Instance Migration in Heterogeneous Ontology Environments

Luciano Serafini and Andrei Tamilin

Foundation B. Kessler - IRST (formerly ITC-IRST)
Via Sommarive 18, 38050 Trento, Italy
{serafini,tamilin}@itc.it

Abstract. In this paper we address the problem of migrating instances between heterogeneous overlapping ontologies. The instance migration problem arises when one wants to reclassify a set of instances of a source ontology into a semantically related target ontology. Our approach exploits mappings between ontologies, which are used to reconcile both conceptual and individual level heterogeneity, and further used to draw the migration process. We ground the approach on a distributed description logic (DDL), in which ontologies are formally encoded as DL knowledge bases and mappings as bridge rules and individual correspondences. From the theoretical side, we study the task of reasoning with instance data in DDL composed of $\mathcal{SHIQ}$ ontologies and define a correct and complete distributed tableaux inference procedure. From the practical side, we upgrade the DRAGO DDL reasoner for dealing with instances and further show how it can be used to drive the migration of instances between heterogeneous ontologies.

1 Introduction

The semantic web steadily evolves growing into a container of multiple distributed ontologies. Although ontologies are supposed to provide a consensual model of the world, those found in practice are far from being so. They rather formalize a subjective view in accordance with diverse assumptions, such as target goals, available background knowledge, biases, etc. This fact inevitably leads to a situation in which the same domain is represented by more than one ontology in *heterogeneous* ways.

The classical approach to the problem of *reconciling heterogeneity* between ontologies consists of two essential steps: matching and reasoning. By matching one creates a set of *mappings*—comprising semantic correspondences—between elements of different ontologies. For example, a mapping can express the fact that the concept Scholar in one ontology is equivalent to the concept Student in another ontology, or that the individual Samuel Clemens in one ontology corresponds to the individual Mark Twain in the other ontology. Mappings can be created manually, by domain experts, or discovered (semi-)automatically by existing matching techniques [8]. Once mappings are stated, it is necessary to provide a method for *reasoning* with them. Formally, this amounts to evaluating logical consequences of mappings on the mapped ontologies.

Mappings form a source to a target ontology can be used to transfer knowledge between the two ontologies. When ontologies are represented in OWL (formally corresponding to DL knowledge bases) there are two types of knowledge that can be transferred: terminological knowledge (i.e., mappings can force new concept subsumptions

in the target ontology) and assertional knowledge (i.e., mappings can assert new instances of a certain concept). To motivate and explain the problem, let us consider a scenario in which ontology mappings are used for migrating instances.

Example 1 (Motivating scenario). A computer science department of some university employed an ontology $O^{\text{CS Dept}}$ for accessing data related to organization of the department. Successively, a human resource office of the university decides to make use of ontologies to represent data about the employees of the whole university. For this purpose they create an ontology $O^{\text{HRO Uni}}$ and further populate it with instance data. To save time, they decide to reuse the work done by the computer science department. However, to do that they need to solve the following two problems. First, the conceptualization in $O^{\text{HRO Uni}}$ is different from the one in $O^{\text{CS Dept}}$, so the classification of instances done in $O^{\text{CS Dept}}$ must be adapted to the classification schema implemented in $O^{\text{HRO Uni}}$. Second, the identifiers used in $O^{\text{CS Dept}}$ for people do not coincide with the identifiers used in $O^{\text{HRO Uni}}$, thus they have to provide a way to transform (regenerate) identifiers of the people they want to insert in their ontology.

The main objective of the paper is to provide a logical characterization of the assertional information enforced by a set of mappings. Based on this characterization, we propose a sound and complete distributed tableaux algorithm that reclassifies instances of a source ontology into a target ontology in accordance to the mappings. The feasibility of this approach is shown by describing the implementation of such an algorithm in the DRAGO DDL Reasoner[1]. The approach described in the paper, relies on the logical framework of distributed description logics (DDL) [4]. In such a framework a *distributed knowledge base* consists of a family of standard DL knowledge bases, corresponding to each given ontology, a set of *bridge rules*, corresponding to mapping between pairs of terminologies (T-boxes), and *individual correspondences*, corresponding to rules for transforming individuals across instance storages (A-boxes). The concrete contribution of the present work includes:

- characterization of the role that bridge rules and individual correspondences play when reasoning with instances in DDL; in particular, we show that they are capable of propagating concept membership assertions across ontologies
- definition of a sound and complete distributed extension to $\mathcal{SHIQ}$-A-box tableau for reasoning with instances in DDL; the algorithm implements the backward chaining strategy for computation of propagated assertions by distributed communication with DL tableaux reasoners attached to other ontologies
- extending the DRAGO DDL Reasoner, currently limited to reasoning without instances, with the algorithm proposed in this paper and further showing its application to drive instance migrations.

The paper is organized as follows. In Section 2 we introduce the definition of a DDL distributed knowledge base with bridge rules and individual mappings. In Section 3 we study how bridge rules and individual mappings propagate information across knowledge bases. The implementation in DRAGO and the application to the instance migration scenario is further outlined in Section 4. We end up with an overview of related work and concluding remarks.

[1] `http://sra.itc.it/projects/drago`

2 Distributed Knowledge Bases

As introduced by Borgida and Serafini in [4], DDL is a formalism for representing multiple ontologies *pairwise* interconnected by directional semantic mappings. In this section we recall and extend the basic definitions of DDL.

2.1 Syntax and Semantics

A distributed knowledge base formalizes a set of ontologies interconnected by semantic mappings. The first component of a distributed knowledge base is a family of knowledge bases $\mathcal{K} = \{\mathcal{K}_i\}_{i \in I}$. According to a standard DL definitions, each $\mathcal{K}_i$ consists of a terminological component $\mathcal{T}_i$ (T-box) and an assertional component $\mathcal{A}_i$ (A-box). Since the very same symbol can be used in two knowledge bases with different meaning, to unambiguously refer to elements of $\mathcal{K}_i$, they are prefixed with the index i of the knowledge base. The notations $i : a$ $i : C$, $i : C \sqsubseteq D$, $i : C(a)$ and $i : R(a, b)$, stand for an individual a, concept C, subsumption $C \sqsubseteq D$, assertions $C(a)$ and $R(a, b)$, respectively in the knowledge base $\mathcal{K}_i$.

Mappings from $\mathcal{K}_i$ to $\mathcal{K}_j$ knowledge bases with ($i \neq j$) are encoded as bridge rules and individual correspondences, which are expressions of the following forms:

- $i : C \xrightarrow{\sqsubseteq} j : D$ (into-bridge rule)
- $i : C \xrightarrow{\sqsupseteq} j : D$ (onto-bridge rule)
- $i : a \longmapsto j : b$ (individual correspondence)

where C and D are concept names of $\mathcal{T}_i$ and $\mathcal{T}_j$, and a and b are individuals of $\mathcal{A}_i$ and $\mathcal{A}_j$ respectively[2].

Both bridge rules and individual correspondences from $\mathcal{K}_i$ to $\mathcal{K}_j$ express relations between $\mathcal{K}_i$ and $\mathcal{K}_j$ viewed from the j-th *subjective* point of view. Intuitively, the into-bridge rule i : PhDThesis $\xrightarrow{\sqsubseteq}$ j : Thesis states that, from the j-th point of view the concept PhDThesis in i is less general than its local concept Thesis. Similarly, the onto-bridge rule i : InProceedings $\xrightarrow{\sqsupseteq}$ j : ConferencePaper expresses the more generality relation. The individual correspondence i : mario_phd_thesis $\longmapsto$ j : mario_thesis expresses the fact the $\mathcal{A}_j$'s individual mario_thesis is one of the possible translations in the language of $\mathcal{K}_j$ of the $\mathcal{A}_i$'s individual mario_phd_thesis. In the general case we admit that an individual can have more than one translation.

A *distributed T-box* $\mathfrak{T}$ consists of T-boxes $\mathcal{T}_i$ and a collection $\mathfrak{B}$ of bridge rules between them. A *distributed A-box* $\mathfrak{A}$ consists of A-boxes $\mathcal{A}_i$ and a collection of individual correspondences $\mathfrak{C}$. A *distributed knowledge base* $\mathfrak{K}$ is then a tuple $\langle \mathfrak{T}, \mathfrak{A} \rangle$.

The semantics of DDL is defined with a fundamental assumption that each knowledge base $\mathcal{K}_i$ in the family is *locally interpreted* on its *local interpretation domain*. To support directionality, (i.e., mappings from i to j only propagate in the i-to-j-direction),

[2] In this work we concentrate only on individual correspondences, and don't consider complete correspondences which does not have any additional effect in data migration

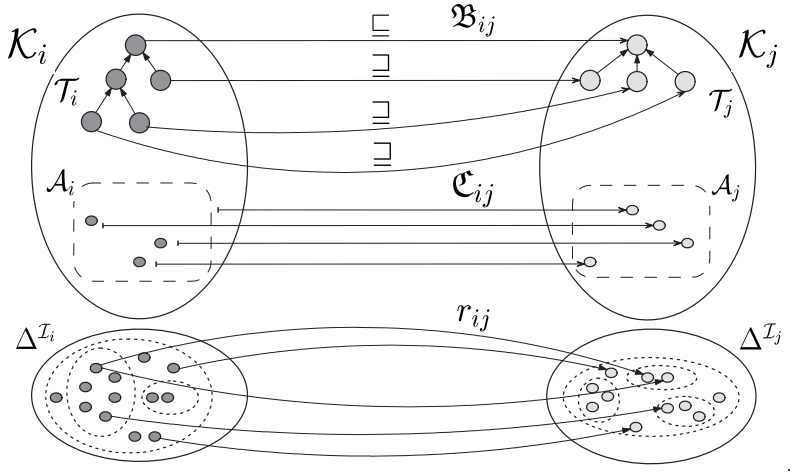


Fig. 1. Visualized semantics of DDL framework

we admit the hole interpretation $\mathcal{I}_\epsilon$ with empty domain (see more details in [15])[3]. By definition, we impose that $\mathcal{I}_\epsilon$ satisfies any knowledge base. To resolve heterogeneity between different domains the DDL defines a binary *domain relation* r between pairs of these domains. Figure 1 intuitively depicts component elements of DDL semantics.

A *distributed interpretation* $\mathfrak{I}$ of a distributed knowledge base $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{A} \rangle$ consists of a family of local interpretations $\mathcal{I}_i$ on local interpretation domains $\Delta^{\mathcal{I}_i}$ and a family of domain relations $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ between pairs of local domains.

A distributed interpretation $\mathfrak{I}$ *satisfies* a distributed knowledge base $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{A} \rangle$, is called a model of $\mathfrak{K}$, if all its' components are satisfied according to the following rules:

- $\mathcal{I}_i$ satisfies $\mathcal{K}_i$
- $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$ for all $i : C \xrightarrow{\sqsupseteq} j : D$
- $r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$ for all $i : C \xrightarrow{\sqsubseteq} j : D$
- $b^{\mathcal{I}_j} \subseteq r_{ij}(a^{\mathcal{I}_i})$ for all $i : a \longmapsto j : b$

2.2 Inference services

Although both in DL and Distributed DL the fundamental reasoning services lay in verification of concept satisfiability/subsumption and instance checking/retrieval within a certain ontology, in DDL, besides the ontology itself, the other ontologies and mappings between them should be taken into account. Given a distributed knowledge base $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{A} \rangle$, the *distributed inference services* can be defined as follows:

[3] Classically, DL interpretation maps every individual into an *element* of the domain, while the hole maps everything into the empty *set*. To allow homogeneous treatment of standard DL interpretations and holes, we require that any individual x is standardly interpreted into a singleton set, rather than into an element of the domain. Hence, $\mathcal{I}_i \models C(a) \Longleftrightarrow a^{\mathcal{I}_i} \subseteq C^{\mathcal{I}_i}$, rather than $a^{\mathcal{I}_i} \in C^{\mathcal{I}_i}$. We thank the anonymous reviewer for pointing at this mismatch

Satisfiability: A concept C is *satisfiable* in i with respect to $\mathfrak{K}$ if there exists a distributed interpretation $\mathfrak{I}$ of $\mathfrak{K}$ such that $C^{\mathcal{I}_i} \neq \emptyset$.

Subsumption: A concept C is *subsumed* by a concept D in i with respect to $\mathfrak{K}$ if for every distributed interpretation $\mathfrak{I}$ of $\mathfrak{K}$ we have that $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. In this case we will write $\mathfrak{K} \models i : C \sqsubseteq D$.

Instantiation: An individual a is an *instance of* a concept C in i with respect to $\mathfrak{K}$ if for every distributed interpretation $\mathfrak{I}$ of $\mathfrak{K}$ we have that $a^{\mathcal{I}_i} \subseteq C^{\mathcal{I}_i}$. In this case we will write $\mathfrak{K} \models i : C(a)$.

Retrieval: Computing the individuals in $\mathcal{K}_i$ that instantiate a given concept C in i with respect to $\mathfrak{K}$.

The group of concept satisfiability/subsumption services is typically referred to as terminological reasoning services, while the remaining instantiation/retrieval services are grouped into assertional reasoning services.

The question of providing terminological services for DDL has been already studied in [15]. It has been shown that certain combinations of into- and onto-bridge rules can lead to the propagation of knowledge in form of subsumption axioms across ontologies participating in DDL. Moreover, in case of DDL with $\mathcal{SHIQ}$ components without instances adding these additional propagation rules to existing DL tableaux algorithms leads to a correct and complete reasoning in DDL. The presented method has been also implemented in the DRAGO DDL reasoner.

In the consequent sections, we investigate the assertional reasoning services.

3 Reasoning with Instances in Distributed Knowledge Bases

For the sake of clarity, we start considering the case of DDL with two component knowledge bases and unidirectional sets of bridge rules and individual correspondences. For the general results and proofs we refer the interested reader to the technical report [16].

3.1 Inference patterns

In the following we characterize the knowledge propagated from a knowledge base i (the source) to j (the target) by a set of *propagation rules* of the form:

$$\frac{(1)\text{ facts in } i, \quad (2)\text{ bridge rules from } i \text{ to } j, \quad (3)\text{ individual mappings from } i \text{ to } j}{(4)\text{ fact in } j}$$

which must be read as: if the facts in (1) are true in $\mathcal{K}_i$, the bridge rules in (2) are contained in $\mathfrak{B}_{ij}$, the individual correspondences in (3) are contained in $\mathfrak{C}_{ij}$, then the fact in (4) must be true in $\mathcal{K}_j$.

Following the semantics of mappings in DDL outlined in the previous section, it can be observed that the individual correspondences can interact with into-bridge rules with the effect of propagating concept membership assertions:

$$\frac{i : C(a), \quad i : C \xrightarrow{\sqsubseteq} j : D, \quad i : a \longmapsto j : b}{j : D(b)} \tag{1}$$

Indeed, $b^{\mathcal{I}_j} \subseteq r_{ij}(a^{\mathcal{I}_i}) \subseteq r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$.

In practice, this means that if an ontology O_1 defines an instance mario of a concept PhDStudent, and an ontology O_2 has some individual name person_123, then a bridge rule $1 : \mathsf{PhDStudent} \xrightarrow{\sqsubseteq} 2 : \mathsf{Student}$ and an individual correspondence $1 : \mathsf{mario} \longmapsto 2 : \mathsf{person\_123}$ entail the O_2's assertion that person_123 is an instance of Student.

In languages that support disjunction, the above propagation can be generalized to the propagation of instance membership over a disjunction of $n \geqslant 0$ concepts:

$$\frac{i : (C_1 \sqcup \ldots \sqcup C_n)(a), \quad i : C_k \xrightarrow{\sqsubseteq} j : D_k \ (1 \leqslant k \leqslant n), \quad i : a \longmapsto j : b}{j : (D_1 \sqcup \ldots \sqcup D_n)(b)} \tag{2}$$

Several observations on the stated propagation pattern require a specific attention.

Generality Rule (2) appears to be the *most general* form of assertion propagation in DDL when individual correspondences are restricted to be *functional*. A set of individual correspondences $\mathfrak{C}_{ij}$ is functional if for every individual a of $\mathcal{A}_i$ the set $\mathfrak{C}_{ij}$ contains at most one individual correspondence $i : a \longmapsto j : b$. For the sake of presentation, in this paper we restrict ourself to functional individual correspondences, leaving the most general case to the technical report [16])[4].

Inconsistency propagation When $n = 0$, the inference pattern in (2) becomes the following inference rule:

$$\frac{i : \bot(a), \quad i : a \longmapsto j : b}{j : \bot(b)} \tag{3}$$

which states that to propagate the inconsistency of $\mathcal{K}_i$ to $\mathcal{K}_j$ it's enough to have one single individual correspondence. From the representational point of view this inference rule is very fragile. We currently do not see an easy solution to fix this sensitivity to inconsistency propagation. This topic will be subject for further studies.

Instance migration Up to now, we have supposed that individual correspondences are explicitly enumerated in $\mathfrak{C}_{ij}$. However in real situation, with thousands of individuals, one cannot expect to pre-compile all the individual mappings. However, the formalism support a more compact approach of declaring individual correspondences via a translation function f_{ij}, defined on the domain of the source ontology i and producing individuals in the domain of target ontology j. For example, f_{ij} can be the identity function, hence its application yields all instances from $\mathcal{K}_i$ to be

[4] To give an intuition of the effect of non functional individual mappings, consider the case in which there are two into-bridge rules $i : C_1 \xrightarrow{\sqsubseteq} j : D_1$ and $i : C_2 \xrightarrow{\sqsubseteq} j : D_2$ and, the non functional set of individual mappings $\{i : a \longmapsto j : b, \ \ i : a \longmapsto j : c\}$. Then the fact that $\mathcal{K}_i \models C_1 \sqcup C_2(a)$ entails the disjunctive assertion $(D_1(b) \wedge D_1(c)) \vee (D_2(b) \wedge D_2(c))$. This implies that, for the general case we have to introduce the technicalities for disjunctive A-boxes

copied to $\mathcal{K}_j$. Such an approach is practically applicable to the instance migration scenario described in the introduction.

Once a translation function f_{ij} is defined, we can revisit propagation pattern (2):

$$\frac{i : (C_1 \sqcup \ldots \sqcup C_n)(a), \quad i : C_k \xrightarrow{\sqsubseteq} j : D_k \ (1 \leqslant k \leqslant n)}{j : (D_1 \sqcup \ldots \sqcup D_n)(f_{ij}(a))} \tag{4}$$

This later means that a fresh individual $f_{ij}(a)$ is injected into $\mathcal{K}_j$ and asserted as an instance of the disjunction of the D_k's.

3.2 Soundness and completeness

To demonstrate the correctness and completeness of the inference pattern presented in Section 3.1, we follow the approach similar to the one taken in [15]. The main idea consists in construction of an operator which essentially applies the generalized inference pattern (2) to extend knowledge bases with new assertions induced by mappings.

Given a set of bridge rules $\mathfrak{B}_{12}$ and set of individual correspondences $\mathfrak{C}_{12}$ from $\mathcal{K}_1$ to $\mathcal{K}_2$, the *individual correspondence operator* $\mathfrak{C}_{12}(\cdot)$, taking as input a knowledge base $\mathcal{K}_1$ and producing an A-box of $\mathcal{K}_2$, is defined as follows:

$$\mathfrak{C}_{12}(\mathcal{K}_1) = \left\{ (D_1 \sqcup \ldots \sqcup D_n)(b) \;\middle|\; \begin{array}{l} \mathcal{K}_1 \models (C_1 \sqcup \ldots \sqcup C_n)(a) \\ 1 : C_k \xrightarrow{\sqsubseteq} 2 : D_k \in \mathfrak{B}_{12} \ (1 \leqslant k \leqslant n) \\ 1 : a \quad \longmapsto 2 : b \quad \in \mathfrak{C}_{12} \end{array} \right\}$$

It is remarkable that *onto*-bridge rules do not affect instance propagation. The reason is that onto-bridge rules impose only existence of preimages of objects that already exists in the target ontology. Into-bridge rules, instead, constraint the individual mappings to be defined whithin a certain range. The individual correspondence operator formalizes the assertional knowledge that is propagated across ontologies.

The characterization of the propagation of the terminological knowledge is characterized by an analogous operator, called *bridge operator*, introduced in [15] and defined as follows: $\mathfrak{B}_{12}(\cdot)$, taking as input a knowledge base $\mathcal{K}_1$ and producing a T-box of $\mathcal{K}_2$:

$$\mathfrak{B}_{12}(\mathcal{K}_1) = \left\{ B \sqsubseteq D_1 \sqcup \ldots \sqcup D_n \;\middle|\; \begin{array}{l} \mathcal{T}_1 \models A \sqsubseteq C_1 \sqcup \ldots \sqcup C_n \\ 1 : C_k \xrightarrow{\sqsubseteq} 2 : D_k \in \mathfrak{B}_{12} \ (1 \leqslant k \leqslant n) \\ 1 : A \quad \xrightarrow{\sqsupseteq} 2 : B \quad \in \mathfrak{B}_{12} \end{array} \right\}$$

With the remarkable exception of inconsistency propagation—by rule (3)—the individual correspondences do not affect the propagation of terminological knowledge. The inferences formalized by the two operators described above *completely* describe the possible propagations that are forced by a set of bridge rules and individual correspondences. This is formally stated in the following theorem.

Theorem 1 (Soundness and completeness). *Let $\mathfrak{K}_{12}$ be a distributed knowledge base consisting of $\mathcal{K}_1$, $\mathcal{K}_2$ $\mathcal{SHIQ}$ knowledge bases, and $\mathfrak{B}_{12}$, $\mathfrak{C}_{12}$ mappings between them. For any statement ϕ (of the form $C \sqsubseteq D$ or $C(a)$) in the language of $\mathcal{K}_2$*

$$\mathfrak{K}_{12} \models 2 : \phi \quad \Longleftrightarrow \quad \langle \mathcal{T}_2 \cup \mathfrak{B}_{12}(\mathcal{K}_1), \ \mathcal{A}_2 \cup \mathfrak{C}_{12}(\mathcal{K}_1) \rangle \models \phi$$

The proof of the generalization of the Theorem 1 is fully described in the technical report. Some remarks are necessary.

Independence between terminological and assertional propagation From the characterization above, one can see that propagation of terminological and assertional knowledge are orthogonal. The two effects can be computed in parallel and independently. What is more important, however, is that the change of the A-box does not affect the propagation of the terminological knowledge. This means that if the source T-box does not change the terminological propagation is computed once for all.

Local propagation of assertional knowledge Assertional propagation operator ensures, if a change of the source A-box involves only the set of individuals $\{a_1, \ldots, a_n\}$, then assertional propagation must be computed only for the portion of the target A-box $\mathcal{A}_2$ concerning the set of individuals $\{b \mid 1 : a_i \longmapsto 2 : b \in \mathfrak{C}_{12}\}$.

Upper bound and complexity If the mapping from 1 to 2 is finite and contains m into-bridge rules, n onto-bridge rules, and o individual correspondences, then the bridge operator $\mathfrak{B}_{12}$ applied to any knowledge base generates at most $n * 2^m$ subsumption statements, and the individual correspondence operator $\mathfrak{C}_{12}$ generates at most $o * 2^m$ instantiation statements. In total, the maximal number of statements that can propagate from $\mathcal{K}_1$ to $\mathcal{K}_2$ via mappings is $(n + o) * 2^m$. Since the propagation of statements needs checking subsumption and instantiation in the source knowledge base, which is EXPTIME complete, we have that computing subsumption and instantiation in a distributed setting is EXPTIME complete in the dimension of the source knowledge base plus mappings.

Vanilla implementation The above theorem supports a vanilla implementation of *forward chaining* inference engine for DDL. The implementation consists of three steps: computation of propagation operators $\mathfrak{B}_{12}(\mathcal{K}_1)$ and $\mathfrak{C}_{12}(\mathcal{K}_1)$, construction of extended version of knowledge base $\mathcal{K}_2$ as $\langle \mathcal{T}_2 \cup \mathfrak{B}_{12}(\mathcal{K}_1),\ \mathcal{A}_2 \cup \mathfrak{C}_{12}(\mathcal{K}_1)\rangle$, and finally applying to this knowledge base one of existing DL reasoners, such as FaCT++ [18], Racer [10], or Pellet [17].

This approach to reasoning has a strong advantage of reuse of existing highly optimized DL reasoners, however it can be very costly for situations when semantic mappings are changed dynamically or when the required number of reasoning questions to be verified is relatively small. In the next section, we propose an alternative, *backward chaining* reasoning approach, which does "lazy", or on demand, computation of propagated axioms and hence better fits to instable and short-living distributed environments.

3.3 Distributed tableaux algorithm

In this section we present a distributed tableaux algorithm for reasoning with instances in DDL. The main design idea consists in constructing a network of standard DL tableaux, one for each ontology in DDL, which communicate via mappings in a backward fashion.

Since we restricted the expressivity of ontologies participating in DDL to $\mathcal{SHIQ}$ DL, we will consider in the following that ontologies $\mathcal{K}_1$ and $\mathcal{K}_2$ from a distributed

knowledge base $\mathfrak{K}_{12} = \langle \mathfrak{T}_{12}, \mathfrak{A}_{12} \rangle$ are attached with $\mathcal{SHIQ}$-tableau reasoning procedures $\mathbf{Tab}_1$ and $\mathbf{Tab}_2$ [13]. Due to the reduction of reasoning with concepts to reasoning with instances [2], we suppose that each procedure $\mathbf{Tab}_i(\alpha)$ can check the satisfiability of any statement α of form $i : C \sqsubseteq D$, $i : C(a)$.

As described in [13], the $\mathcal{SHIQ}$-tableau works on a so called "completion forest", a collection of trees whose root nodes correspond to instances in A-box. Given a knowledge base, the algorithm initializes a completion forest $\mathcal{F}$ with a set of root nodes $\mathbf{x}_0 = \{x_0^k\}$ corresponding to a set of instances b_k in A-box, labels each x_0^k with a set $\mathcal{L}(x_0^k)$ of concepts C for each concept assertion $C(b_k)$ in A-box, and finally draws an edge between x_0^k and x_0^m for each role assertion $R(h_k, h_m)$ in A-box. After that, the set of $\mathcal{SHIQ}$ completion rules expanding the forest $\mathcal{F}$ is applied. The fully expanded forest then represents a model of the knowledge base. To test entailment of arbitrary assertion $X(a)$, $\neg X(a)$ is added to A-box and further the tableau is expanded to see whether a model of such knowledge base can be constructed or not.

To accommodate the knowledge propagation from $\mathcal{K}_1$ to $\mathcal{K}_2$ in $\mathfrak{K}_{12}$, we intervene in the completion process of $\mathbf{Tab}_2$ in order to capture new facts induced by bridge rules and individual correspondences. Hence, we get a *distributed tableaux procedure* $\mathbf{DTab}_2$ which extends $\mathbf{Tab}_2$ with two additional expansion rules:

$\mathfrak{C}_{12}$-rule:

if 1. $x \in \mathbf{x}_0$, such that x is a node corresponding to individual b and $1 : a \longmapsto 2 : b$,
$\mathbf{H} \subseteq \{H_k \mid 1 : B_k \xrightarrow{\sqsubseteq} 2 : H_k \in \mathfrak{B}_{12}\}$,
$\mathbf{B} = \{B_k \mid H_k \in \mathbf{H}, 1 : B_k \xrightarrow{\sqsubseteq} 2 : H_k \in \mathfrak{B}_{12}\}$,
2. $\mathbf{Tab}_1((\bigsqcup \mathbf{B})(a)) =$ *true* for $\bigsqcup \mathbf{H} \notin \mathcal{L}(x)$,
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\bigsqcup \mathbf{H}\}$

$\mathfrak{B}_{12}$-rule:

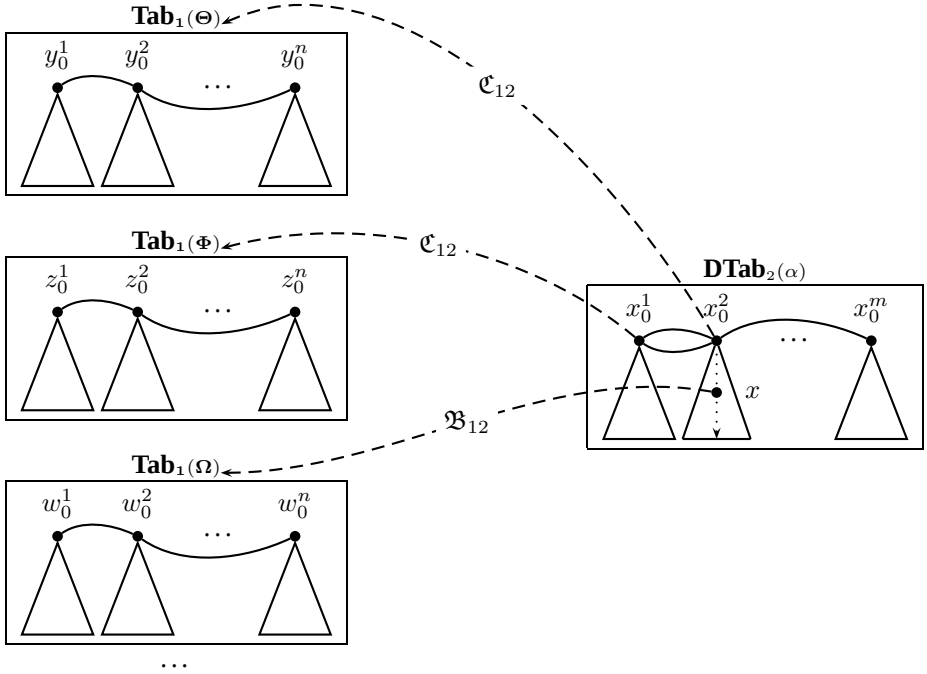
if 1. $G \in \mathcal{L}(x)$, such that $1 : A \xrightarrow{\sqsupseteq} 2 : G \in \mathfrak{B}_{12}$,
$\mathbf{H} \subseteq \{H_k \mid 1 : B_k \xrightarrow{\sqsubseteq} 2 : H_k \in \mathfrak{B}_{12}\}$,
$\mathbf{B} = \{B_k \mid H_k \in \mathbf{H}, 1 : B_k \xrightarrow{\sqsubseteq} 2 : H_k \in \mathfrak{B}_{12}\}$,
2. $\mathbf{Tab}_1(A \sqsubseteq \bigsqcup \mathbf{B}) =$ *true* for $\bigsqcup \mathbf{H} \notin \mathcal{L}(x)$,
then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\bigsqcup \mathbf{H}\}$

The principle idea of these additional expansion rules consists in implementing backward versions of bridge and individual correspondences operators introduced in Section 3.2. According to rule $\mathfrak{C}_{12}$, if $\mathbf{DTab}_2$ encounters a root node x connected by an individual correspondence, then a disjunction of concepts $\bigsqcup \mathbf{H}$ should be added to the label $\mathcal{L}(x)$ if $\bigsqcup \mathbf{H}(x)$ is entailed by interaction of individual correspondence with into-rules. To determine this entailment, $\mathbf{DTab}_2$ remotely requests foreign $\mathbf{Tab}_1$ to check if it is the case that $\bigsqcup \mathbf{B}(b)$ in $\mathcal{K}_1$.

The role of $\mathfrak{B}_{12}$-rule is to analyse the nodes of completion forest and import consequences of subsumption propagations. If $\mathbf{DTab}_2$ encounters a node x which contains

a label G connected by an onto-bridge rule, then if $G \sqsubseteq \bigsqcup \mathbf{H}$ is entailed by the bridge rules, the label $\bigsqcup \mathbf{H}$ is added to x. While in order to determine the entailment, **DTab**$_2$ invokes the procedure **Tab**$_1$ with a question whether a subsumption $A \sqsubseteq \bigsqcup \mathbf{B}$ holds in $\mathcal{K}_1$.

Graphically, the distributed execution of **DTab**$_2$ can be depicted as follows:



Theorem 2 (Termination, Soundness, Completeness). *Given $\mathcal{SHIQ}$ DL knowledge bases $\mathcal{K}_1$ and $\mathcal{K}_2$, let $\mathfrak{K}_{12} = \langle \langle \{\mathcal{T}_1, \mathcal{T}_2\}, \mathfrak{B}_{12} \rangle, \langle \{\mathcal{A}_1, \mathcal{A}_2\}, \mathfrak{C}_{12} \rangle \rangle$ be a distributed knowledge base. Then, given a $\mathcal{SHIQ}$ statement α*

1. *a distributed procedure* **DTab**$_2(\alpha)$ *terminates, and*
2. *α is satisfiable in $\mathcal{K}_2$ with respect to $\mathfrak{K}_{12}$ if and only if* **DTab**$_2(\alpha)$ *yields a complete and clash-free completion forest.*

It can be shown that the proposed algorithm enjoys generalization to arbitrary number of $\mathcal{SHIQ}$ knowledge bases participating in DDL, and moreover can be extended to distributed knowledge bases containing cyclical pathes of bridge rules and individual correspondences. For the sake of clarity, we omit the discussion of these generalizations and refer the reader to the technical report [16] for details.

4 Implementation and Application

In this section we first outline the implementation of the distributed tableaux procedure on top of the DRAGO DDL Reasoner; second we describe its application to a problem of instance migration between heterogeneous ontologies.

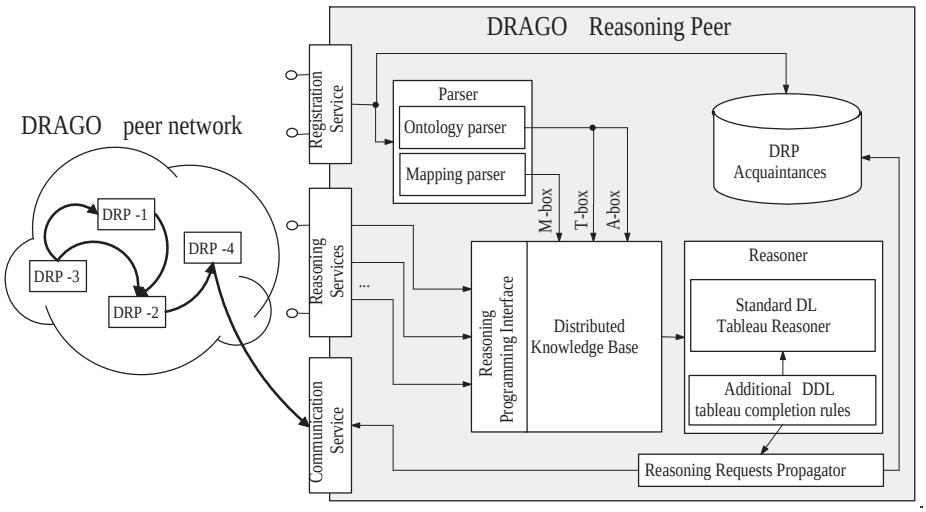


Fig. 2. DRAGO Architecture

4.1 DRAGO DDL Reasoner

DRAGO DDL Reasoner implements a peer-to-peer architecture for reasoning with a set of ontologies $\{O_i\}_{i\in I}$ interconnected by semantic mappings $\{M_{ij}\}_{i\neq j\in I}$. The principal component of DRAGO is the *DRAGO Reasoning Peer*, or shortly DRP. Each DRP P can host a set $\mathbf{O}_P = \{O_i\}_{i\in I_P}$ (where $I_P \subseteq I$) of ontologies, and a set of mappings $\mathbf{M}_P = \{M_{ij}\}_{i\in I, j\in I_P}$ incoming into $\mathbf{O}_P$. Mappings can come both from ontologies of the peer P, or from ontologies of other peers. Each DRP P supports both *local* and *distributed* reasoning tasks on each ontology in $\mathbf{O}_P$. Local reasoning tasks are standard DL reasoning tasks defined on one single ontology. Distributed reasoning tasks are those defined on the distributed knowledge base induced by the ontologies and mappings managed by all the DRP's. When the DRP P executes a distributed reasoning task on an ontology with mappings coming from another peer Q, P submits the reasoning sub-tasks to Q. Figure 4.1 sketches the overall architecture of DRAGO and displays principle components forming the Reasoning Peer.

The *Reasoner* is the central component of DRP. It is implemented by extending a standard tableaux based DL reasoner with additional completion rules that implements the distributed reasoning. DRAGO is based on the Pellet reasoner, an open-source Java implementation of tableau for reasoning with OWL ontologies [17]. In accordance with algorithm presented in Section 3.3, these new completion rules comprise the bridge rule and individual correspondence. Due to the remark to Theorem 1 on independence of terminological and assertional propagation, we reused the bridge completion rule from available purely terminological version of DRAGO, and implemented an individual completion rule, new due to the present paper.

Practically, DRAGO works with ontologies represented in OWL [3] and semantic mappings encoded in C-OWL [5]. Syntactically, C-OWL extends OWL with constructs for specification of semantic mappings, while semantically it is founded on the presented DDL framework.

4.2 Execution of migrations with DRAGO

In the following we describe how to use DRAGO to execute the task of instance migration between heterogeneous ontologies. Given a source ontology O_s and a target ontology O_t, the task of instance migration from O_s to O_t can be encoded into the following steps:

1. Match concepts of O_t with concepts of O_s and then encode discovered semantic relations into a set $\mathfrak{B}_{st}$ of bridge rules between O_t and O_s. This task can be done manually or with the help with some (semi-)automatic ontology matcher.
2. Choose a translation f_{st} from individuals of O_s to individuals of O_t, and generate the set of individual correspondences $\mathfrak{C}_{st} = \{s : x \longmapsto t : f_{st}(x) | x \text{ individual of } O_s\}$. The simplest case of translation function is the identity function, when exactly the same set of individuals of O_s are included into the target ontology O_t. However, this is not always the case. E.g., if O_s and O_t follow different naming conventions, then individuals of O_s are needed to be renamed in accordance with rules O_t.
3. Instantiate DRAGO Reasoning Peer DRP_s for O_s and DRP_t for O_t with semantic mapping $\mathfrak{B}_{st} \cup \mathfrak{C}_{st}$.
4. Ask DRP_t to classify translated individuals in accordance with O_t.

4.3 Experimental run

To see the instance migration in work and get the practical impression from the implemented distributed tableaux, we emulated and executed the motivating scenario described in the introduction. As a source of instances, we used a publicly available ontology populated with data on publications at the Semantic Web Conference[5]. To evidence the correctness of migration, as a target ontology we used a source ontology with all instance data removed from it. As required, the establishment of into-bridge rules between the same concepts of source and target and the application of identity function to translation of source ontology instances yields the exactly the same classification of migrated instances in target ontology.

Besides the correctness, the performed practical run demonstrated the necessity of developing optimization strategies for completion of distributed tableaux for instance migration. This is due to the relatively slow speed of migration (e.g., the reclassification of 50 instances of the selected source ontology through 30 established into-bridge rules takes around 5 minutes). The slow speed is a consequence of the necessity to consider all possible disjunctions of concepts connected by into-bridge rules when completing distributed tableaux. In the next study, we investigate possible optimization strategies for reducing the amount of disjunctions to be considered to still guarantee the complete reasoning result.

[5] `http://annotation.semanticweb.org/iswc/iswc.owl`

5 Related Work

The problem of heterogeneity is one of the crucial issues to be resolved on the semantic web. This explains the big research interest to devising frameworks capable of representing and reasoning with multiple ontologies interrelated by semantic mappings. While DL is already a standard for working with web ontologies, the question of formal representations and reasoning with mappings is still a subject to the standardization. Hence, multiple frameworks co-exist.

In *OntoEngine* [7], the authors address the problem of translating instances from a source ontology into a heterogeneous target. In their approach mappings between ontologies are represented using the same primitives as for encoding knowledge within ontologies themselves, i.e., using "subClassOf","subPropertyOf", etc. axioms. The reasoning with mapping is based on idea of merging ontologies together with the mappings into a single ontology, in which further the standard reasoner can execute instantiation queries over vocabulary of target ontology. The main drawback of this approach is its strong centralization, which is not typically affordable on the web.

In contrast, the *SomeWhere* [9] targets a question of decentralized approach to querying heterogeneous ontologies. Similarly to *OntoEngine*, mappings in *SomeWhere* has a form of a subsumption statements, but the reasoning is based on rewriting techniques for combining reasoning over heterogeneous ontologies. The big advantage of the presented approach is its scalability, while the disadvantage is its limitation to a "propositional" ontologies, containing only disjunction, conjunction and negation.

Another recent example of decentralized infrastructure for querying distributed ontologies is *KAONp2p* [11, 12]. The authors adopt the approach of [6] to express mappings as correspondences between conjunctive queries over ontologies. The querying further requires the terminologies and mapping to be merged into a single global ontology, while instance data is then retrieved from distributed instance storages.

The recent study of query answering in *distributed description logics* has been proposed in [1]. The main idea consist in constructing a closure ontology by forward propagating, via DDL mappings, relevant axioms contained in other mapped ontologies (in a vein of vanilla implementation of DDL reasoner discussed in the current study). Doing so, further enables reformulation of distributed query answering problem into local query answering. Although the approach of [1] is sound, the authors point out the incompleteness of their study.

Another important framework is $\mathcal{E}$*-connections* [14]. Original purpose of $\mathcal{E}$-connections is to aggregate ontologies that model different (non-overlapping) aspects of the world, rather then integrate those overlapping as in DDL. Nonetheless, it has been shown in [14] that mathematically DDL constructs can be simulated in $\mathcal{E}$-connections, however sacrificing the directionality of knowledge propagation. Another difference concerns with reasoning approach. In contrast to distributed coordinating tableaux in DDL, in $\mathcal{E}$-connections a global tableau, both theoretically and practically, needs to be constructed.

6 Conclusion

In the present study, we investigated a task of correct and complete migration of instances of one ontology into another heterogeneous ontology. We formally grounded

our approach on DDL framework, which allowed us to instantiate the problem of migration into the problem of reasoning with instances in DDL distributed knowledge base. We theoretically formalized this inference and defined the distributed tableau algorithm for reasoning with multiple $\mathcal{SHIQ}$ DL onotlogies. Do demonstrate the feasibility, we implemented the preliminary version of the algorithm in DRAGO Reasoner and applied it to a simple migration task.

References

1. F. Alkhateeb and A. Zimmermann. Query Answering in Distributed Description Logics. In *Proc. of the 1st Conference on New Technologies, Mobility and Security (NTMS)*, 2007.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. 2003.
3. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L. Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, February 2004. `http://www.w3.org/TR/owl-ref`.
4. A. Borgida and L. Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *Journal of Data Semantics*, 1:153–184, 2003.
5. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing Ontologies. *Journal on Web Semantics*, 1(4):325–343, 2004.
6. D. Calvanese, G. De Giacomo, and M. Lenzerini. A Framework for Ontology Integration. In *Proc. of the Semantic Web Working Symposium (SWWS-2001)*, pages 303–316, 2001.
7. D. Dou, D. McDermott, and P. Qi. Ontology Translation on the Semantic Web. *The Journal on Data Semantics*, 3360:35–57, 2005.
8. J. Euzenat and P. Shvaiko, editors. *Ontology Matching*. Springer Verlag, 2007.
9. F. Goasdoué and M-C. Rousset. Querying Distributed Data through Distributed Ontologies: a Simple but Scalable Approach. *IEEE Intelligent Systems*, 18(5):60–65, 2003.
10. V. Haarslev and R. Moller. RACER System Description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR-2001)*, pages 701–706, 2001.
11. P. Haase and B. Motik. A Mapping System for the Integration of OWL-DL Ontologies. In *Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems (IHIS 05)*, pages 9–16. ACM Press, 2005.
12. P. Haase and Y. Wang. A Decentralized Infrastructure for Query Answering over Distributed Ontologies. In *Proceedings of the 22nd Annual ACM Symposium on Applied Computing (SAC-2007)*, 2007.
13. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic SHIQ. In *Proceedings of the 17th International Conference on Automated Deduction (CADE-2000)*, pages 482–496, 2000.
14. O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-Connections of Abstract Description Systems. *Artificial Intelligence*, 156(1):1–73, 2004.
15. L. Serafini, A. Borgida, and A. Tamilin. Aspects of Distributed and Modular Ontology Reasoning. In *Proc. of the 19th Joint Conference on Artificial Intelligence (IJCAI)*, 2005.
16. L. Serafini and A. Tamilin. Reasoning with Instances in Distributed Description Logics. Technical report, Fondazione Bruno Kessler - IRST, 2007. `http://sra.itc.it/people/tamilin/publications/2007/iswc/tr.pdf`.
17. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A Practical OWL-DL Reasoner. *Journal of Web Semantics*, 2006.
18. D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130, pages 292–297, 2006.