

Semantic Web Complex Ontology Mapping

Nuno Silva and João Rocha

GECAD - Knowledge Engineering and Decision Support Research Group

Instituto Superior de Engenharia do Porto

4200-072 Porto – Portugal

Nuno.Silva@dei.isep.ipp.pt; jrocha@ipp.pt

Abstract

Ontology mapping is the process whereby semantic relations are defined between two ontologies at conceptual level which in turn are applied at data level transforming source ontology instances into target ontology instances. Ontology mapping faces new challenges in the context of Semantic Web, especially concerning heterogeneity, dynamics, distribution and limitations on representation technology. The work described in this work runs in scope of MAFRA – Mapping FRAMework, a conceptual description of the ontology mapping process. This paper though focuses on the formalization, representation, specification and execution during the ontology mapping process. The architecture of the system is based on the notion of Service, representing not only the system transformation capabilities, but also the expertise in the manipulation of specific semantic relations. In fact, the process of choosing, applying, validating, evolving or negotiating a semantic relation depends on both the ontologies contents and the transformations available in the system. This new architecture and mapping process are being tested and validated in MAFRA Toolkit, a specific implementation of MAFRA. The main contributions of this paper are the formalization of the ontology mapping process, the specification and execution methodology and the service-oriented architecture.

1 Introduction

Semantic Web, the next WWW trend, suggests the annotation of Web resources with machine-processable metadata, which can provide tools to analyse meaning and semantic relations between documents and their parts. Ontologies as means for conceptualizing and structuring knowledge are seen as the key to the realization of the Semantic Web vision. Ontology allows the explicit specification of a domain of discourse, which permits to access to and reason about an agent knowledge. Ontologies raise the level of specification of knowledge, incorporating semantics into the data, and promote its exchange in an explicit understandable form. Semantic Web and ontologies are therefore fully geared as a valuable framework for distinct applications, namely business applications like E-Commerce and B2B. However, ontologies do not overcome *per se* any interoperability problems, since it is hardly conceivable that a single ontology is applied for all kind of domains and applications. Ontology mapping does not intend to unify ontologies and their data, but to transform ontology instances according to the semantic relations (mapping relations) defined at conceptual level. Repositories are therefore kept separated, independent and distinct, maintaining their complete semantics and contents. The solution proposed in the paper adopts a declarative specification of mappings, hiding the procedural complexity of specification and execution, while its preconized open architecture allows the integration of new mapping relations into the system, improving mapping capabilities along the overall process.

This paper is divided into ten more sections. Section 2 presents the motivations and goals associated with this work. Section 3 overviews the Mapping FRAMework in order to call attention to the multiples phases that constitute the ontology mapping process, which in turn justifies many of the further described approaches and decisions. Section 4 formalises the ontology mapping process, followed in section 5 by the Semantic Bridging Ontology formalizations. Section 6 presents the service-oriented architecture of the system, describing the operationalization of the ideas proposed in prior sections. Section 7 presents some examples of ontology mapping specification, contributing to the understanding of the Semantic Bridging Ontology application. In section 8 the execution process is extensively described providing deep details of the execution phase process. The work presented in sections 4, 5, 6 and 8 represents the core contributions for the ontology mapping research field. Relevant projects in the area of ontology mapping will be described and compared, in section 9. Section 10 describes some experiences and comparison with a similar tool. Finally, section 11 will provide an overview of achievements and main contributions of the paper followed by a description of current and future efforts.

2 Motivations and Goals

2.1 Motivations

Considering ontology an explicit specification of a conceptualization [1], it encodes individual (or corporate) perspectives of the universe, leading later to integration conflicts. The distributed nature of the web amplifies this problem, and even the suggested exploitation of lexical and domain super ontologies do not overcome integration problems. In fact, living the same world does not mean every one see and understand it the same way. Additionally ontologies are typically developed for private use only, with minimal interconnection purposes. As consequence, there is no requirement to annotate or link conceptual descriptions to more abstract ontologies.

Ontology mapping is a transversal key technology for many different problems regarding data integration. Also referred as data translation, data integration is concerns with transforming one dataset described according to a certain model into another dataset respecting another model and each-one specificities and requirements. In special we distinguish between two categories of problems:

- Off-line data integration. With the advent of enterprise and organizations integration and merging, separated heterogeneous databases have to be integrated in a fast and reliable manner. In this scenario source and target information repositories are known a priori and a (long) integration phase is possible. Data migration, data warehouse clean & transform [2], data model evolution [3] and data driven portal generation [4] are variant problems of off-line data integration;
- On-line data integration. Web services are emerging as major paradigm for the semantic web implementation of business-process, promotion of B2B and E-Commerce to unprecedented level. It is necessary to develop mechanisms capable of mediation and alignment between distinct data models. Syntactical, structural and semantic heterogeneity arise in different degrees and variations between different information repositories. The ability to semi-automatically map between different knowledge bases and answer agents queries from different agents becomes of primordial importance to the success of truly dynamic and autonomous business. Negotiation and argumentation between partners becomes therefore a major requirement for ontology mapping systems.

The final motivation respects the supporting technology. In context of Semantic Web, the base ontology representation language is RDFS. RDFS stands for Resource Description Framework Schema and it is an extension to RDF, allowing the specification of domain vocabulary. Other ontology representation languages have been proposed in the scope of Semantic Web, most of them extend RDFS and inherit its primitive representation mechanisms. However, none provides minimal constructs for mapping between ontologies. Elements like *rdfs:sameAs* are clearly insufficient. Description-logics theory included in OIL, DAML and OWL provide useful but insufficient constructs.

2.2 Goals

An infinite number of possible mapping relations between two ontologies may exist. Consequently, ontology mapping process is not a straightforward or a mathematical problem. Thus the need for goals and quality indicators. Five driving vectors have been identified:

- **Applicability** concerns what type of mapping relations are possibly solved in the system, as well as the applicability of the proposed approach;
- **Semantic Expressivity** concerns to how explicit and described is the mapping specification;
- **Modularization** concerns with the system characteristic to be build upon the combination of small, simple modules into a more complex whole;
- **Reutilization** of components, namely concerning applying knowledge created from previous mapping experiences and its recycling when obsolete;
- **Declarativity** concerns the capabilities of the system to supply conditions to domain expert focus on semantic instead of in *how-to*. Maximizing declarativity, will improve quality and productivity, while minimizing software development and customization mistakes and therefore costs.

The ontology mapping system must maximise these driving vectors. Additionally, it is important to notice that this work grounds in Semantic Web technology and must primarily provide technology for the Semantic Web. In special, we consider:

- Ontologies as a domain description modelled in RDFS or any language that can be grounded to RDFS;
- The distributed and ever-evolving nature of semantic web, namely respecting distributed ontologies and its unpredictable evolution;
- The ambiguous nature of conceptualizations, namely due to the private (or corporate specification of ontologies);
- The incomplete nature of semantic web, namely respecting the description of models, constraints and data.

The following sections will consider this constraints and drivers in the analysis, specification and development of the ontology mapping system.

3 MAFRA – Mapping FRamework

According to the generic description presented in Introduction, the ontology mapping process is a two phases process: specification and execution, but not only. MAFRA framework, firstly presented in [5] aims to cover all requirements introduced in section 2. The resulting framework is outlined in Figure 1.

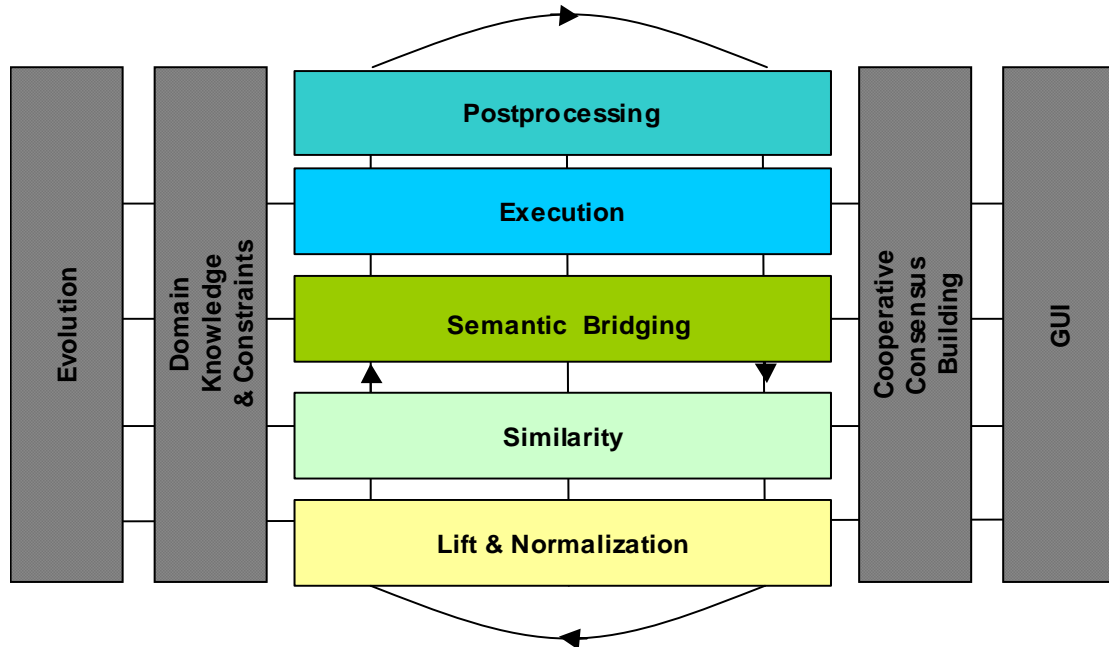


Figure 1 – MAFRA Conceptual Framework

The framework identifies the phases commonly found in any ontology mapping process through the horizontal components (i.e. Lift & Normalization, Similarity Measuring, Semantic Bridging, Execution and Post-processing). Simultaneously, it identifies and organises a set of modules that are not part of the process, but represent crucial functionalities existing in any system, providing (automatic) support along the entire ontology mapping process. While the five process phases can be executed either in manual or automatic manner, the four vertical modules are part of the system to automate or assist the domain expert during the entire process. The framework suggests a cyclic, iterative and interactive operation, promoting the continuous improvement and recycling of the mapping.

3.1 Horizontal Dimension of MAFRA

Within the horizontal dimension, we identified the following five modules corresponding to five process phases:

Lift & Normalization focuses on coping with language and lexical heterogeneity [6] between source and target ontology. Language lift focuses on minimizing syntactical and structural representation heterogeneity, while lexical normalization focuses on minimizing ontology contents lexical heterogeneity. Lexical normalization is responsible for the normalization of abbreviations and acronyms. It acquires special relevance in the Similarity Measuring.

Similarity Measuring aims to acquire, infer and establish similarities measures between source ontology entities and target ontology entities. Similarities might be explicitly defined by domain expert, or automatically acquired through complementary mechanisms. Different automatic approaches exists depending on operating context and other (self-imposed) constraints [7;8], but most of them exploit external knowledge sources like background domain knowledge bases, thesaurus and dictionaries.

Semantic Bridging phase establishes the semantic relations between set of source and target entities and defines the transformation function relating the sets. Again, the process associated with this phase may be manual or (semi-) automatic. In manual approach, the prior phase is not mandatory, and similarities are implicit in domain expert knowledge. In an automatic process, the similarities measures resulting from prior phase are exploited in both (i) grouping semantic similar entities and (ii) in inferring the transformation functions to apply for each set. Most of the automatic mapping approaches [7] provide support only for 1:1 (one source entity relates to one target entity) mappings. The automation of more complex relations such 1:n and n:1 are of most important requirement. The resulting of this phase is an ontology mapping document.

Execution phase transforms instances from the source ontology into target ontology instances by evaluating the ontology mapping document defined in previous phase. Any ontology mapping system makes sense only if this phase is automatic. In general two distinct modes of operation are possible, namely offline (static, one-time transformation) and online (dynamic, continuous mapping between source and the target) execution depending on operation environment and requirements. Query-based execution is a variant of the online operation, in which the system replies according to the queries.

Post-processing takes the results of the execution module to check and improve the quality of the transformation results. The most challenging task of post-processing is establishing object identity - recognizing that two (or more) instances represent the same real-world object, and recognize that two (or more) similar entities represent two different objects.

3.2 Vertical Dimension of MAFRA

In the vertical dimension of MAFRA four complementary modules have been identified:

Evolution. The evolution module focuses on maintenance of ontology mapping document, which is due to at least two reasons: (i) synchronise with the changes in the source and target ontologies, and (ii) changes in domain or application requirements. While research on evolution of ontology mapping is missing, extensive research exist concerning evolution of ontologies in the context of the Semantic Web [9;10]. This research is a valuable starting point for mapping evolution.

Cooperative Consensus Building. Semantic clashes easily arise between heterogeneous and autonomous entities populating the Semantic Web. Additionally, due to ontology mapping evolution, choosing the correct ontology mapping (version) may arise difficulties. Though the automatic agreements are hard to achieve, the cooperative consensus-building module assists and promotes negotiation between the two knowledge communities involved in the process. Research in the area of meaning negotiation [11;12] is of major importance for this module development.

Domain Constraints and Background Knowledge. The quality of similarity computation and semantic bridging may be dramatically improved by introducing background knowledge and domain constraints, e.g. by using glossaries to help identify synonyms or by using lexical ontologies, such as WordNet or domain-specific thesauri, to identify similar concepts.

Graphical User Interface. Mapping is a difficult and time-consuming process, which is not less difficult than building an ontology itself. Ontology mapping process require deeply understand of both ontology conceptualizations and their semantic similarities. Special difficulties arise during the specification phase of the process, which require human expertise. Browsing structure of both ontologies, definition and customization of semantic relations between sets of entities require extensive support. Graphical user interface arises therefore as a fundamental tool in the system, allowing and promoting better ontology mappings.

4 Ontology Mapping Problem

Ontology mapping as defined in scope of this paper is the process whereby semantic relations are defined at ontological level between source ontology entities and target ontology entities; and are further applied at instance level transforming source ontology instances into target ontology instances. Ontology mapping contains therefore two phases.

The first phase, named (semantic bridging) specification phase, occurs at meta-level, in the sense that the object of process are the domains at instance-level. This phase is formally defined as a function between the source and target ontology elements:

$$\mathcal{M}: \mathcal{O}_s \rightarrow \mathcal{O}_t$$

\mathcal{M} is an **ontology mapping document**, or simply ontology mapping, which contains the necessary and sufficient information required in second phase. The ontology mapping document will be further described in section 5.

The second phase, named (semantic bridging) execution phase, occurs at instance level. This phase is formally defined as the function between source knowledge base and target knowledge base, according to the ontology mapping document defined previously:

$$\mathcal{T}(\mathcal{M}): \mathcal{KB}_s \rightarrow \mathcal{KB}_t$$

Ontology and knowledge base are therefore the core arguments in the ontology mapping process. Because many definitions exist for ontology and knowledge base it is important to univocally define both terms.

Ontology is a tuple in the form of:

$$\mathcal{O} := (\mathcal{C}; is_a; \mathcal{P}; \sigma)$$

where

- \mathcal{C} is the set whose elements are called concepts, defined by the domain expert, plus the LITERAL construct which is responsible for encoding primitive types such strings and numbers;
- is_a is a partial order on \mathcal{C} representing the hierarchical relation between concepts (i.e., a binary relation $is_a \subseteq \mathcal{C} \times \mathcal{C}$ which is reflexive, transitive, and anti-symmetric);
- \mathcal{P} is a set whose elements are called relation names (or relations for short);
- σ is a function which assigns to each property name its arity. Because the RDF/S framework adopt a triple representation of information [13], σ is always a binary function in the form¹:

$$\sigma : \mathcal{P} \rightarrow 2^{\mathcal{C} \times \mathcal{C} \cup LITERAL}$$

The following functions are available:

$domain : \mathcal{P} \rightarrow 2^{\mathcal{C}}$, gives the set of domain concepts \mathcal{C} of property $p \in \mathcal{P}$;

$range : \mathcal{P} \rightarrow 2^{\mathcal{C} \cup LITERAL}$, gives the set of domain concepts \mathcal{C} of property $p \in \mathcal{P}$;

When the range of a property is a domain-defined concept it is said to be a Relation. When the range of a property is a primitive type it is said to be an Attribute.

Besides it is not explicitly referred in the ontology definition, **ontology entity** refers to any $\varepsilon \in \mathcal{C} \cup \mathcal{P}$, i.e. a Concept or a Property. An ontology entity represents a conceptual element of the domain of discourse.

A **knowledge base** is an instantiated ontology, which is formally defined as:

$$\mathcal{KB} := (\mathcal{O}, \mathcal{I}, inst\mathcal{C}, inst\mathcal{P})$$

where \mathcal{O} is an ontology, \mathcal{I} is a set of elements called instances, and $inst\mathcal{C}$ and $inst\mathcal{P}$ are defined as follow:

$$inst\mathcal{C} : \mathcal{C} \rightarrow 2^{\mathcal{I}}$$

$$inst\mathcal{P} : \mathcal{P} \rightarrow 2^{\mathcal{I} \times \mathcal{I}}$$

Accordingly, an **ontology instance** is any data element represented according to (and coherent with) the domain ontology.

As an example, one may declare the following knowledge base:

$$\begin{aligned} \mathcal{KB}_1 &:= (\mathcal{C}_1, \mathcal{I}_1, inst\mathcal{C}_1, inst\mathcal{P}_1) \\ \mathcal{C}_1 &:= (\mathcal{C}_1, is_a_1, \mathcal{P}_1, \sigma_1) \\ \mathcal{C}_1 &:= \{Person, Institution\} \\ is_a_1 &:= \{\} \\ \mathcal{P}_1 &:= \{has_name, works_in, in_city\} \\ \sigma_1 &:= \{has_name(Person, LITERAL), has_name(Institution, LITERAL), works_in(Person, Institution), in_city(Institution, LITERAL)\} \\ \mathcal{I}_1 &:= \{i_1, i_2, i_3, i_4, "Nuno Silva", "GECAD - ISEP", "WIM - FZI", "Porto", "Karlsruhe"\} \\ inst\mathcal{C}_1 &:= \{Person(i_1), Person(i_2), Institution(i_3), LITERAL("Nuno Silva"), LITERAL("João Rocha"), LITERAL("FZI - WIM"), \\ &LITERAL("GECAD - ISEP"), LITERAL("Porto"), LITERAL("Karlsruhe")\} \\ inst\mathcal{P}_1 &:= \{has_name(i_1, "Nuno Silva"), has_name(i_2, "João Rocha"), has_name(i_3, "GECAD - ISEP"), has_name(i_4, "FZI - WIM"), \\ &works_in(i_1, i_3), works_in(i_1, i_4), works_in(i_2, i_3)\} \end{aligned}$$

5 Semantic Bridging Ontology

SBO is the MAFRA companion respecting description and representation of ontology mappings. SBO has been previously presented in [5] using UML notation. Using an abstract, mathematical notation, this section formally presents SBO, explicitly defining its semantics and coping with misunderstandings and ambiguities. Additionally, grounding it to mathematical notation promotes syntax independence.

SBO is an ontology for ontology mapping. It specifies, classifies and describes the types of ontology mapping relations, inter-relate them and provides other modelling constructs necessary to express ontology mapping documents. Since ontologies conceptual entities are the objects to be mapped, SBO is in fact characterized as a meta-ontology for

¹ Besides the latest RDFS recommendations accept the use of XML types the current specifications consider the primitive type LITERAL only. LITERAL represent the value of any primitive type as a string.

ontology mapping. An instantiation of the SBO is an ontology mapping document, representing the semantic relations and their inter-relations.

5.1 Semantic Bridge

The explicit definition of a semantic relation, with all necessary information to be applied at execution phase is named Semantic Bridge. In SBO this concept is formally defined as:

$$\mathcal{B} := (\mathcal{E}_s, \mathcal{E}_t, T, \mathcal{A}, \mathcal{K})$$

where

- \mathcal{E}_s is a set of source ontology entities;
- \mathcal{E}_t is a set of target ontology entities;
- T is the transformation (function) to apply over the source instances to transform them into target instances;
- \mathcal{A} is a set of assertions defining semantic bridge execution behaviour independent of the attached transformation;
- \mathcal{K} is a set of conditions that must hold to execute the semantic bridge.

The ontology definition presented in section 4 clearly distinguishes between Concept and Property elements. Following the same approach, SBO semantically specializes Semantic Bridge into Concept Bridges and Property Bridges.

5.2 Concept Bridge

Concept Bridges are responsible for the transformation of concept instances. Concept bridges specialize the semantic bridge concept in two details:

1. The cardinality of a concept bridge is always 1:1. It means that in a concept bridge exactly one source concept is semantically related to exactly one target concept. In order to semantically relate the same source concept with many different target concepts, one must define many concept bridges;
2. Because the transformation process executed in transformation concept instances into target instances is always the same, it can be made implicit and therefore ignore its specification.

Accordingly, a concept bridge is formally defined as:

$$\mathcal{B}^c := (c_s, c_t, \mathcal{A}, \mathcal{K}), c_s \in \mathcal{C}_s \wedge c_t \in \mathcal{C}_t$$

However, this form do not provides the mechanisms to address the problem in semantically relating one source concept to multiple target concepts. This problem is sometime referred as a Property-to-Concept semantic relation. In practice, one single source instance will give raise to multiple target instances, causing problems in addressing the correct instance in creating inter-relations between such target instances.

Extensional specification provides the mechanism to distinguish and address between different perspectives of the same (source) instance. Such approach can be understood as the virtual creation of multiple source instances. This approach is based on Description Logics basic principles, and permits to reason upon the source ontology according to the target ontology and ontology mapping. The process associated with this approach is explained in section 8.2.

In order to adapt the formal definition introduced in section 5 to this requirement, concept bridge tuple should encompass a new element:

$$\mathcal{B}^c := (c_s, c_t, \mathcal{X}, \mathcal{A}, \mathcal{K})$$

where

\mathcal{X} is the set of extensional specifications. An extensional specification is a condition expression as defined bellow.

5.3 Property Bridge

Property Bridges transforms property instances. The cardinality of a property bridge is depends on the transformation service associated with it. Unlike concept bridges, property bridges cardinality can be 1:1, 1:n, m:1 and m:n. Moreover, the transformations occurring in property bridges differ enormously according to numerous factors, therefore the need to explicitly specify the associated transformation:

$$\mathcal{B}^p := (\mathcal{L}_s, \mathcal{L}_t, \mathcal{Q}, T, \alpha, \delta, \varphi, \mathcal{A}, \mathcal{K})$$

where

$\mathcal{L}_s, \mathcal{L}_t$ are sets of source and target Paths respectively;

\mathcal{Q} is a set of constants, which can in turn be sets of literals or simple literals;

α is a function that associates source paths with source arguments of the transformation function;

δ is a function that associates constants with source arguments of the transformation function;

φ is a function that associates target paths with target arguments of the transformation function.

5.4 Path

Paths allow query the knowledge base, whose result is a particular view over the knowledge base data. A path is a non-empty list of steps, which is a tuple in the form of:

$$s := (c, p, o, d)$$

where

- $c \in \mathcal{C}$ represents the subject of a relation. The function $SUBJECT : 2^{sxc \in \mathcal{C}}$ states whether c is or is not the subject of s ;
- $p \in \mathcal{P}$ represents the predicate of a relation, The function $PREDICATE : 2^{sxp \in \mathcal{P}}$ states whether p is or is not the predicate of s ;
- $o \in \mathcal{C} \cup LITERAL$ represents the object of a relation. The function $OBJECT : 2^{sxc \in \mathcal{C} \cup LITERAL}$ states whether o is or is not the object of s ;
- d is a boolean value defining the direction the predicate is read, i.e. “from subject” or “to subject” (see example below).

Each tuple must represent an ontology valid relation, i.e.:

$$\forall p \in \mathcal{P}, \exists c \in domain(p) \wedge o \in range(p)$$

and the subject concept of a tuple must be the same concept of the object of the previous tuple:

$$\forall c \in \mathcal{C}, L \in \mathcal{L}, s_i \in L : SUBJECT(s_{i+1}, c) \Rightarrow OBJECT(s_i, c)$$

From the knowledge base presented in section 4, the path $L_1 := [(Person, works_in, Institution, true), (Institution, in_city, LITERAL, true)]$ represents all cities in which the person works. Instantiating this path for i_1 Person instance, the result would be the view:

Person	(work_in) Institution	(in_city) LITERAL
i_1	i_3	“Porto”
i_1	i_4	“Karlsruhe”

The path $L_2 := [(Institution, works_in, Person, false), (Person, has_name, LITERAL, true)]$ represents the name of all persons working in a certain institution. When instantiating this path for i_3 Institution instance the result would be:

Institution	Person (work_in)	(has_name) LITERAL
i_3	i_1	“Nuno Silva”
i_3	i_2	“João Rocha”

5.5 Assertion

Assertions are declarations concerning the behaviour of the semantic bridge execution. Ontology mapping should adhere the best is possible to the ontology modelling language constructs. Typical ontologies modelling languages, and RDFS in particular, follow an object-oriented (OO) modelling approach. One of the constructs commonly used in OO languages, for example, is the “abstract concept”. An abstract concept serves to model different perspectives of world, but independently of the reason behind the modelling decision, abstract concepts are not allowed to have instances. Such construct assumes special relevance in the ontology mapping specification, since no instances should be created for a target abstract concept. Even if ontologies modelled in RDFS do not support such modifier, domain expert might be aware of this ontological decision and require similar feature from the ontology mapping specification language. Because other modelling constructs can be referred, the approach followed in SBO suggests a set of assertions instead of individual parameters. An assertion is a tuple in the form of:

$$A := (\text{assertion_name}, \text{value})$$

5.6 Condition Expression

Condition Expressions are attached to each semantic bridges in order to constraint their execution. Condition expressions are boolean expressions over mathematical expressions. Using the BNF notation, a condition expression is described as:

```

<condition_expression> ::= <and> | <or> | <xor> | <not> | <comparison>
<and> ::= and( <expression> { ( , <expression> ) } )
<or> ::= or( <expression> { ( , <expression> ) } )
<xor> ::= xor( <expression> { ( , <expression> ) } )
<not> ::= not( <expression> )
<comparison> ::= <PATH> <OPERATOR> ( <PATH> | <LITERAL> )

```

The <PATH> token corresponds to the Path entity described above, and their application in condition expressions follows the same constraints enumerated then. Considering the knowledge base presented in section 4, a valid condition expression would be:

$$K := \text{and}(\text{Person.works_in.Institution.has_name} = \text{"GECAD - ISEP"}, \text{Person.works_in.MaxCard} 1)$$

Typically, condition expressions orderly operate in four distinct phases of execution:

1. Semantic bridge pre-conditions, checks if the source instance conform the requirements, otherwise the semantic bridge is not executed;
2. Instance pre-conditions are checked before the transformation of each set of source instance values. If conditions hold, the property bridge is executed, otherwise a new set of source instance values is processed;
3. Instance post-condition checks requirements over each set of target instance values, immediately after the transformation but before the storage in knowledge base. If conditions hold the set of instance values are kept for further storage;
4. Semantic bridge post-conditions checks whether the target instance values conform the requirements. If conditions hold, the instance values are definitely stored in knowledge base otherwise they are discarded.

Redundant condition expressions are acceptable. Besides this is an implementation issue, validation procedures only consider the ontology the expression address. In case the condition expression is associated with a pre-condition group, paths in condition expression should address source ontology entities. If condition expression is associated with a post-conditions group, then the paths should address target ontology entities.

Previous section outlines the fundamental elements in SBO, but such concepts clearly lack semantics and applicability. The analysis of inter-relations and constraints holding between them provides the missing features of SBO.

5.7 Semantic Bridges Inter-relations

Inter-relations between semantic bridges are defined in the context of the ontology mapping document. Ontology mapping document is formally defined as:

$$\mathcal{M} := (\mathcal{O}_s, \mathcal{O}_t, \mathcal{B}^c, \mathcal{B}^p, \mathcal{T}, \mathcal{A}^{B^c}, \mathcal{A}^{B^p}, \prec, \perp^{\mathcal{B}^c}, \perp^{\mathcal{B}^p}, \diamond)$$

where:

- \mathcal{O}_s is the source ontology;
- \mathcal{O}_t is the target ontology;
- \mathcal{B}^c is the set of concept bridges;
- \mathcal{B}^p is the set of property bridges;
- \mathcal{T} is the set of available transformation services. A transformation service T is defined as a set of arguments and their types.
- \mathcal{A}^{B^c} is the set of containers named Alternative Bridges containing concept bridges;
- \mathcal{A}^{B^p} is the set of containers, named Alternatives Bridges, containing property bridges;
- $\prec: 2^{\mathcal{B}^c \times \mathcal{B}^c}$ corresponds to a reflexive, anti-symmetric and transitive relation between concept bridges. It corresponds to a hierarchical structure of concept bridges. The following constraint holds:

$$\forall cb_1, cb_2 \in \mathcal{B}^c : source_concept(cb_1, c_1^s) \wedge target_concept(cb_1, c_1^t) \wedge \\ source_concept(cb_2, c_2^s) \wedge target_concept(cb_2, c_2^t) \wedge \prec(cb_2, cb_1) \Rightarrow is_a(c_2^s, c_1^s) \wedge is_a(c_2^t, c_1^t)$$

- $\perp^{\mathcal{B}^c} : 2^{\mathcal{B}^c \times \mathcal{A}^{\mathcal{B}^c}}$ represents the disjoint relation between concept bridges. For example, $\perp^{\mathcal{B}^c}(b_1, a_1) \wedge \perp^{\mathcal{B}^c}(b_2, a_1)$ state that concept bridges b_1 and b_2 are disjoint in alternative bridge a_1 . Further, the following constraint holds over $\perp^{\mathcal{B}^c}$:

$$\forall b \in \mathcal{B}^c \forall a_1, a_2 \in \mathcal{A}^{\mathcal{B}^c} : \perp^{\mathcal{B}^c}(b, a_1) \wedge \perp^{\mathcal{B}^c}(b, a_2) \rightarrow a_1 = a_2.$$

- $\perp^{\mathcal{B}^p} : 2^{\mathcal{B}^p \times \mathcal{A}^{\mathcal{B}^p}}$ represents the disjoint relation between property bridges. The constraint defined for concept bridges do not holds for property bridge, since property bridge can be disjoint in many context.
- $\diamond : 2^{\mathcal{B}^c \times (\mathcal{B}^p \cup \mathcal{A}^{\mathcal{B}^p})}$ represents the aggregation relation between concept bridges and property bridges. Property bridges are executed in scope of concept bridges, which means that the property bridge will be executed for all, but only for those property instances defined in the source concept instances. Therefore, this function specifies in which concepts bridges a property bridge is scoped. For example: $\diamond(b_1, b_2), b_1 \in \mathcal{B}^c, b_2 \in \mathcal{B}^p \cup \mathcal{A}^{\mathcal{B}^p}$ specifies that the b_2 bridge will be executed in scope of concept bridge b_1 . If b_2 is an alternative bridge of property bridges, the process attempts to execute property bridges until one is successful.

This aggregation relation further implies that the concept of both concept bridge and root concept (the subject concept of the first step of the path.) of each path of the property bridge must be the same. In fact, if property bridges are executed in scope of a certain instance, it is fundamental that the properties values are accessible from such instance. This constraint is formally represented as the following disjunction:

$$\forall cb \in \mathcal{B}^c, \forall pb \in \mathcal{B}^p, \forall L \in L_s^{pb} : \diamond(cb, pb) \wedge s_0 \in L \wedge source_concept(cb, c) \Rightarrow SUBJECT(s_0, c) \vee \\ \forall cb \in \mathcal{B}^c, \forall pb \in \mathcal{B}^p, \forall L \in L_t^{pb} : \diamond(cb, pb) \wedge s_0 \in L \wedge target_concept(cb, c) \Rightarrow SUBJECT(s_0, c)$$

Additionally, target paths must have exactly one element. Besides not representing a conceptual requirement but instead an implementation decision, four important advantages have been identified:

1. It allows to ignore the execution order of semantic bridges, since if a multi-step target path is specified, it would be necessary to previously execute the property bridges that create the relation between instances;
2. As consequence, it prevents recursive dependencies between semantic bridges;
3. Promotes modularization, systematization and error detection, respecting concept bridges. In fact, if a multi-step target path is absolutely necessary, it directly imply the need for a new concept bridge;
4. Evolution procedures are simplified. Since fewer dependencies exist between semantic bridges, less propagation of changes arises between semantic bridges.

SBO describes taxonomic relations and other non-hierarchical inter-relations between semantic bridges, constraining their meaning and application, improving their semantics consequently.

The paper enters now in the description of the ontology mapping system development. Concrete parts of the system are described and implementation decisions are explained.

6 Service-Oriented Approach

In this section the system architecture is described, which respect the identifications of the main components of the systems, their inter-relations and their role in the system.

The conceived system architecture came up from two simple observations:

1. It is virtually impossible to provide all possible transformation requirements using a monolithic and static ontology mapping system;
2. The transformation results of and ontology mapping execution largely depends on the transformation service available and associated with semantic bridges.

These simple observations, together with the five goals previously introduced, lead to the adoption of a modular, decentralized architecture, where independent transformation modules are attached to the system functional core modules (i.e. bridging, execution, negotiation, evolution, etc.) [5]. These independent transformation modules are called *Services* and provide their resources to the MAFRA core modules through the MAFRA Service Interface (Figure 2).

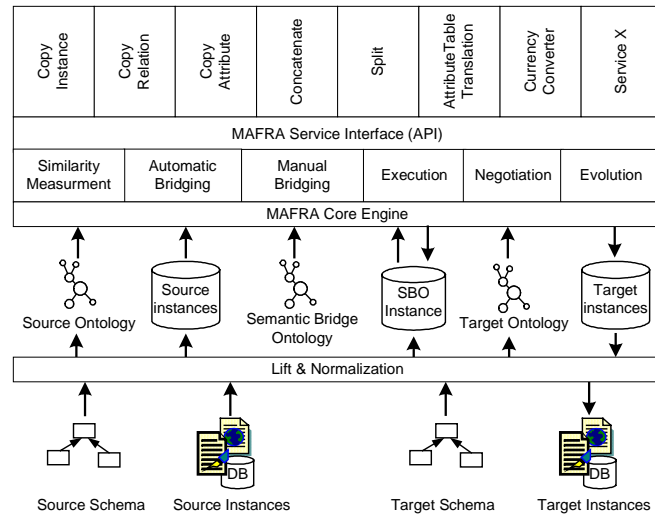


Figure 2. MAFRA System Architecture

The service-oriented approach suggested in this architecture advocates the need to exploit the knowledge and capabilities associated with each Service in order to increase automation and quality of the overall mapping process. Services represent expertise in dealing not only with the transformation process but also with other task related to the manipulation of associated bridges. Hence, services assume special relevance in automatic matching, automatic bridging, evolution of semantic bridges, validation and negotiation. Each Service interface is improved depending on the capabilities it provides to the MAFRA Core Modules.

Services are described and specified through a simple ontology, enumerating a set of arguments and their characteristics. Each service is then responsible for the validation of semantic bridge it is associated to. Other information might be stored depending on the service and their implemented features. For example, some services define similarity measures, which are typically used during the automatic similarity-measuring phase to determine matches (semantic equivalence) between source and target ontologies entities. Deeper details on service-oriented approach can be found in [14].

Table 1 presents some examples of services and their arguments, currently available in MAFRA Toolkit.

Table 1. Some MAFRA prototypal mapping relations and their interface

CopyInstance	Creates target concept instances for each source concept instance that fulfils the conditions. This is the service implicitly associated with concept bridges.	
Argument ID	Type	Comment
Source Concept	Class	Source ontology class whose instances will be transformed
Extensional Specif.	ArrayOfConditions	Extensional definition of source class instances
Target Concept	Class	Target ontology class to create
CopyRelation	Creates a relation between concepts instances based	
Source Path	Path	Source ontology path for each path the bridge will be executed
Extensional Specif.	ArrayOfConditions	Extensional definition of source class instance
Target Path	RelationPath	Target ontology path to create
CopyAttribute	Copies (no changes) the source property value (string) to the target property.	
Source Attribute	AttributePath	Source ontology attribute whose instances will be copied
Target Attribute	AttributePath	Target ontology attribute to copy to
CountRelations	Counts the number of instances of a relation.	
Source Path	Path	Source ontology path to copy
Extensional Specif.	ArrayOfConditions	Extensionally defines source class instance
Target Attribute	AttributePath	Target ontology attribute to create
Split	Splits by separators, the literal in source attribute.	
Source Attribute	AttributePath	Source ontology attribute whose instances will be splitted
Separators	ArrayOfLiterals	Literals or regular expressions to split by
Target Attributes	ArrayOfAttributePaths	List of attributes to instantiate with splitted values

Concat	Concatenate several attribute values each one separated by a literal.	
Source Attributes	ArrayOfAttributePath	List of source ontology attribute whose instances will be concatenated
Separators	ArrayOfLiterals	Literals to concatenate between attribute values
Target Attribute	AttributePaths	Target attributes to instantiate with concatenated values

7 Mapping examples

This section will raise some issues concerning the instantiation of SBO according to some simple examples. The presented examples try to use wide acceptable semantic relations, but typical semantic relations tend to be subjective and ambiguous.

To ease reading of examples, a declarative and intuitive language will be used instead of the formal language used in section 5.

7.1 Object-oriented approach

SBO model suggests a hierarchical, object-oriented approach of semantic bridges definition. Accordingly, the MAFRA methodology advises its use when possible, promoting modularity, reusability and readability of the mapping definition. Consider Figure 3 where excerpts of two ontologies are represented in UML. In the left side, playing the role of source ontology is the Gedcom ontology [15] and in the right, playing the role of target ontology is the Gentology ontology [16].

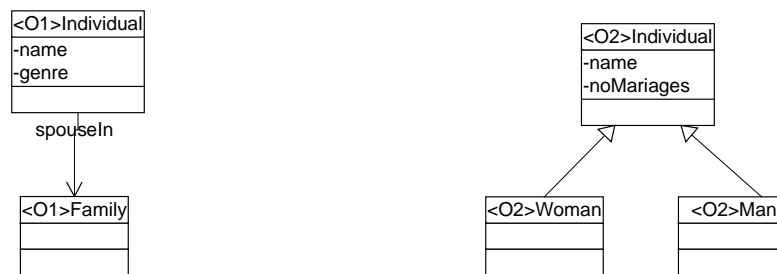


Figure 3. Excerpts of Gedcom and Gentology ontologies

Both source and target ontologies define the concept Individual, which according to domain expert should be mapped. Because the target entity is a concept, a concept bridge will be used:

```
Individual2Individual=conceptBridge(<O1>Individual,<O2>Individual,{abstract=True},{})
```

<O2>Individual represents an abstract concept since it is never instantiated. This means that there are no <O2>Individual instances, but only <O2>Woman and <O2>Man instances. Accordingly, the Individual2Individual bridge is state abstract. In fact, <O1>Individual will originate either <O2>Woman or <O2>Man, depending on the value of the <O1>Individual/genre/LITERAL attribute in each instance. Consequently, two concept bridges are to be defined, each requiring a condition expression to check the value of <O1>Individual/genre/LITERAL. If equals "M", a <O2>Man instance will be created, otherwise a <O2>Woman instance will be created.

```
Individual2Woman=conceptBridge(<O1>Individual,<O2>Woman,{},{<O1>Individual/genre/LITERAL="F"})
Individual2Man=conceptBridge(<O1>Individual,<O2>Man,{},{<O1>Individual/genre/LITERAL="M"})
subBridgeOf(Individual2Woman,Individual2Individual)
subBridgeOf(Individual2Man,Individual2Individual)
```

Several comparison operators are available in the current implementation of MAFRA, namely Equal, Different, Match (regular expression match) and Like (*a la* SQL substring match). Or, And, Xor and Not logic operators are also available, allowing the specification of complex conditional expressions. For example, one could specify a more elaborated condition for Individual2Man bridge:

```
{ Or(
    <O1>Individual/genre/LITERAL="M",
    <O1>Individual/genre.LITERAL Like "Male",
    <O1>Individual/genre/LITERAL Match "^M*" )}
```

7.2 Disjoint bridges

The specification of conditions in independent bridges might not be sufficient for controlling the execution of the mapping. In fact, definition of conditions ensures that the bridge is executed only if the condition holds, but it does not ensure that other bridges are not executed. This would not be a problem in the scenario of Figure 3 if the <O1>Individual/genre/LITERAL for each instance of <O1>Individual is either female or male. But if by some reason

both values are specified, two <O2>Individual instances will be created which is a semantic error. Similar situation can additionally occur in case no sufficient conditions are specified for one of disjoint bridges. In order to explicitly express this constraint, the disjoint constraint is applied:

```
ManOrWoman=disjointBridge(list(Individual2Woman,Individual2Man))
```

7.3 Property Bridges

Once concept bridges are defined, property bridges are defined and attached to concept bridges. In the running example, two property bridges are to be defined. Bridge name2name relates <O1>Individual.name to <O2>Individual.name, and bridge spouseIn2noMariages relates <O1>Individual.spouseIn to <O2>Individual.noMariages. Property bridge name2name will use the CopyAttribute service, since no transformation in the source values is required. On the other hand, the spouseIn2noMariages property bridge will use the CountRelation service.

```
name2name=propertyBridge(
    {SourcePath=<O1>Individual/name/LITERAL},
    {TargetPath=<O2>Individual/name/LITERAL},
    CopyAttribute,
    {}, {})
spouseIn2noMariages=propertyBridge(
    {SourcePath=<O1>Individual/spouseIn/Family},
    {TargetPath=<O2>Individual/noMariages/LITERAL},
    CountRelation,
    {}, {})
◇( Individual2Individual,name2name)
◇( Individual2Individual,spouseIn2noMariages)
```

Remark that property bridges are defined within the scope of the Individual2Individual concept bridge, but they will be executed within the scope of Individual2Man and Individual2Woman, since these are sub-bridges of Individual2Individual, and this is an abstract concept bridge. This hierarchical inheritance mechanism, similar to object-oriented modelling, profits and provides the benefices recognized to object-oriented approach, specially when applied to distributed, well-structured, inter-dependent ontologies, which are progressively adopted and supported in Semantic Web [17].

8 Execution process

The execution process corresponds to the MAFRA execution module, where source ontology instances are actually transformed/translated into target ontology instances. The execution process comprises two phases:

- In first phase all concept bridges run for each and all instances of the source concept defined in concept bridge. This phase creates the target instances and attaches them a new identity;
- In second phase property bridges are executed in scope of each newly created instance. The property bridges to execute are those defined in scope of concept bridge and those defined in scope of all super concept bridges, if some exists.

This clear separation between the execution of concept bridges and property bridges allows increased modularization and simplicity of transformation. The Copy Relation service is paradigmatic of this modularization and simplicity. Imagine linking the instance to another one that do not yet exists. In such situation, it would be necessary to run the concept bridge responsible for the creation of the target entity in the scope of the first concept bridge. Furthermore, all property bridge of second concept bridge would be executed too, which could again imply the execution of concept bridges, and so on.

Instance transformations are performed by the CopyInstance service, which is part of MAFRA core system and cannot be substituted by another externally, provided service. This constraint is necessary due to specificities of instance transformation requirements. In special, these specificities have important impact in CopyRelation service. In fact, these are the fundamental services required to manage concept instances (i) CopyInstance create the instances and (ii) CopyRelation inter-relates them.

8.1 Instance transformation

Due to the need to copy source instances relations to target instances, it is necessary to maintain information about which source instance gave rise to which target instance. The CopyInstance service is responsible for this task. The so-called Transformation Table is used to maintain this information, through a list of tuples in the form of:

```
<source_instance_id, extensional_specification, target_instance_id, concept_bridge_id >
```

A reference to the concept bridge responsible for the creation of the target instance is kept in the table in order to access its property bridges (that are defined in the scope of the concept bridge) in second phase of the execution process.

Extensional specification is responsible for the univocal specification of source instance, necessary when many target instances are created from the same source instance. It will be described below.

Consider the mapping scenario of Figure 4 respecting excerpts of TourinFrance [18] and SIGRT [19] ontologies. TourinFrance (namespace TIF) is the source ontology and SIGRT (namespace SIGRT) is the target ontology. SIGRT ontology is composed of three interrelated concepts, while TourinFrance ontology is composed of two inter-related concepts.

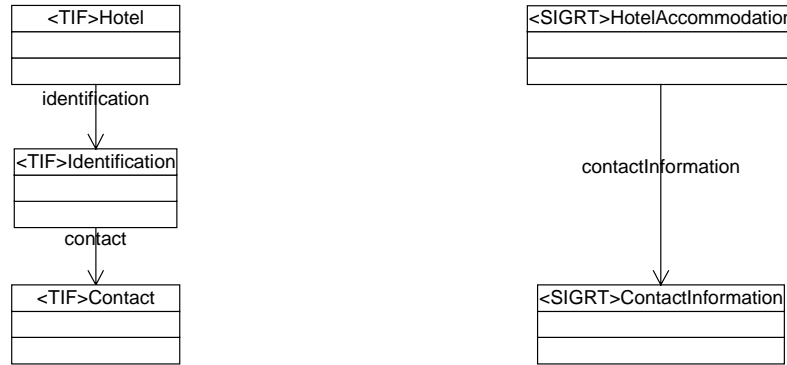


Figure 4. Excerpts of TourinFrance and SIGRT ontologies.

Two concept bridges are easily perceived (<TIF>Identification has no counterpart in SIGRT ontology and therefore is not mapped):

```
Hotel2HotelAccommodation=conceptBridge(<TIF>Hotel,<SIGRT>HotelAccommodation,{},{})
Contact2ContactInformation=conceptBridge(<TIF>Contact,<SIGRT>ContactInformation,{},{})
```

Figure 5 represents the target instances obtained from source ontology instance when executing previous concept bridges.

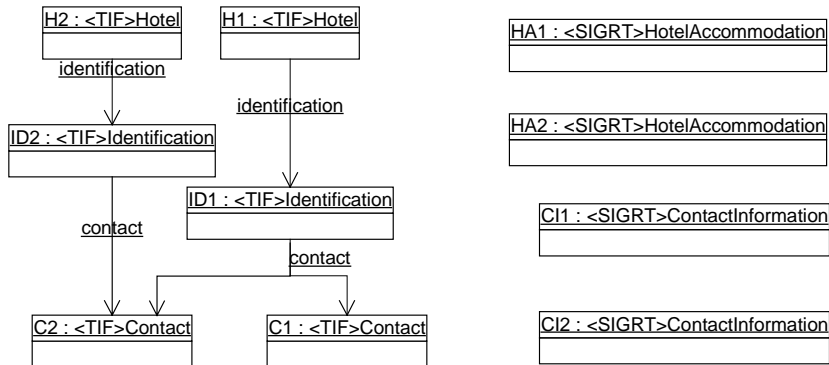


Figure 5. Source and target instances transformation

The Transformation Table resulting from these semantic bridges execution can be found in Table 2.

Table 2 – Transformation Table respecting scenario of Figure 5

Source Instance ID	Extensional Specification	Target Instance ID	Concept Bridge
H1		HA1	H2HA
H2		HA2	H2HA
C1		CI1	C2CI
C2		CI2	C2CI

8.2 Extensional Specification

Extensional Specification is required when many target instances are created from one single source instance and those target instances need to be related.

When applied in concept instance transformation, it allows creating one target instance from each source instance perspective. When applied in creating relations between concept instances it allows determining the correct pair of target instances from the source instance perspective.

Extensional specification complements the instance identity with values of instance properties to create a virtual instance in the Transformation Table. Hence each pair of source identity and extensional specification is unique in the

Transformation Table. Each of these pairs relate to only one target entity, providing the means to univocally choose between many target instances derived from the same source instance.

In order to illustrate this feature, consider the inverse scenario of Figure 4, where the SIGRT ontology is to be mapped to TourinFrance. Two concept bridges are easily perceived:

```
HotelAccommodation2Hotel=conceptBridge(<SIGRT>HotelAccommodation,<TIF>Hotel,{},{})
ContactInformation2Contact=conceptBridge(<SIGRT>ContactInformation,<TIF>Contact,{},{})
```

The need to apply extensional specification arises because no source concept is directly related to the <TIF>Identification, and because it is fundamental to create instances of <TIF>Identification in order to recreate the relation between <SIGRT>HotelAccommodation and <SIGRT>ContactInformation. The general solution advises to semantically map the target concept with one or more property values of a concept. This implies the use of the extensional specification method.

In this particular example, the domain expert decides that one <TIF>Identification instance should be created for each <SIGRT>HotelAccommodation.contactInformation.ContactInformation property value. This means that for each <SIGRT>ContactInformation instance related to <SIGRT>HotelAccommodation, an instance of <TIF> Identification will be created. According to the CopyInstance service interface, the extensional specification is a source argument. Here is the concept bridge specification:

```
HotelAccommodation2Identification=conceptBridge(
    <SIGRT>HotelAccommodation,
    <TIF>Hotel,
    {<SIGRT>HotelAccommodation/contactInformation/ContactInformation=
    <SIGRT>HotelAccommodation/contactInformation/ContactInformation},
    {},{})
```

Figure 6 represents the set of source instances and the obtained set of target instances after the execution of the three previous concept bridges.

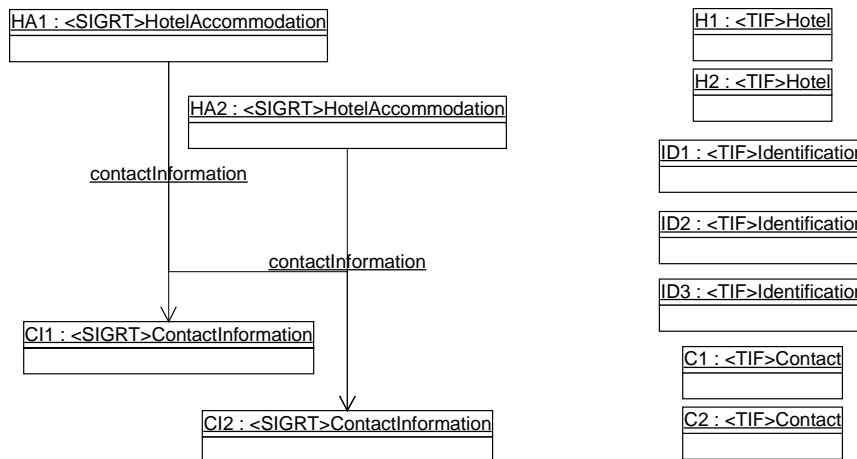


Figure 6. Source and target instance transformation representation

Table 3 represents the resulting Transformation Table.

Table 3. Transformation Table declaring extensional specification of source instances

Source Instance ID	Extensional Specification	Target Instance ID	Concept Bridge
HA1		H1	HA2H
HA2		H2	HA2H
HA1	contactInformation=CI1	ID1	HA2I
HA1	contactInformation=CI2	ID2	HA2I
HA2	contactInformation=CI2	ID3	HA2I
CI1		C1	CI2C
CI2		C2	CI2C

8.3 Inter-relation of instances

The creation of relations between instances is achieved through the CopyRelation service. Besides it is substitutable, it is expected that the provided implementation would fulfil all requirements associated with the copy relation process. In its simplest instantiation, CopyRelation service takes two arguments:

- The source path that will provide the value to iterate through in creating the target relation. The source path can be either an attribute or a relation path;
- The target relation path to instantiate.

This simple form is enough to complete the mapping scenario of Figure 6, relation HotelAccommodation to ContactInformation:

```
contact2contactInformation=propertyBridge(
    {SourcePath=<TIF>Hotel/identification/Identification},
    {TargetPath=<SIGRT>HotelAccommodation/contactInformation/ContactInformation},
    CopyRelation,
    {},{ })
◇(Hotel2HotelAccommodation,contact2contactInformation)
```

The first step in the execution of CopyRelation service is to project all paths values of the source and condition arguments. This step enumerates all possible combinations of values. Including the condition paths in the projection simplifies the verification of conditions and includes it in the same process.

CopyRelation tries to instantiate the target relation <SIGRT>HotelAccommodation/contactInformation/ContactInformation for each instance created for <SIGRT>HotelAccommodation.

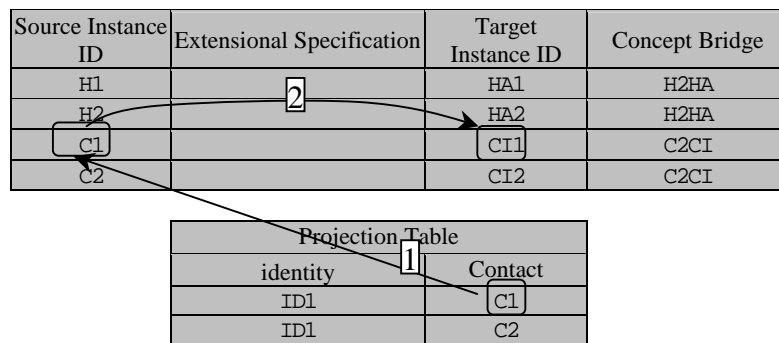


Figure 7 – Transformation and projection tables' example.

Figure 7 depicts the Transformation Table and Projection Table created during the execution of the contact2contactInformation semantic bridge in scope of instance HA1. In this case, two combinations are derived because source instance H1 is related to ID1, which is in turn related to two <TIF>Contact instances (C1 and C2). The source instance identity for each row of the projection table is then searched in the Transformation Table (step 1). The respective target instance (step 2) corresponds to the value to instantiate in the target relation. This process is executed for all lines of the projection table that fulfil the conditions associated, including cardinality of the created relation.

The CopyRelation service requires extra information if the target concept is created through extensional specification. There is the need to extensionally specify the source instance that gave rise to the target instance. Consider the inverse scenario of Figure 4 (instantiated in Figure 6). The following property bridges are additionally needed to concept bridges specified in section 8.2:

```
contactInformation2identification=propertyBridge(
    {SourcePath=<SIGRT>HotelAccommodation/contactInformation/ContactInformation},
    ExtensionalSpecification=
    {<SIGRT>HotelAccommodation/contactInformation/ContactInformation=
    <SIGRT>HotelAccommodation/contactInformation/ContactInformation}},
    {TargetPath=<TIF>Hotel/identification/Identification},
    {},{ })
◇(HotelAccommodation2Hotel,contactInformation2identification)
contactInformation2contact=propertyBridge(
    {SourcePath=<SIGRT>HotelAccommodation/contactInformation/ContactInformation},
    {Target Path=<TIF>Hotel/identification/Identification},
    {},{ })
◇(HotelAccommodation2Identification,contactInformation2contact)
```

Figure 8 illustrates the execution process for the creation of relation between H1 and ID1. The projection table includes all source properties specified as arguments: <SIGRT>HotelAccommodation as Source Path argument and

<SIGRT>HotelAccommodation/contactInformation/ContactInformation as extensional specification argument. HA1 source ontology instance is the source instance for both H1 and ID1. The extensional specification in the CopyRelation bridge, complements the identification of the target instance (step 1), allowing the correct identification of the target instance (step 2).

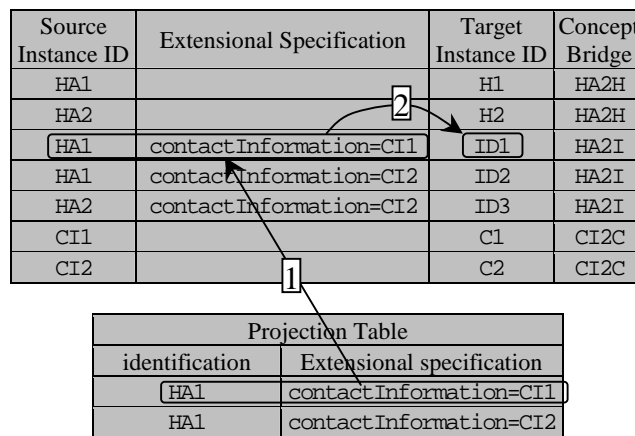


Figure 8. Transformation and projection tables with extensional specification constraints

Several other arguments are available to control the execution of bridges, namely the cardinality and control of conditions, but their functionality is service dependent.

9 Related work

Much valuable work has been done in the area of distributed databases in integrating and exchanging data between entities, but the work described here focus on the Semantic Web environment, its characteristics, constraints and technologies.

Both the centralized mediated data schema and ontology merging adopt a centralized approach in which data source schemas/ontologies are unified into a single schema/ontology. The “centralized” approach of the mediation is probably not flexible enough to be scaled up to the Web. Consequently, ontology mapping is empirically perceived as a more dynamic knowledge sharing process than centralized schema mediation or ontology merging, and arises naturally as a potential solution for ontology-based interoperability problems. Three distinct ontology mapping projects are considered paradigmatic approaches. In [20] authors describe an extension to Protégé to map between domain ontologies and problem solving methods. This approach defines a valuable set of desiderata and mapping dimensions, but its known implementation lacks some important features especially in allowing mapping between multiple concepts. Yet, there is no record of experiments that apply it to the Semantic Web environment. The second approach is RDFT [21], a meta-ontology that describes Equivalence and Versioning relations between either an XML DTD or RDFS document and another XML DTD or RDFS document. An RDFT instantiation describes the semantic relations between source and target documents, which will be further applied in the transformation of documents. Thirdly, the Buster project [22] applies information integration to the GIS domain. Two distinct approaches are proposed: rule-based transformation and re-classification. The rule-based approach applies a procedural transformation to instance properties, while classification applies class membership conditions to infer target classification through description-logic tools. However, these two approaches are not integrated, which limits mapping capabilities.

10 Experiences

MAFRA Toolkit has been adopted as the development, representation and transformation engine in the Harmonise project [23]. This project intends to overcome the interoperability problems occurring between major tourism operators in Europe. Problems arise due to the use of distinct information representation languages like XML and RDF, and different business and information specifications, like those provided by SIGRT [19], by TourinFrance [18] and by FTB [24]. Harmonise uses an “Interoperability Minimum Harmonisation Ontology” as *lingua franca* between agents. MAFRA is responsible for the acquisition, representation and execution of the ontology mapping between each agent specific ontology and IMHO. IMHO describes the tourism domain in about 64 concepts, 120 attributes and 213 inter-relations between concepts. IMHO and partner ontologies are very different. For example, the FTB information schema specifies 1 concept with 48 attributes and SIGRT defines about 50 concepts with typically less than ten properties. Even if many and different semantic and syntactic mismatches occur in mapping these ontologies, no conceptual limitations

have been detected in MAFRA Toolkit, and only a few refinements of the prototypal mapping services have been required.

Concerning performance issues, some simple experience has been made. Considering the lack of experience reports with ontology mapping tools, the report contained in [25] constitute a simple but valuable reference. They report the experience in transforming a dataset of 21164 instances respecting the Gedcom ontology [15], into instances respecting the Gentology ontology [16]. These are two very similar ontologies, whose mapping requires only simple semantic relations. Figure 9 presents a screen-shot during the ontology mapping specification of this example using MAFRA Toolkit graphical user interface.

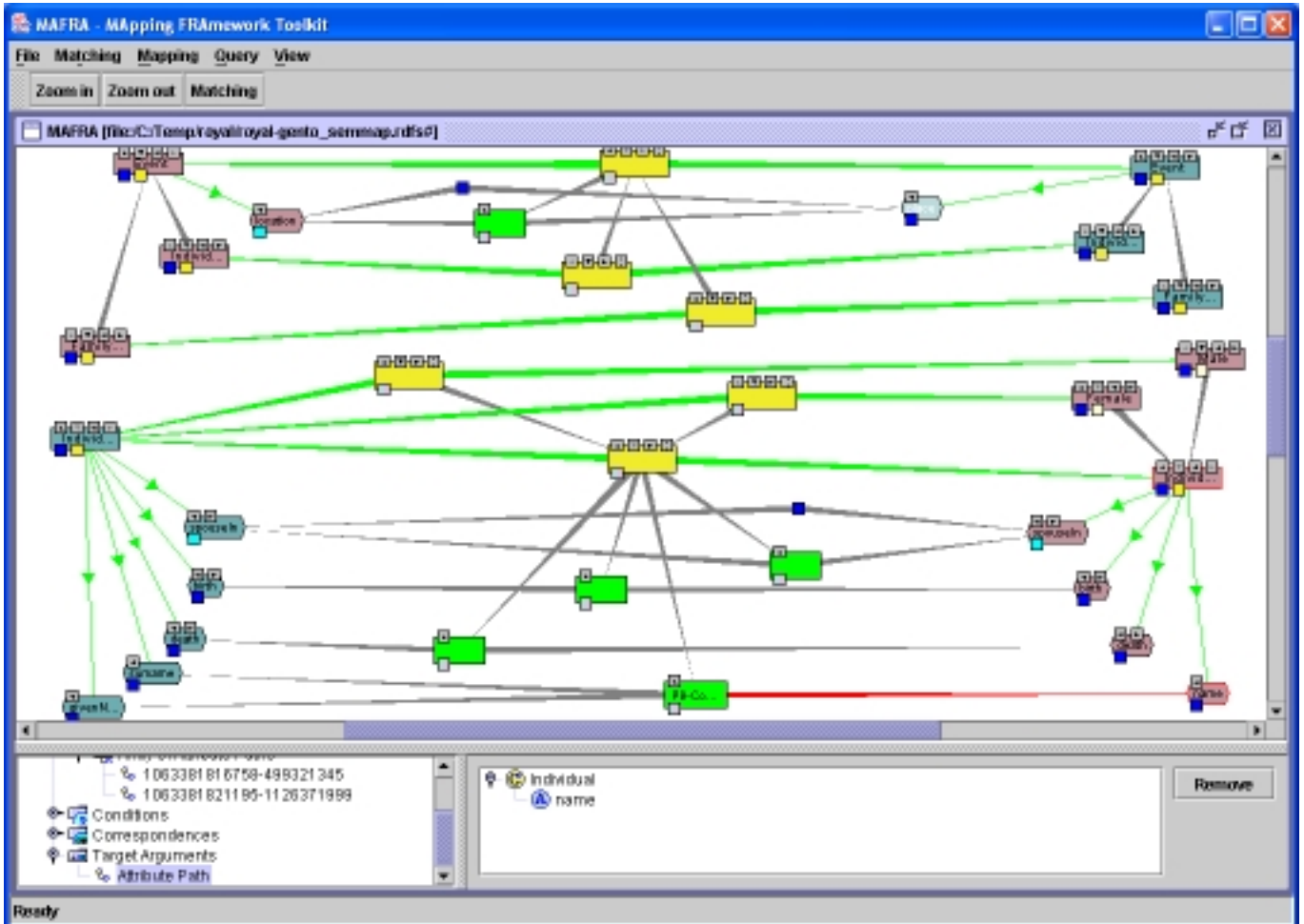


Figure 9 - MAFRA Toolkit screen-shot: specification or/and customization of semantic bridges

The ontology mapping developed using MAFRA Toolkit specifies the semantic relations presented in the previously mentioned report and others harvest from the examples found in the OntoMerge web pages. No distinctions have been detectable from both transformations. Ontologies are represented in DAML, which is not directly supported by MAFRA Toolkit. However, a representation translator from DAML to RDFS is available, which transform ontologies in a few seconds. Dataset is represented in RDF, thus excusing any transformation in MAFRA Toolkit execution. On the contrary, OntoMerge requires transformations if both ontologies and dataset. OntoMerge reports a 22 minutes execution time in a Pentium III at 800MHz, while MAFRA Toolkit achieved the same results in less then 2 minutes in a Pentium II at 350 MHz. If a Pentium 4M 2.0Mhz is used, MAFRA requires less then 1 minute. Recently, the OntoMerge authors reported the same test execution in similar time as MAFRA Toolkit, which seems a more reasonable scenario.

11 Conclusion

The work described in this paper has been developed in scope of the SANSKI project, in which semi-automatic ontology mapping technology is developed to increase interoperability between artificial agents operating in Semantic Web. Several domain of research are addressed in this project but in this particular paper the focus is the specification, representation and execution of ontology mappings.

Ontology mapping and Semantic Bridging Ontology (SBO) have been formally defined, promoting a univocally understanding of the process. SBO represent our conceptualization of the ontology mapping problem, describing the inputs, outputs, components, their dependencies and constraints. Due to its formal description, multiple notations and syntaxes can be used to represent ontology mapping documents.

The Semantic Bridging Ontology and consequently the specification and execution phases puts forward a new strategy, combining rule-based transformation with Description Logic-like approaches. The rule-base approach allows the declarative specification of semantic relations providing an intuitive and immediate mapping specification. The DL approach serves to overcome conceptual mismatches between ontologies. Its modular structure, in which each concept is semantically bridged independently of others concepts provides the features to easily evolve the mapping according to ontologies changes.

The system architecture in turn complements this feature allowing the easy integration of additional transformation services. The architecture of system based on the notion of multi-dimensional service. Such services are responsible not only for the traditional instance transformation but also for other services dependent tasks. In fact, the service-oriented paradigm applied to the specification and execution phases is generalized to other phases namely evolution, validation and automatic bridging. The concept of transformation service is therefore considerably extended to include other capabilities then those respecting transformation. The domain expert know-how is acquired and integrated in the system not as one monolithic structure but multiple modules evolving and adapting separately.

Though not all tasks are addressed in this paper, in its current status MAFRA Toolkit provides support in four modules of the MAFRA framework: lift and normalization of source ontologies and datasets [26], support for the specification of matches between ontology entities, specification of semantic bridges, automatic proposal of semantic bridges according to matches, instance transformation, and an easy and intuitive graphical user interface.

Currently, our efforts are focused in the evolution the ontology mapping process according to the ontologies changes. It is not difficult for ontology mapping to become incoherent when a number of changes occur in mapped ontologies. The adopted service-oriented architecture provides a good starting point, in the sense that once again the competence and know-how to deal with specific changes are forwarded to transformation services.

A middle-term project should facilitate the mapping acquisition between different agents using meaning negotiation. This phase will also potentially benefice from the service-oriented architecture, relying on services the argumentation upon proposed semantic relations. While experiences and comparisons with other ontology mapping tools are insufficient, they showed that MAFRA Toolkit fulfils real-world requirements with a good performance.

12 Acknowledgements

This work is partially supported by the Portuguese MCT-FCT project POCTI/2001/GES/41830. Many thanks to Alexander Maedche for his ideas and support during my stay at FZI. Thanks to Oliver Fodor for his feedback on the application of MAFRA Toolkit in the Harmonise project. Thanks to Jorge Santos for his revisions and many valuable discussions.

13 References

- [1] Gruber, T. R.; "Towards Principles for the Design of Ontologies Used for Knowledge Sharing"; Formal Ontology in Conceptual Analysis and Knowledge Representation; Ed. Guarino, N. and Poli, R.; Deventer, The Netherlands; 1993.
- [2] Critchlow, T., Ganesh Madhavan and Musick Ron; "Automatic Generation of Warehouse Mediators Using an Ontology Engine"; 5th Workshop KRDB-98; Ed. Borgida, A., Chaudhri, V., and Staudt, M.; pp. 8.1-8.8; 1998.
- [3] Mitra, P. and Wiederhold, G.; "An Algebra for Semantic Interoperability of Information Sources"; 2nd. IEEE Symp. on BioInformatics and Bioengineering (BIBE'2001); Ed. Bourbakis, N.; pp. 174-182; Bethesda, USA; 2001.
- [4] Maedche, A., Staab, S., Studer, R., Sure, Y. and Volz, R.; "SEAL - Tying Up Information Integration and Web Site Management by Ontologies"; IEEE Data Engineering Bulletin; 25(1), pp. 10-17; 2002.
- [5] Maedche, A., Motik, B., Silva, N. and Volz, R.; "MAFRA - A MAPPING FRAMework for Distributed Ontologies in the Semantic Web"; Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002; pp. 60-68; Lyon, France; 2002.
- [6] Visser, P. R. S., Jones, D. M., Bench-Capon, T. J. M. and Shave, M. J. R.; "An Analysis of Ontological Mismatches: Heterogeneity versus Interoperability"; 1997.

- [7] Bernstein, P. A. and Rahm, E.; "On Matching Schemas Automatically"; Microsoft Research; MSR-TR 2001-17; Redmond, WA, USA; 2001.
- [8] Doan, A., Madhavan, J., Domingos, P. and Halevy, A.; "Learning to map ontologies on the Semantic Web"; World-Wide Web Conference (WWW-02); Honolulu, Hawaii, USA; 2002.
- [9] Klein, M., Kiryakov, A., Ognyanov, D. and Fensel, D.; "Ontology Versioning and Change Detection on the Web"; 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02); pp. 197-212; Heidelberg; 2002.
- [10] Stojanovic Ljiljana, Maedche, A., Motik, B. and Stojanovic Nenad; "User-driven Ontology Evolution Management"; 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02); pp. 197-212; Heidelberg; 2002.
- [11] Bailin, S. C. and Truszkowski, W.; "Ontology Negotiation between Agents Supporting Intelligent Information Management"; Workshop on Ontologies in Agent Systems (OAS'2001) at the 5th International Conference on Autonomous Agents; 2001.
- [12] van Elst, L. and Abecker, A.; "Negotiating Domain Ontologies in Distributed Organizational Memories"; AAAI-02 Workshop on Meaning Negotiation (MeaN-02) held in conjunction with Eighteenth National Conference on Artificial Intelligence; pp. 32-35; 2002.
- [13] Hsieh, D.; "A Logic to Unify Semantic Network Knowledge Systems with Object-Oriented Database Models"; SRI International; SRI-CSL-90-15; 1990.
- [14] Silva, N., Rocha, J. and Cardoso, J.; "E-Business Interoperability through Ontology Semantic Mapping"; IFIP Working Conference on Virtual Enterprises; Ed. Camarinha-Matos, L. and Afsarmanesh, H.; Lugano, Switzerland; 2003.
- [15] Gedcom; <http://www.daml.org/2001/01/gedcom/gedcom.daml>; accessed in 2003.
- [16] Gentology; <http://orlando.drc.com/daml/Ontology/Genealogy/3.1/Gentology-ont.daml>; accessed in 2003.
- [17] Maedche, A., Motik, B., Stojanovic Ljiljana, Studer, R. and Volz, R.; "An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies"; Proceedings of the WWW 2003; pp. 439-448; Budapest, Hungary; 2003.
- [18] TourinFrance; <http://www.tourisme.gouv.fr>; accessed in 2003.
- [19] SIGRT; <http://www.dgturismo.pt/irt/>; accessed in 2003.
- [20] Park, J. Y., Gennari, J. H. and Musen, M. A.; "Mappings for Reuse in Knowledge-based Systems"; 11th Workshop on Knowledge Acquisition, Modelling and Management (KAW 98); Banff, Canada; 1998.
- [21] Omelayenko, B.; "RDF: A Mapping Meta-Ontology for Business Integration"; Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002; pp. 76-83; Lyon, France; 2002.
- [22] Stuckenschmidt, H. and Wache, H.; "Context Modeling and Transformation for Semantic Interoperability"; Workshop "Knowledge Representation meets Databases" (KRDB 2000) at ECAI'2000; pp. 115-126; Berlin, Germany; 2000.
- [23] Harmonise; www.harmonise.org; accessed in 2003.
- [24] Finnish Tourist Board; <http://www.mek.fi>; accessed in 2003.
- [25] Dou, D., McDermott, D. and Qi, P.; "Ontology translation on the semantic web"; International Conference on Ontologies, Databases and Applications of Semantics; Catania (Sicily), Italy; 2003.
- [26] Fodor, O., Dell'Erba, M., Ricci, F., Spada, A. and Werthner, H.; "Conceptual Normalisation of XML Data for Interoperability in Tourism"; Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002; pp. 69-76; 2002.