U[I] \text{ / there is a valuation } v \text{ for } \boldsymbol{V} \text{ in } I \text{ such that } I,v \vDash B[x] \text{ and } v(x)=a \,\}$$

and the *interpretation* of a rule body of the form $B[x,y]$ in $I$ is the binary relation

$$B[x,y]^I = \{\, (a,b) \in U[I] \times U[I] \text{ / there is a valuation } v \text{ for } \boldsymbol{V} \text{ in } I \text{ such that } I,v \vDash B[x,y] \text{ and } v(x)=a \text{ and } v(y)=b \,\}$$

Finally, we say that a set $I$ of triples of $S$ is *consistent* iff $I$ satisfies all constraints of $S$.

We will use the above definitions in Section 4.3.4 to discuss the consistency of concept mappings induced by vocabulary matchings, a notion we define in the next section.

## 4.3.3. Concept mappings induced by vocabulary matchings

Given an OWL schema, we say that a class $f$ *dominates* a class $c$ or the intersection $c = d \sqcap e$ of two classes $d$ and $e$ iff there is a sequence $(c_1,c_2,...,c_n)$ such that

- $f=c_1$ and $c=c_n$
- $c_{n-1}$ subsumes $c_n$

- for each $i \in [1,n-2)$, either
  - $c_{i+1}$ and $c_i$ are classes and $c_{i+1}$ is declared as a subclass of $c_i$, or
  - $c_{i+1}$ is a class, $c_i$ is an object property and $c_{i+1}$ is declared as the range of $c_i$, or
  - $c_{i+1}$ is an object property, $c_i$ is a class and $c_i$ is declared as the domain of $c_{i+1}$

We also say that $\pi=(c_1,c_2,...,c_n)$ is a *dominance path from c to d* and $\theta=(p_{k_1}, p_{k_2},..., p_{k_m})$, the subsequence of $\pi$ consisting of the object properties that occur in $\pi$, is the *property path corresponding to* $\pi$ (note that $\theta$ may be the empty sequence).

Let $S$ and $T$ be two (OWL Extralite) schemas in what follows. Recall that a contextualized vocabulary matching between $S$ and $T$ is a finite set $\mu_V$ of quadruples $(v_1,e_1,v_2,e_2)$.

A contextualized vocabulary matching $\mu_V$ from $S$ into $T$ is *structurally correct* iff, for all $(v_1,e_1,v_2,e_2) \in \mu_V$ such that $v_1$ and $v_2$ are properties:

(i) there is a class $f$ of $S$ such that $\mu_V$ matches $f$ with the domain of $v_2$ and $f$ dominates $d_1 \sqcap e_1$, where $d_1$ is the domain of $v_1$

(ii) if $v_1$ is a datatype property, then the range of $v_1$ is a subtype of the range of $v_2$

(iii) if $v_1$ is an object property, then $\mu_V$ matches the range of $v_1$ with the range of $v_2$

Let $\mu_V$ be a structurally correct contextualized vocabulary matching. A *concept mapping M from S into T induced by* $\mu_V$ is a set of rules derived from the quadruples of $\mu_V$ as follows.

For each quadruple $(v_1,e_1,v_2,e_2) \in \mu_V$, the concept mapping $M$ contains the following rules:

*Case 1:* $v_1$ and $v_2$ be classes. Then, $M$ contains rules of the form

$$v_2(x) \leftarrow v_1(x)$$

$s(x) \leftarrow v_1(x)$ for each superclass $s$ of $v_2$

*Case 2:* $v_1$ and $v_2$ are properties. Let $d_1$ and $d_2$ be the domains, and $r_1$ and $r_2$ be the ranges of $v_1$ and $v_2$ (recall that $r_1$ and $r_2$ are XML Schema data types, if $v_1$ and $v_2$ are datatype properties, and that $\mu_V$ matches the range of $v_1$ with the range of $v_2$, if $v_1$ and $v_2$ are object properties).

*Case 2.1:* $\mu_V$ matches $d_1$ with $d_2$. Then, $M$ contains a rule of the form

$v_2(x,y) \leftarrow v_1(x,y), e_1(x)$

*Case 2.2:* $\mu_V$ does not match $d_1$ with $d_2$. Let $f$ be a class of $S$ such that $\mu_V$ matches $f$ with $d_2$ and $f$ dominates $d_1 \sqcap e_1$. Let $p_{k_1}, p_{k_2}, ..., p_{k_m}$ be the property path corresponding to a dominance path from $f$ to $d_1 \sqcap e_1$. Then, $M$ contains a rule of the form

$$v_2(x,y) \leftarrow p_{k_1}(x, x_1), p_{k_2}(x_1, x_2), ..., p_{k_m}(x_{m-1}, z), v_1(z,y), e_1(z)$$

if the property path is nonempty; otherwise the rule reduces to that of case 2.1. (Note that, since $\mu_V$ is structurally correct, a dominance path from $f$ to $d_1$ indeed exists. Also note that, since the dominance path may not be unique, the concept mapping induced by $\mu_V$ is not unique).

Note that the contextualized vocabulary matching $\mu_V$ may have more than one quadruple for the same concept $v_2$ of the target schema, which implies that the above process may generate more than one rule for $v_2$. In addition, $v_2$ may be a superclass of more than one class, which again implies that the above process, by Case 1, may generate more than one rule for $v_2$. Therefore, as a last step in the construction of the concept mapping $M$, we collect together all rules for $v_2$ as a single rule with a disjunctive body. More precisely, if $v_2$ is a class and the above process generates rules

$$v_2(x) \leftarrow B_i[x], \text{ for } i \in [1,n]$$

then we replace all such rules by a single rule $\rho$ of the form

$$v_2(x) \leftarrow B_1[x] \; ; \; ... \; ; \; B_n[x]$$

and likewise, if $v_2$ is a property.

We say that a rule $\rho$ in *M defines* a concept $v_2$ of *T* iff the head of $\rho$ is of the form $v_2(x)$, if $v_2$ is a class, or of the form $v_2(x,y)$, if $v_2$ is a property (by the transformation described above, *M* has at most one rule for each concept of *T*).

However, there might be a concept $v_2$ of *T* such that *M* has no rule that defines $v_2$. We therefore define *T/M* as the subset of *T* restricted to the concepts that *M* defines. Then, the constraints of *T/M* are the constraints of *T* defined over such vocabulary. In particular, we can prove that the superclasses, domains and ranges are properly defined in *T/M*.

**Proposition 1**: Let $\mu_V$ be a structurally correct contextualized vocabulary matching and $M$ be a concept mapping from *S* into *T* induced by $\mu_V$. Then:

(i) for any class *c* of *T/M*, if *s* is a superclass of *c* in *T*, then *s* is also a class of *T/M*.

(ii) for any property *p* of *T/M*, the domain of *p* is also a concept of *T/M*.

(iii) for any object property *p* of *T/M*, the range of *p* is also a concept of *T/M*.

**Proof**

(i) Let *c* be a class of *T/M* and *s* be a superclass of *c* in *T*. Since *c* is a class of *T/M*, by Case 1 of the construction of *M*, there is a rule in *M* of the form $c(x) \leftarrow p_1(x)$. Since *s* is a superclass of *c*, again by Case 1, there is a rule in M of the form $s(x) \leftarrow p_1(x)$. Hence, *s* is defined in *M*, that is, *s* is a class of *T/M*.

(ii) Let *p* be a property of *T/M*. Let *d* be domain of *p*. Since *p* is a property of *T/M*, by Case 2, there is a rule in *M* of the form $p(x,y) \leftarrow B[x,y]$ and a class *f* of *S* such that $\mu_V$ matches *f* with the domain *d* of *p*. Then, by Case 1, there is a rule in *M* of the form $d(x) \leftarrow f(x)$. Hence, *d* is defined in *M*, that is, *d* is a class of *T/M*.

(iii) Let *p* be an object property of *T/M*. Let *r* be the range of *p*. Since *p* is an object property of *T/M*, by Case 2, there is a rule in *M* of the form $p(x,y) \leftarrow B[x,y]$ and a class *g* of *S* such that $\mu_V$ matches *g* with the range *r* of *p*. Then, by Case 1, there is a rule in *M* of the form $r(x) \leftarrow g(x)$. Hence, *r* is defined in *M*, that is, *r* is a class of *T/M*.

**Corollary 1**: *T/M* is a well-defined OWL Extralite schema.

Finally, we define the function $\overline{M}$ *induced by M* as the mapping from sets of triples of *S* into sets of triples of *T/M* such that, for each set of triples *I* of *S*, *J*

$= \overline{M}(I)$ iff, for each rule $\rho$ in $M$

- if $\rho$ is of the form $c(x) \leftarrow B[x]$, then $J$ contains a triple $(i,$`:type`$,c)$ iff $i \in B[x]^I$
- if $\rho$ is of the form $p(x,y) \leftarrow B[x,y]$, then $J$ contains a triple $(i,p,j)$ iff $(i,j) \in B[x,y]^I$

We stress that $M$ is used to map queries submitted to the target schema $T$ into queries of the source schema $S$, whereas $\overline{M}$ is a theoretical device to prove the correctness of the concept mapping, as discussed in the next section.

### 4.3.4. Consistent concept mappings

We denote the *minCardinality* and the *maxCardinality* of a property $p$ by $mC[p]$ and $MC[p]$, respectively. By convention, we take $mC[p]=0$ (and $MC[p]=\infty$), if *minCardinality* (or *maxCardinality*) is not declared for $p$.

A property $q$ is *no less constrained* than a property $p$ iff

- $mC[p] \leq mC[q]$ and
- $MC[p] \geq MC[q]$ and
- if $p$ is declared as inverse functional then so is $q$

Note that the above definition applies even if $p$ and $q$ are from different schemas.

In what follows, let $S$ and $T$ be two (OWL Extralite) schemas, $\mu_V$ be a structurally correct contextualized vocabulary matching from $S$ into $T$, $M$ be a concept mapping from $S$ into $T$ induced by $\mu_V$ and $\overline{M}$ be the function induced by $M$.

Let $\rho$ be a rule in $M$ of the form $p(x,y) \leftarrow B[x,y]$. Recall that $p$ is a property of $T$ and all classes and properties that occur in $B[x,y]$ belong to $S$. We introduce, by definition, a property of $S$, denoted $prop[B]$, whose semantics is $prop[B]^I \equiv B[x,y]^I$, for each set $I$ of triples of $S$. We say that $\rho$ is *correct* iff $prop[B]$ is no less constrained than $p$.

We say that $M$ is *correct* iff any rule in $M$ of the form $p(x,y) \leftarrow B[x,y]$ is correct. We note that this definition takes advantage of the fact that, for each concept $c$ of $T$, the concept mapping $M$ has at most one rule (possibly with a disjunctive body) that defines *c*.

We then say that *M* is *consistent* iff, if *I* is a consistent set of triples of *S*, then any constraint of *T/M* is true in $J=\overline{M}(I)$.

**Lemma 1**: Let $\mu_V$ be a structurally correct contextualized vocabulary matching and $M$ be a concept mapping from *S* into *T* induced by $\mu_V$. Assume that $M$ is correct. Then, $M$ is consistent.

**Proof**

Let $\mu_V$ be a structurally correct contextualized vocabulary matching and $M$ be a concept mapping from *S* into *T* induced by $\mu_V$. Assume that $M$ is correct. Let *I* be a set of triples of *S*, and $J = \overline{M}(I)$. Assume that *I* is consistent. We have to prove that any constraint of *T/M* is true in *J*.

Let σ be a constraint of *T/M*. There are five cases to consider

**Case 1:** Constraint σ declares that a property $p_2$ of *T/M* has domain $d_2$ and range $r_2$.

We have to prove that $p_2^J \subseteq d_2^J \times r_2^J$. Let $(i,j) \in p_2^J$. By the construction of *M*, since $p_2$ be a property of *T/M*, there is a property $p_1$ of *S*, with domain $d_1$ and range $r_1$, such that there is $(p_1,e_1,p_2,e_2) \in \mu_V$. Let $\pi$ be the rule $p_2(x,y) \leftarrow B[x,y]$ in *M* that defines $p_2$. Since $(i,j) \in p_2^J$, by construction of *J*, we have that $(i,j) \in B[x,y]^I$. Recall that $B[x,y]$ is a disjunction of the form "$B_1[x,y];...;B_n[x,y]$". Assume that $(i,j) \in B_k[x,y]^I$.

There are two cases to consider for $B_k[x,y]$, corresponding to Cases 2.1 and 2.2 of the construction of *M*.

**Case 1.1**: Case 2.1 of the construction of *M* applies.

Then, $\mu_V$ matches $d_1$ with $d_2$. In this case, $B_k[x,y]$ is a conjunction of the form "$p_1(x,y), e_1(x)$". Since $(i,j) \in B_k[x,y]^I$, we have that $(i,j) \in p_1^I$, $i \in d_1^I$ and $j \in r_1^I$, since $d_1$ and $r_1$ are the domain and range of $p_1$. Since $\mu_V$ matches $d_1$ with $d_2$, by Case 1 of the construction of *M*, there is a rule $\delta$ in *M* of the form $d_2(x) \leftarrow d_1(x)$. Hence, by construction of *J* and since $i \in d_1^I$, we have that $i \in d_2^J$.

**Case 1.1.1**: $p_2$ is a datatype property of *T/M*.

Since $\mu_V$ is structurally correct, $r_1$ is a subtype of $r_2$. Hence, $j \in r_1^I$ implies that

$j \in r_2^J$.

**Case 1.1.2**: $p_2$ is an object property of $T/M$.

Since $\mu_V$ is structurally correct, $\mu_V$ matches $r_1$ with $r_2$. Hence, by Case 1 of the construction of $M$, there is a rule $\rho$ in $M$ of the form $r_2(x) \leftarrow r_1(x)$. Hence, by construction of $J$ and since $j \in r_1^I$, we have that $j \in r_2^J$.

**Case 1.2**: Case 2.2 of the construction of $M$ applies.

Then, $\mu_V$ does not match $d_1$ with $d_2$ and there is a class $f$ of $S$ such that $\mu_V$ matches $f$ with $d_2$ and $f$ dominates $d_1 \sqcap e_1$. In this case, $B_k[x,y]$ is a conjunction of the form

$$p_{k_1}(x,x_1), p_{k_2}(x_1,x_2),..., p_{k_m}(x_{k_{m-1}},z), p_1(z,y), e_1(z)$$

where $p_{k_1}, p_{k_2},..., p_{k_m}$ is the property path corresponding to a dominance path from $f$ to $d_1 \sqcap e_1$. Since $(i,j) \in B_k[x,y]^I$, there are $i_1,...,i_{m-1},a$ such that

$$(i,i_1) \in p_{k_1}^I, (i_1,i_2) \in p_{k_2}^I,...,(i_{m-1},a) \in p_{k_m}^I, (a,j) \in p_1^I$$

Since $f$ dominates $d_1 \sqcap e_1$ and $p_{k_1}, p_{k_2},..., p_{k_m}$ is the property path corresponding to a dominance path $\Pi$ from $f$ to $d_1 \sqcap u_1$, by definition of dominance path, there is a non-empty prefix $f_1,...,f_t$ of $\Pi$ such that $f_1 = f$, $f_t = d_{k_1}$ and $f_t \sqsubseteq f_{t-1} \sqsubseteq ... \sqsubseteq f_1$, where $d_{k_1}$ is the domain of $p_{k_1}$. Hence, $(i,i_1) \in p_{k_1}^I$ implies $i \in d_{k_1}^I$, which in turn implies that $i \in f^I$. But $\mu_V$ matches $f$ with $d_2$. By Case 1 of the construction of $M$, there is a rule $\delta$ in $M$ of the form $d_2(x) \leftarrow f(x)$. Hence, by construction of $J$ and since $i \in f^I$, we have that $i \in d_2^J$.

**Case 1.2.1**: $p_2$ is a datatype property of $T/M$.

Since $\mu_V$ is structurally correct, $r_1$ is a subtype of $r_2$. Hence, $j \in r_1^I$ implies that $j \in r_2^J$.

**Case 1.2.2**: $p_2$ is an object property of $T/M$.

Since $\mu_V$ is structurally correct, $\mu_V$ matches $r_1$ with $r_2$. Hence, by Case 1 of the construction of $M$, there is a rule $\rho$ in $M$ of the form $r_2(x) \leftarrow r_1(x)$. Hence, by construction of $J$ and since $(a,j) \in p_1^I$ implies $j \in r_1^I$, we have that $j \in r_2^J$.

Therefore, in both Cases 1.1 and 1.2, we have that $(i,j)\in p_2^J$ implies that $i\in d_2^J$ and $j\in r_2^J$.

**Case 2**: σ declares that *p* is inverse functional.

Let $\pi$ be the rule in *M* that defines *p* and assume that $\pi$ is of the form *p(x,y)←B[x,y]*. Since *M* is correct, *p(x,y)←B[x,y]* is also correct, which means that *prop[B]* is no less constrained than *p*. Then, *B[x,y]* must also be inverse functional. Since *I* is consistent, if there are $(i,j)\in prop[B]^I$ and $(g,j)\in prop[B]^I$, then *i=g*. Since, by construction of *J*, $(i,j)\in p^J$ iff $(i,j)\in prop[B]^I$, if there are $(i,j)\in p^J$ and $(g,j)\in p^J$, then *i=g*, which means that σ is true in *J*.

**Case 3**: σ declares that *p* has minCardinality equal to *k* (that is, *mC[p] = k*).

Let $\pi$ be the rule in *M* that defines *p* and assume that $\pi$ is of the form *p(x,y)←B[x,y]*. Since *M* is correct, *p(x,y)←B[x,y]* is also correct, which means that *prop[B]* is no less constrained than *p*. Then, $mC[p] = k \leq mC[prop[B]]$. Since *I* is consistent, $k \leq mC[prop[B]]$ implies that, if there is $(i,j)\in prop[B]^I$, there are at least *k* pairs $(i,j_1),...,(i,j_k)\in prop[B]^I$. Since, by construction of *J*, $(i,j)\in p^J$ iff $(i,j)\in prop[B]^I$, if there is $(i,j)\in p^J$, there are at least *k* pairs $(i,j_1),...,(i,j_k)\in p^J$, which means that σ is true in *J*.

**Case 4**: σ declares that maxCardinality equal to *k* (that is, *MC[p] = k*).

(Follows as in Case 3).

**Case 5**: σ declares that *s* is a superclass of *c*.

Let $\pi$ be the rule in *M* that defines *c* and assume that $\pi$ is of the form *c(x)←B[x]*. Let $i\in c^J$. Then, by construction of *J*, $i\in B[x]^I$. We may assume without loss of generality that $\pi$ is of the form *c(x)←...;d(x);....* and that $i\in d^I$. Since *s* is a superclass of *c*, by Case 1 of the construction of *M*, there is also a rule $\rho$ in *M* of the form *s(x)←...;d(x);.....* Then, by construction of *J*, and since $i\in d^I$, we have that $i\in s^J$.

Therefore, $i\in c^J$ implies that $i\in s^J$, which means that σ is true in *J*. □

Lemma 1 is not entirely helpful, though, since it is not entirely obvious how to test if a rule of the form *p(x,y)←B[x,y]* is correct, i.e, if *prop[B]* is no less constrained than *p*. Proposition 1 below provides sufficient conditions guaranteeing correctness of one such rule.

**Proposition 2**: Let $\rho$ be a rule in $M$ of the form *p(x,y)←B[x,y]*. Assume that *B[x,y]* is of the form $B_1[x,y];...;B_n[x,y]$. Let $p_{i,1}, p_{i,2},..., p_{i,m_i}$ be the properties of *S* that occur in $B_i[x,y]$, for $i \in [1,n]$. Then,

(i) $mC[prop[B[x,y]]] \geq min\{ \prod_{j=1}^{m_i} mC[\, p_{i,j}\,] \,/\, i = 1,...n \}$

(ii) $MC[prop[B[x,y]]] \leq \sum_{i=1}^{n} \prod_{j=1}^{m_i} MC[\, p_{i,j}\,]$

(iii) If all properties that occur in *B[x,y]* are inverse functional and $prop[B_1[x,y]],...,prop[B_n[x,y]]$ are pairwise disjoint, then *prop[B[x,y]]* is inverse functional.

(iv) If all properties that occur in *B[x,y]* are inverse functional and *B[x,y]* has just one conjunction (that is, $n=1$), then *prop[B[x,y]]* is inverse functional.

**Proof**

Let $\rho$ be a rule in $M$ of the form *p(x,y)←B[x,y]*. Assume that *B[x,y]* is of the form $B_1[x,y];...;B_n[x,y]$. Let $p_{i,1}, p_{i,2},..., p_{i,m_i}$ be the properties of *S* that occur in $B_i[x,y]$, for $i \in [1,n]$. We first prove that

(1) $mC[prop[B_i[x,y]]] = \prod_{j=1}^{m_i} mC[\, p_{i,j}\,]$

(2) $MC[prop[B_i[x,y]]] = \prod_{j=1}^{m_i} MC[\, p_{i,j}\,]$

There are two cases to consider for $B_i[x,y]$, corresponding to Cases 2.1 and 2.2 of the construction of *M*.

**Case 1**: Case 2.1 of the construction of *M* applies.

Then, $B_i[x,y]$ is a conjunction of the form

$$p_{i,1}(x,y), e_i(x)$$

Since $e_i(x)$ restricts the domain of $p_i$, but not the range, we trivially have that

(3) $mC[prop[B_i[x,y]] =$

$$mC[prop[\,p_{i,1}(z,y),e_i(z)]] = mC[prop[\,p_{i,1}(z,y)]] = mC[\,p_{i,1}\,]$$

(4) $MC[prop[B_i[x,y]] = MC[prop[\,p_{i,1}(z,y),e_i(z)]] = MC[\,p_{i,j}\,]$

**Case 2**: Case 2.2 of the construction of *M* applies.

Then, $B_i[x,y]$ is a conjunction of the form

$$p_{i,1}(x,x_1),...,p_{i,m_i-1}(x_{m_i-2},z),p_{i,m_i}(z,y),e_i(z)$$

Since a conjunction $C[x,y]$ of the form "$q_1(x,z),q_2(z,y)$" captures the composition of two properties $q_1$ and $q_2$, by definition of minCardinality, we have that

(5) $mC[\,prop[q_1(x,z),q_2(z,y)]] = mC[\,prop[\,q_1(x,z)]]\,.\,mC[\,prop[\,q_2(z,y)]] =$
$mC[q_1\,].mC[q_2\,]$

Again, since $e_i(z)$ restricts the domain of $p_{i,m_i}$, but not the range, we have that

(6) $mC[\,prop[\,p_{i,m_i}(z,y),e_i(z)]] = mC[\,prop[\,p_{i,m_i}(z,y)]] = mC[\,p_{i,m_i}\,]$

By repeatedly applying (5) and using (6) at the last step, we have that

(7) $mC[prop[B_i[x,y]]] =$

$$mC[\,prop[\,p_{i,1}(x,x_1),...,p_{i,m_i-1}(x_{m_i-2},z),p_{i,m_i}(z,y),e_i(z)]] =$$
$$mC[\,p_{i,1}\,]\,.\,mC[\,p_{i,2}\,]\,.\,...\,.\,mC[\,p_{i,m_i-2}\,]\,.\,mC[\,p_{i,m_i-1}\,]\,.\,mC[\,p_{i,m_i}\,] =$$
$$\prod_{j=1}^{m_i} mC[\,p_{i,j}\,]$$

Using a similar argument and the definition of maxCardinality, we have that

(8) $MC[prop[B_i[x,y]]] =$

$$MC[\,prop[\,p_{i,1}(x,x_1),...,p_{i,m_i-1}(x_{i,m_i-2},z),p_{i,m_i}(z,y),e_i(z)]] =$$
$$MC[\,p_{i,1}\,]\,.\,MC[\,p_{i,2}\,]\,.\,...\,.\,MC[\,p_{i,m_i-2}\,]\,.\,MC[\,p_{i,m_i-1}\,]\,.\,MC[\,p_{i,m_i}\,] =$$
$$\prod_{j=1}^{m_i} MC[\,p_{i,j}\,]$$

We now prove (i) and (ii).

Recall that, since that *B[x,y]* is a disjunction of the form $B_1[x,y];...;B_n[x,y]$, for any interpretation *I* of *S*, we have that

(9) $B_i[x,y]^I \subseteq B[x,y]^I$, for any $i \in [1,n]$

From (9), by definition of minCardinality, we have that

(10) $mC[prop[B[x,y]]] \geq mC[\ prop[\ B_i\ [x,y]]]$, for any $i \in [1,n]$

Hence, from (10) and (1), we have that:

$$(11)\ mC[prop[B[x,y]]] \geq min\{\ mC[prop[B_i\ [x,y]]] \,/\, i = 1,\ldots n\ \} = \\ min\{\ \textstyle\prod_{j=1}^{m_i} mC[\ p_{i,j}\ ] \,/\, i = 1,\ldots n\ \}$$

Also, for any interpretation $I$ of $S$, we have that

(12) $B[x,y]^I = B_1[x,y]^I \cup \ldots \cup B_n[x,y]^I$

From (12), by definition of maxCardinality, we have that

(13) $MC[prop[B[x,y]]] \leq \sum_{i=1}^{n} MC[\ prop[\ B_i\ [x,y]]]$

Hence, from (13) and (2), we have that:

(14) $MC[prop[B[x,y]]] \leq \sum_{i=1}^{n} MC[\ prop[\ B_i\ [x,y]]] = \sum_{i=1}^{n} \prod_{j=1}^{m_i} mC[\ p_{i,j}\ ]$

Note that (iv) directly follows from (iii). So, we only prove (iii). Assume that

(15) all properties that occur in $B[x,y]$ are inverse functional
(16) $prop[B_1[x,y]],\ldots,prop[B_n[x,y]]$ are pairwise disjoint

using an argument similar to that of Cases 1 and 2 above for (i) and (ii), by (15), we have that

(17) $B_i[x,y]$ is inverse functional

From (12) and (17), by (16), we have that $B[x,y]$ is inverse functional.

Note that, in (iii), we cannot establish that $prop[B_1[x,y]],\ldots,prop[B_n[x,y]]$ are pairwise disjoint within OWL Extralite, since this dialect of OWL does not support class or property disjointness (hence the simplified statement in (iv)).

We may combine Lemma 1 and Proposition 1 and list sufficient conditions for consistency.

**Corollary 2**: Let $\mu_V$ be a structurally correct contextualized vocabulary matching and $M$ be a concept mapping from *S* into *T* induced by $\mu_V$. Then, $M$ is consistent if, for any rule in $M$ of the form *p(x,y)←B[x,y]*, we have:

(i) $\min\{\prod_{j=1}^{m_i} mC[\,p_{i,j}\,] \,/\, i = 1,...n\} \geq mC[p]$

(ii) $\sum_{i=1}^{n} \prod_{j=1}^{m_i} MC[\,p_{i,j}\,] \leq MC[p]$

(iii) If *p* is inverse functional, then all properties that occur in *B[x,y]* are inverse functional and either $prop[B_1[x,y]],...,prop[B_n[x,y]]$ are pairwise disjoint, or *B[x,y]* has just one conjunction (that is, *n=1*)

where *B[x,y]* is of the form $B_1[x,y];...;B_n[x,y]$ and $p_{i,1}, p_{i,2},..., p_{i,m_i}$ are the properties of *S* that occur in $B_i[x,y]$, for $i \in [1,n]$.

## 4.4. Summary and contributions

In this chapter we focused on the more complex problem of matching two schemas that belong to an expressive OWL dialect. The matching technique is based on the notion of similarity. We decomposed the problem of OWL schema matching into the problem of vocabulary matching and the problem of concept mapping. We also introduced sufficient conditions guaranteeing that a vocabulary matching induces a correct concept mapping.

We developed a similarity function based on the contrast model (Tversky and Gati 1978), which proved to efficiently capture the notion of similarity, and described heuristics that lead to practical OWL matchings.

We introduced the OWL Extralite dialect because it is as expressive as UML and, yet, it avoids the complex constructions of OWL Full, such as subproperties and multiple inheritances.

Unlike any of the instance-based techniques previously defined, the OWL schema matching process, we proposed to use similarity functions to induce vocabulary matchings in a non-trivial way.

Contrasting with (Doan et al. 2001, Madhavan et al. 2005), we do not use machine learning techniques to acquire knowledge about matchings. Instead, we capture semantic similarity by adopting similarity functions and heuristics that depend on the schema concepts. We consider this strategy to be more general because it can identify matching candidates which were not in the training corpus.

Contrasting with (Brauner et al. 2007b, Wang et al. 2004), which measure the similarity between concepts only by the commonalities between sets of values, we use similarity functions that take into account not only the commonalities, but also the differences between concepts. Such models proved to increase the precision of the matching process. In addition, we use similarity heuristics that operate at the level of data values, that is, they permit comparing data values based on their similarity, and not just on their exact equality.

We overcome the limitation of representing an instance by a string constructed out of all its property values, introduced in (Bilke and Naumann 2005), by representing an instance by a string constructed out of the values only of those properties that match, in a first approximation.

In summary, unlike the techniques listed in Section 1.2, we proposed hybrid matching techniques that are uniformly grounded on similarity functions to generate matchings between simple catalogue schemas and between more complex OWL schemas.