DRAGO: Distributed Reasoning Architecture for the Semantic Web*

Luciano Serafini[1] and Andrei Tamilin[2]

[1] ITC-IRST, Trento 38050, Italy,
luciano.serafini@itc.it,
[2] DIT, University of Trento, Trento 38050, Italy,
andrei.tamilin@dit.unitn.it

Abstract. The paper addresses the problem of reasoning with multiple ontologies interrelated with semantic mappings. This problem is becoming more and more relevant due to the necessity of building a scalable ontological reasoning tools for the Semantic Web. In contrast to the so called global approach, in which reasoning with multiple semantically related ontologies is performed in a global knowledge base that encodes both ontologies and semantic mappings, we propose a *distributed reasoning approach* in which reasoning is the result of combination via semantic mappings of local reasonings chunks performed in single ontologies. The paper presents a tableau-based distributed reasoning procedure which is sound and complete w.r.t. Distributed Description Logics, the formal framework used to represent multiple semantically connected ontologies. The paper also describes the design and implementation principles of a distributed reasoning system, called DRAGO (Distributed Reasoning Architecture for a Galaxy of Ontology), that implements such distributed decision procedure.

1 Introduction

The number of ontologies appearing on the web is growing steadily. Each ontology describes a domain of interest from a subjective perspective and level of granularity. This inevitably leads to a *heterogeneity* between ontologies describing even the very same domain. As a consequence, making multiple heterogeneous ontologies *interoperate*, is becoming a significant problem on the Semantic Web.

The common approach for supporting ontology interoperability is based on the definition of semantic relations between entities belonging to different ontologies, called a *semantic mapping*. A simple example of semantic mapping is the one stating that the concept Student in one ontology is more specific than the concept Person of another ontology.

Several proposals of languages for expressing semantic mappings have been done so far. Some of them have a well-designed semantics, for example C-OWL

* We thank Alexander Borgida for very inspiring discussions on the DDL framework. We also grateful to Fausto Giunchiglia, Maxym Mykhalchuk and Yuting Zhao for discussions about C-OWL.

[3], $\mathcal{E}$-connected OWL [9]. Examples of less formally grounded proposals are RDF Transformation [18] and MAFRA Semantic Bridge Ontology [16].

However, semantic mappings are not enough to guarantee ontology interoperability. One has also to provide the capability of *reasoning* within a system comprised of multiple ontologies interrelated by semantic mappings. So far, the reasoning approach dominating on the current Semantic Web rephrases the problem of reasoning with multiple interconnected ontologies into a problem of reasoning in a *global* ontology that encodes both ontologies and mappings in a unique blob. This approach, however, brings a number of drawbacks, such as (i) non-scalability, (ii) loosing language and reasoning specificity, (iii) loosing privacy and autonomy of ontological knowledge.

In this paper, we suggest an alternative approach based on the contextual reasoning paradigm. Namely, the reasoning with multiple ontologies is performed by a suitable combination, via semantic mappings, of local reasoning chunks, internally executed in each distinct ontology. In a nutshell, we propose a distributed tableau algorithm, which is capable of checking concept satisfiability in a set of interconnected ontologies by combining local (standard) tableaux procedures that check satisfiability inside the single ontology. This first proposal focuses on ontologies which can be expressed in the $\mathcal{SHIQ}$ fragment of Description Logic[14]. Suggested decision procedure is sound and complete w.r.t. Distributed Description Logics framework[2], used to represent multiple semantically connected ontologies.

Comparing to the global approach, proposed distributed reasoning is more scalable, since the reasoning process is performed in a partitioned search space and propagates through bridge rules, which are used to guide the search. It respects privacy and supports information hiding by requiring only *local reasoning services* rather than the direct access to ontologies. Finally, it supports languages specificity, since it combines different local reasoning procedures, each of which can be tailored on the local ontology language.

The distributed tableaux proposed in this paper has been implemented in a system called DRAGO (Distributed Reasoning Architecture for a Galaxy of Ontologies). DRAGO represents a peer-to-peer like architecture in which every peer registers a set of ontologies and mappings, and the reasoning is implemented using local reasoning in the registered ontologies and by coordinating with other peers when local ontologies are semantically connected with the ontologies registered in other peers.

The paper is structured as follows. In the first part we recall the Distributed Description Logics framework and enunciate the main properties. A wider description of DDL is provided by the technical report [20]. In Section 3 we describe the abstract distributed tableau algorithm that computes subsumption in DDL. In Section 4 we describe the ongoing work in DRAGO system and then in Section 5 compare DRAGO with other approaches and systems relevant for reasoning with distributed ontologies.

2 Distributed Description Logics

Description Logic (DL) has been advocated as the suitable formal tool to represent and reason about ontologies. Distributed Description Logics (DDL) [2]

is a *natural* generalization of the DL framework designed to formalize multiple ontologies interconnected by semantic mappings. In this section we briefly recall the definition of DDL.

As defined by Borgida and Serafini in [2], *Distributed Description Logics* provides a syntactical and semantical framework for formalization of multiple ontologies *pairwise* linked by semantic mappings. In DDL, ontologies correspond to description logic theories (T-boxes), while semantic mappings correspond to collections of *bridge rules* ($\mathfrak{B}$).

Given a non empty set I of indexes, used to identify ontologies, let $\{\mathcal{DL}_i\}_{i\in I}$ be a collection of description logics[3]. For each $i \in I$ let us denote a T-box of $\mathcal{DL}_i$ as $\mathcal{T}_i$. In this paper, we assume that each $\mathcal{DL}_i$ is description logic weaker or at most equivalent to $\mathcal{SHIQ}$. Thus a T-box will contain all the information necessary to define the terminology of a domain, including not just concept and role definitions, but also general axioms relating descriptions, as well as declarations such as the transitivity of certain roles.

We call $\mathbf{T} = \{\mathcal{T}_i\}_{i\in I}$ a family of T-Boxes indexed by I. Intuitively, $\mathcal{T}_i$ is the description logic formalization of the i-th ontology. To make every description distinct, we will prefix it with the index of ontology it belongs to. For instance, the concept C that occurs in the i-th ontology is denoted as $i : C$. Similarly, $i : C \sqsubseteq D$ denotes the fact that the axiom $C \sqsubseteq D$ is being considered in the i-th ontology.

Semantic mappings between different ontologies are expressed via collections of *bridge rules*.

Definition 1 (Bridge rules). *A bridge rule from i to j is an expression of the following two forms:*

1. $i : A \xrightarrow{\sqsupseteq} j : G$, onto-bridge rule
2. $i : B \xrightarrow{\sqsubseteq} j : H$, into-bridge rule

where A, B and G, H are concepts of $\mathcal{DL}_i$ and $\mathcal{DL}_j$ respectively[4].

Bridge rules do not represent semantic relations stated from an external *objective* point of view. Indeed, there is no such global view in the web. Instead, bridge rules from i to j express relations between i and j viewed from the *subjective* point of view of the j-th ontology.

Intuitively, the into-bridge rule $i : B \xrightarrow{\sqsubseteq} j : H$ states that, from the j-th point of view the concept B in i is less general than its local concept H. Similarly, the onto-bridge rule $i : A \xrightarrow{\sqsupseteq} j : G$ expresses the fact that, according to j, A in i is more general than G in j. Therefore, bridge rules from i to j provide the possibility of translating into j's ontology (under some approximation) the concepts of a foreign i's ontology. Note, that since bridge rules reflect a subjective point of view, bridge rules from j to i are not necessarily the inverse of the rules from i to j, and in fact there may be no rules in one or both the directions.

[3] We assume familiarity with Description Logic and related reasoning systems, described in [4].

[4] This is a restricted case of bridge rules w.r.t. definition in [2].

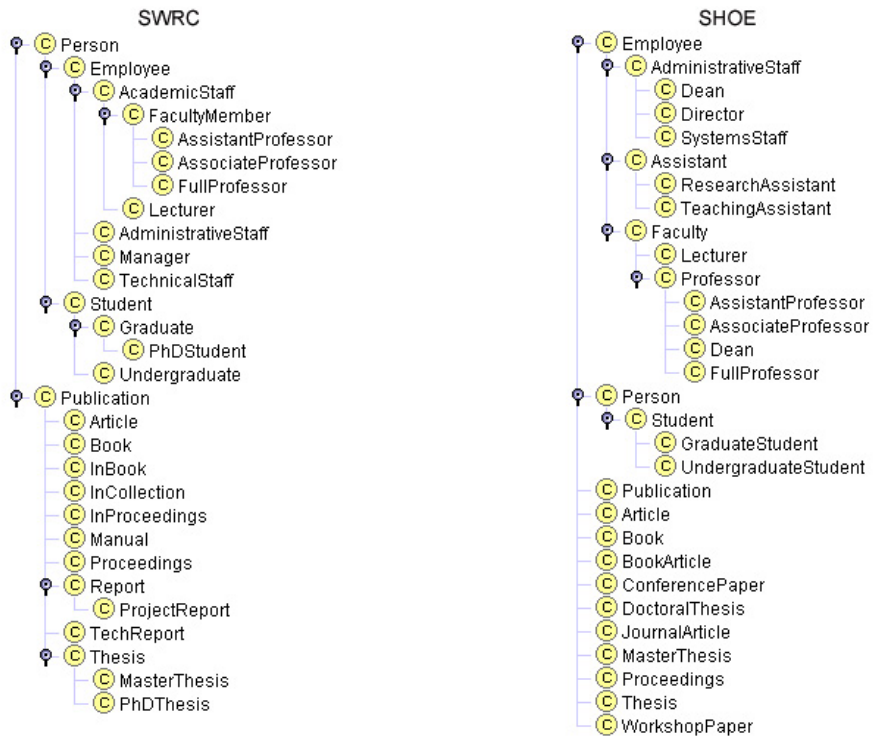

Fig. 1. Extracts of the class hierarchy of SWRC and SHOE.

Example 1. From on-line DAML ontology library we have selected two small and largely overlapping ontologies. First, the semantic web research community ontology (SWRC)[5] that models the research community, its researches, topics, publications, etc. Second, a DAML version of SHOE ontology for describing universities and the activities that occur at them[6]. Figure 1 shows an extract of the class hierarchy of these two ontologies.

The following are examples of bridge rules from SWRC to SHOE.

$$\text{SWRC} : \mathsf{Article} \xrightarrow{\sqsupseteq} \text{SHOE} : \mathsf{ConferencePapers} \tag{1}$$

$$\text{SWRC} : \mathsf{Article} \xrightarrow{\sqsubseteq} \text{SHOE} : \mathsf{Article} \tag{2}$$

$$\text{SWRC} : \mathsf{Article} \xrightarrow{\sqsupseteq} \text{SHOE} : \mathsf{Article} \tag{3}$$

$$\text{SWRC} : \mathsf{PhDStudent} \xrightarrow{\sqsupseteq} \text{SHOE} : \mathsf{GraduateStudent} \tag{4}$$

You can see a richer set of possible bridge rules between OWL encodings of SWRC and SHOE ontologies[7]. We have defined these bridge rules manually, but in many cases bridge rules can be produced by a (semi)-automatic process.

[5] `http://www.semanticweb.org/ontologies/swrc-onto-2000-09-10.daml`
[6] `http://www.cs.umd.edu/projects/plus/DAML/onts/univ1.0.daml`
[7] `http://trinity.dit.unitn.it/drago/examples/eswc05/swrc-shoe.cowl`

Definition 2 (Distributed T-box). *A distributed T-box (DTB)* $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \mathfrak{B}\rangle$ *consists of a collection of T-boxes* $\{\mathcal{T}_i\}_{i\in I}$*, and a collection of bridge rules* $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i\neq j\in I}$ *between them.*

The semantic of DDL is the customization of Local Models Semantics for Multi Context Systems [5, 19]. The basic idea is that each ontology $\mathcal{T}_i$ is *locally interpreted* on a *local domain*. The first component of the semantics of a DTB is therefore a family of interpretations $\{\mathcal{I}_i\}_{i\in I}$, one for each T-box $\mathcal{T}_i$. Each $\mathcal{I}_i$ is called a *local interpretation* and consists of *possibly empty* domain $\Delta^{\mathcal{I}_i}$ and a valuation function $\cdot^{\mathcal{I}_i}$, which maps every concept to a subset of $\Delta^{\mathcal{I}_i}$, every role to a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$. The interpretation on the empty domain is denoted with the apex ϵ.

Notice that, in DL, interpretations are defined always on a non empty domain. Therefore $\mathcal{I}^\epsilon$ is not an interpretation in DL. In DDL however we need to provide a semantics for *partially inconsistent* distributed T-boxes, i.e. DTBs in which some of the local T-boxes are inconsistent. $\mathcal{I}^\epsilon$ provides an "impossible interpretation" which can be associated to inconsistent T-boxes. Indeed, $\mathcal{I}^\epsilon$ satisfies every axiom $X \sqsubseteq Y$ (also $\top \sqsubseteq \bot$) since $X^{\mathcal{I}^\epsilon} = \emptyset$ for every concept and role X.

The second component of the DDL semantic is the family of domain relations.

Definition 3 (Domain relation). *A domain relation* r_{ij} *from* $\Delta^{\mathcal{I}_i}$ *to* $\Delta^{\mathcal{I}_j}$ *is a subset of* $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$*. We use* $r_{ij}(d)$ *to denote* $\{d' \in \Delta^{\mathcal{I}_j} \mid \langle d, d'\rangle \in r_{ij}\}$*; for any subset* D *of* $\Delta^{\mathcal{I}_i}$*, we use* $r_{ij}(D)$ *to denote* $\bigcup_{d\in D} r_{ij}(d)$*; for any* $R \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$ *we use* $r_{ij}(R)$ *to denote* $\bigcup_{\langle d,d'\rangle\in R} r_{ij}(d) \times r_{ij}(d')$*.*

Domain relation r_{ij} does not represent a semantic mapping seen from an external objective point of view. Rather, it represents a possible way of mapping the elements of $\Delta^{\mathcal{I}_i}$ into its domain $\Delta^{\mathcal{I}_j}$, seen from j's perspective. For instance, if $\Delta^{\mathcal{I}_1}$ and $\Delta^{\mathcal{I}_2}$ are the representation of time as Rationals and as Naturals, r_{ij} could be the round off function, or some other approximation relation.

Definition 4 (Distributed interpretation). *A* distributed interpretation $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i\neq j\in I}\rangle$ *of a DTB* $\mathfrak{T}$ *consists of local interpretations* $\mathcal{I}_i$ *for each* $\mathcal{T}_i$ *on local domains* $\Delta^{\mathcal{I}_i}$*, and a family of domain relations* r_{ij} *between these local domains.*

Definition 5. *A distributed interpretation* $\mathfrak{I}$ *satisfies the elements of a DTB* $\mathfrak{T}$ *according to the following clauses: for every* $i, j \in I$

1. $\mathfrak{I} \models i : A \sqsubseteq B$*, if* $\mathcal{I}_i \models A \sqsubseteq B$
2. $\mathfrak{I} \models \mathcal{T}_i$*, if* $\mathfrak{I} \models i : A \sqsubseteq B$ *for all* $A \sqsubseteq B$ *in* $\mathcal{T}_i$
3. $\mathfrak{I} \models i : x \xrightarrow{\sqsubseteq} j : y$*, if* $r_{ij}(x^{\mathcal{I}_i}) \subseteq y^{\mathcal{I}_j}$
4. $\mathfrak{I} \models i : x \xrightarrow{\sqsupseteq} j : y$*, if* $r_{ij}(x^{\mathcal{I}_i}) \supseteq y^{\mathcal{I}_j}$
5. $\mathfrak{I} \models \mathfrak{B}_{ij}$*, if* $\mathfrak{I}$ *satisfies all bridge rules in* $\mathfrak{B}_{ij}$
6. $\mathfrak{I} \models \mathfrak{T}$*, if for every* $i, j \in I$*,* $\mathfrak{I} \models \mathcal{T}_i$ *and* $\mathfrak{I} \models \mathfrak{B}_{ij}$

Definition 6 (Distributed Entailment and Satisfiability). $\mathfrak{T} \models i : C \sqsubseteq D$ *(read as "*$\mathfrak{T}$ *entails* $i : C \sqsubseteq D$*") if for every* $\mathfrak{I}$*,* $\mathfrak{I} \models \mathfrak{T}$ *implies* $\mathfrak{I} \models_d i : C \sqsubseteq D$*.*

$\mathfrak{T}$ *is* satisfiable *if there exists a* $\mathfrak{I}$ *such that* $\mathfrak{I} \models \mathfrak{T}$*. Concept* $i : C$ *is* satisfiable *with respect to* $\mathfrak{T}$ *if there is a* $\mathfrak{I}$ *such that* $\mathfrak{I} \models \mathfrak{T}$ *and* $C^{\mathcal{I}_i} \neq \emptyset$*.*

Some important properties of DDL are listed below:

Monotonicity Bridge rules do not obstruct local subsumptions. Formally:

$$\mathcal{T}_i \vDash A \sqsubseteq B \Longrightarrow \mathfrak{T} \models i : A \sqsubseteq B \tag{5}$$

Directionality T-box without incoming bridge rules is not affected by other T-boxes. Formally, if $\mathfrak{B}_{ki} = \emptyset$ for any $k \neq i \in I$, then:

$$\mathfrak{T} \models i : A \sqsubseteq B \Longrightarrow \mathcal{T}_i \vDash A \sqsubseteq B \tag{6}$$

Simple subsumption propagation A combination of onto- and into- bridge rules allows to propagate subsumptions across ontologies. For example, if $\mathfrak{B}_{ij}$ contains $i : A \xrightarrow{\sqsupseteq} j : G$ and $i : B \xrightarrow{\sqsubseteq} j : H$, then:

$$\mathfrak{T} \models_d i : A \sqsubseteq B \Longrightarrow \mathfrak{T} \models j : G \sqsubseteq H \tag{7}$$

Generalized subsumption propagation If $\mathfrak{B}_{ij}$ contains $i : A \xrightarrow{\sqsupseteq} j : G$ and $i : B_k \xrightarrow{\sqsubseteq} j : H_k$ for $1 \leq k \leq n$ and $n \geq 0$[8] , then:

$$\mathfrak{T} \models i : A \sqsubseteq \bigsqcup_{k=1}^{n} B_k \Longrightarrow \mathfrak{T} \models j : G \sqsubseteq \bigsqcup_{k=1}^{n} H_k \tag{8}$$

We would like to stress the importance of subsumption propagation property since it constitutes a main reasoning pattern in DDL. For the full set of DDL properties and formal proofs we refer reader to a technical report [20].

Example 2. Taking the bridge rules from Example 1 and applying subsumption propagation property we can infer in the hierarchy SHOE that ConferencePaper is a subclass of Article, i.e. that SHOE : ConferencePaper $\sqsubseteq$ Article.

Figure 2 shows how the initial SHOE hierarchy can be enriched (without its modification) via the whole set of possible bridge rules mentioned in Example 1.

3 Distributed tableau for reasoning in DDL

Although both in DL and DDL the fundamental reasoning task lays in a verification of concepts subsumption, in DDL besides the ontology the subsumption depends also on other ontologies that affect it through the semantic mappings. In this section we investigate a decision procedure that computes DDL subsumption and propose a distributed tableau reasoning algorithm for determining whether $\mathfrak{T} \models i : A \sqsubseteq B$.

In order to get the intuition of the algorithm, let us first present an example with some simplifying assumptions. Later on, we relax these assumptions and extend our results to a more general case.

[8] The condition that $n \geq 0$ is intentional. When $n = 0$ - we stipulate that $\bigsqcup_{k=1}^{n} D_k$ denotes $\bot$.

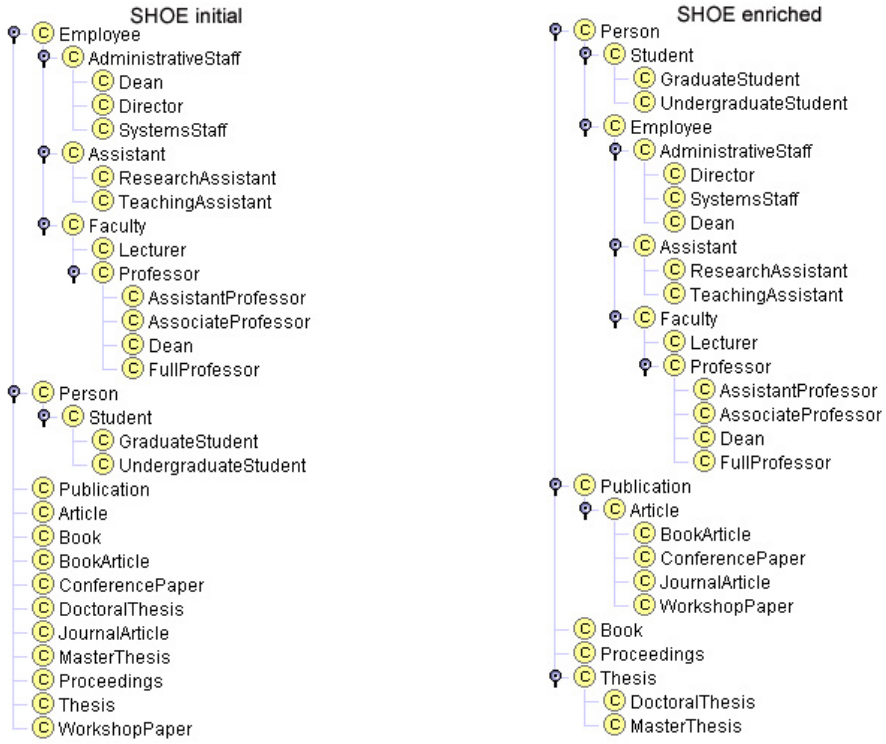

Fig. 2. Initial and enriched via bridge rules hierarchy of SHOE.

Example 3. Consider a distributed T-box $\mathfrak{T}_{12} = \langle \mathcal{T}_1, \mathcal{T}_2, \mathfrak{B}_{12} \rangle$ with only two T-boxes and unidirectional bridge rules between them. Suppose that $\mathcal{T}_1$ contains axioms $\mathsf{Student} \sqsubseteq \mathsf{Person}$ and $\mathsf{Pianist} \sqsubseteq \mathsf{Musician}$, $\mathcal{T}_2$ does not contain any axiom and $\mathfrak{B}_{12}$ contains the following bridge rules:

$$1 : \mathsf{Person} \xrightarrow{\sqsubseteq} 2 : \mathsf{Agent} \qquad 1 : \mathsf{Musician} \xrightarrow{\sqsubseteq} 2 : \mathsf{Artist} \tag{9}$$

$$1 : \mathsf{Student} \xrightarrow{\sqsupseteq} 2 : \mathsf{Graduate} \qquad 1 : \mathsf{Pianist} \xrightarrow{\sqsupseteq} 2 : \mathsf{JazzPianist} \tag{10}$$

Let us show that $\mathfrak{T}_{12} \models 2 : \mathsf{Graduate} \sqcap \mathsf{JazzPianist} \sqsubseteq \mathsf{Agent} \sqcap \mathsf{Artist}$, i.e. that for any distributed interpretation $\mathfrak{I} = \langle \mathcal{I}_1, \mathcal{I}_2, r_{12} \rangle$, $(\mathsf{Graduate} \sqcap \mathsf{JazzPianist})^{\mathcal{I}_2} \subseteq (\mathsf{Agent} \sqcap \mathsf{Artist})^{\mathcal{I}_2}$.

1. Suppose that by contradiction there is an $x \in \Delta_2$ such that $x \in (\mathsf{Graduate} \sqcap \mathsf{JazzPianist})^{\mathcal{I}_2}$ and $x \notin (\mathsf{Agent} \sqcap \mathsf{Artist})^{\mathcal{I}_2}$.
2. Then $x \in \mathsf{Graduate}^{\mathcal{I}_2}$, $x \in \mathsf{JazzPianist}^{\mathcal{I}_2}$, and either $x \notin \mathsf{Agent}^{\mathcal{I}_2}$ or $x \notin \mathsf{Artist}^{\mathcal{I}_2}$.
3. Let us consider the case where $x \notin \mathsf{Agent}^{\mathcal{I}_2}$. From the fact that $x \in \mathsf{Graduate}^{\mathcal{I}_2}$, by the bridge rule (10), there is $y \in \Delta^{\mathcal{I}_1}$ with $\langle y, x \rangle \in r_{12}$, such that $y \in \mathsf{Student}^{\mathcal{I}_1}$.
4. From the fact that $x \notin \mathsf{Agent}^{\mathcal{I}_1}$, by bridge rule (9), we can infer that for all $y \in \Delta^{\mathcal{I}_1}$ if $\langle y, x \rangle \in r_{12}$ then $y \notin \mathsf{Person}^{\mathcal{I}_1}$.
5. But, since $\mathsf{Student} \sqsubseteq \mathsf{Person} \in \mathcal{T}_1$, then $y \in \mathsf{Person}^{\mathcal{I}_1}$, and this is a contradiction.

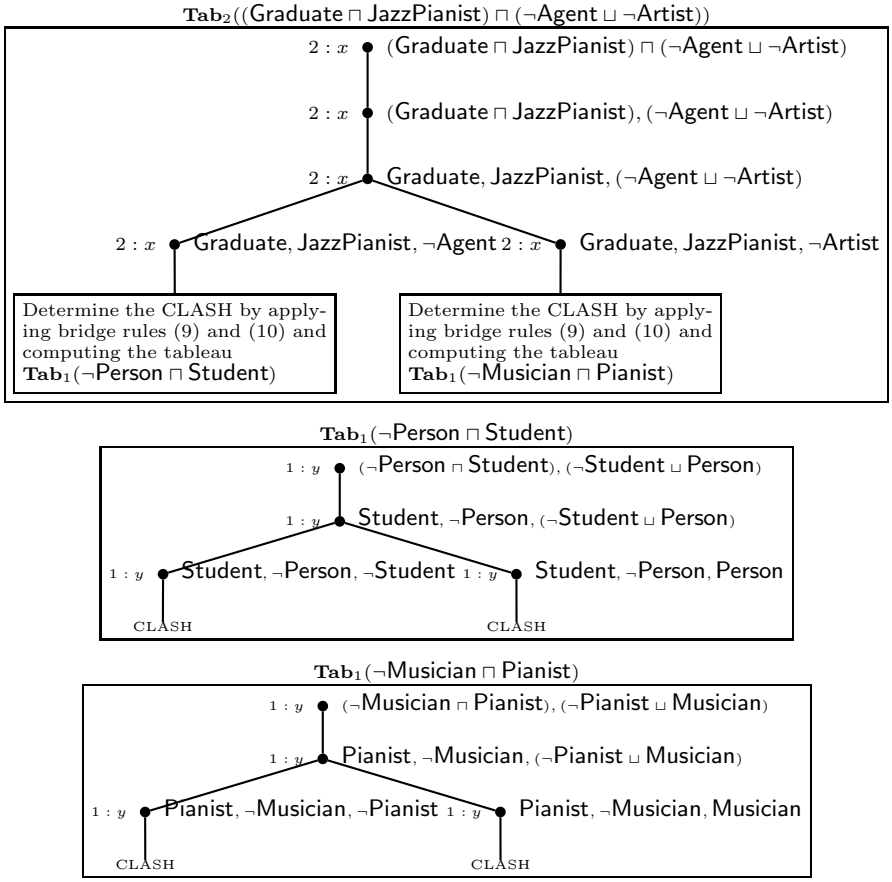

Fig. 3. Illustration of tableaux combination for DDL subsumption.

6. The case where $x \notin \mathsf{Artist}^{\mathcal{I}_2}$ is analogous.

The above reasoning steps can be seen as a combination of a tableau $\mathbf{Tab}_2$ in $\mathcal{T}_2$ with a tableau $\mathbf{Tab}_1$ in $\mathcal{T}_1$ as it is illustrated in Figure 3.

Let us formalize the above example.

Definition 7 (Propagation operator). *Given a set of bridge rules $\mathfrak{B}_{12}$ from $\mathcal{DL}_1$ to $\mathcal{DL}_2$, the operator $\mathfrak{B}_{12}(\cdot)$, taking as input a T-box in $\mathcal{DL}_1$ and producing a T-box in $\mathcal{DL}_2$, is defined as follows:*

$$\mathfrak{B}_{12}(\mathcal{T}_1) = \left\{ G \sqsubseteq \bigsqcup_{k=1}^{n} H_k \;\middle|\; \begin{array}{l} \mathcal{T}_1 \models A \sqsubseteq \bigsqcup_{k=1}^{n} B_k, \\ 1 : A \xrightarrow{\sqsupseteq} 2 : G \quad \in \mathfrak{B}_{12}, \\ 1 : B_k \xrightarrow{\sqsubseteq} 2 : H_k \in \mathfrak{B}_{12}, \\ \textit{for } 1 \leq k \leq n,\ n \geq 0^{9} \end{array} \right\}$$

[9] When $n = 0$ - we stipulate that $\bigsqcup_{k=1}^{n} D_k$ denotes $\bot$.

Theorem 1 (Soundness and completeness). *Let* $\mathfrak{T}_{12} = \langle \mathcal{T}_1, \mathcal{T}_2, \mathfrak{B}_{12} \rangle$ *be a distributed T-box, then:*

$$\mathfrak{T}_{12} \models 2 : X \sqsubseteq Y \Longleftrightarrow \mathcal{T}_2 \cup \mathfrak{B}_{12}(\mathcal{T}_1) \models X \sqsubseteq Y \tag{11}$$

For the formal proof of Theorem 1 we refer reader to a technical report [20].

The main message of Theorem 1 is that in DDL the decision whether $\mathfrak{T}_{12} \models 2 : X \sqsubseteq Y$ can be *correctly* and *completely* rephrased into a *standard* DL subsumption in $\mathcal{T}_2$ extended by application of the propagation operator $\mathfrak{B}_{12}(\cdot)$. Due to that, the main computational task of DDL subsumption algorithm is to calculate the application of the propagation operator.

Theorem 1 can be generalized to an *acyclic distributed T-box*, i.e. any $\mathfrak{T}$ in which the set of indexes I is a partial order $\langle I, < \rangle$ such that $i < j$ if and only if $\mathfrak{B}_{ij} \neq \emptyset$. Generalized version of the DDL subsumption algorithm represents a *backward-chaining* method that checks standard subsumption in a T-box $\mathcal{T}_i$ extended by applying propagation operators to the T-boxes which affect $\mathcal{T}_i$ via bridge rules.

3.1 Description of the algorithm

Similarly to description logics reduction of subsumption to unsatisfiability, we rephrase the problem of deciding whether $\mathfrak{T} \models i : A \sqsubseteq B$ into the problem of not finding a distributed interpretation $\mathfrak{I}$ of $\mathfrak{T}$, such that $(A \sqcap \neg B)^{\mathcal{I}_i} \neq \emptyset$.

Given an *acyclic distributed T-box* $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i \neq j \in I} \rangle$, we suppose to have a set of procedures $\mathbf{Tab}_i$, one for each $i \in I$. Given a concept C, $\mathbf{Tab}_i$ returns a completion tree T which is a finite representation of a local *tableau*[10] for C in $\mathcal{T}_i$. On top of each procedure $\mathbf{Tab}_i$ we define a distributed tableau procedure $\mathbf{dTab}_i$, one for each $i \in I$. Given a concept Φ, $\mathbf{dTab}_i$ returns the result of it DDL (un)satisfiability test.

The Algorithm 1 illustrates suggested distributed tableau procedure. Roughly speaking, the idea behind the distributed tableau is to check whether there are bridge rules capable of closing a local tableau. If the distributed tableau fails to close initial local tableau then the algorithm returns "satisfiable", otherwise it returns "unsatisfiable".

To verify whether the concept Φ is satisfiable in a T-box $\mathcal{T}_j$ of acyclic distributed T-box $\mathfrak{T}$, the distributed algorithm proceeds as follows. First, it applies a local tableau procedure $\mathbf{Tab}_j$ in order to build a local completion tree T. According to the tableau algorithm, each node x introduced during creation of the completion tree is labeled with a function $L(x)$ containing concepts that x must satisfy. If the tree is clashed then we have nothing to do because the concept Φ is unsatisfiable locally. If the tree is not clashed then the algorithm traverses the nodes of open brunches in T and further checks whether the brunches can be closed due to the bridge rules. For example, to check the node x the algorithm looks for the bridge rules that can potentially enrich $L(x)$ and cause a clash in x. If there are such bridge rules, they are verified by calling corresponding foreign distributed tableau procedures. The algorithm completes when all opened brunches in the initial completion tree T are verified.

[10] We use the notion of $\mathcal{SHIQ}$-tableau defined in [14].

Algorithm 1 Distributed tableau procedure

dTab$_j(\Phi)$

BEGIN
T=**Tab**$_j(\Phi)$; {perform local reasoning and create a completion tree}
if (T is not clashed) **then**
  for each open branch β in T **do**
    repeat
      select node $x \in \beta$ and an $i \neq j$;
      $\mathbb{C}_i^{onto}(x) = \{C \mid i : C \xrightarrow{\sqsupseteq} j : D,\ D \in L(x)\}$;
      $\mathbb{C}_i^{into}(x) = \{C \mid i : C \xrightarrow{\sqsubseteq} j : D, \neg D \in L(x)\}$;
      if ($\mathbb{C}_i^{onto}(x) \neq \emptyset$) **then**
        for each $C \in \mathbb{C}^{onto}$ **do**
          if (**dTab**$_i(C \sqcap \neg \bigsqcup \mathbb{C}_i^{into})$ is not satisfiable)[11] **then**
            close β; {clash in x}
            break; {verify next branch}
          end if
        end for
      end if
    until ((β is open) and (there exist not verified nodes in β))
  end for{all branches are verified}
end if
if (T is clashed) **then**
  return unsatisfiable;
else
  return satisfiable;
end if
END

The proposed algorithm has several limitations. It admits acyclic distributed T-boxes without individuals and only allows bridge rules connecting atomic concepts. Despite these restrictions, we see the main advantage of the algorithm in its implementation simplicity. Furthermore, the space complexity of the distributed tableau procedure is the same as the space complexity of the local tableau algorithm.

4 DRAGO reasoning system

In this section we will describe a design and implementation principles that lay in the base of DRAGO, the system for reasoning with multiple ontologies interconnected by semantic mappings.

4.1 Vision

As depicted in Figure 5, DRAGO envisages a web of ontologies distributed amongst a peer-to-peer network of *DRAGO Reasoning Peers* (DRP).

[11] When $\mathbb{C}_i^{into} = \emptyset$ we stipulate that $\bigsqcup \mathbb{C}_i^{into}$ denote $\bot$.

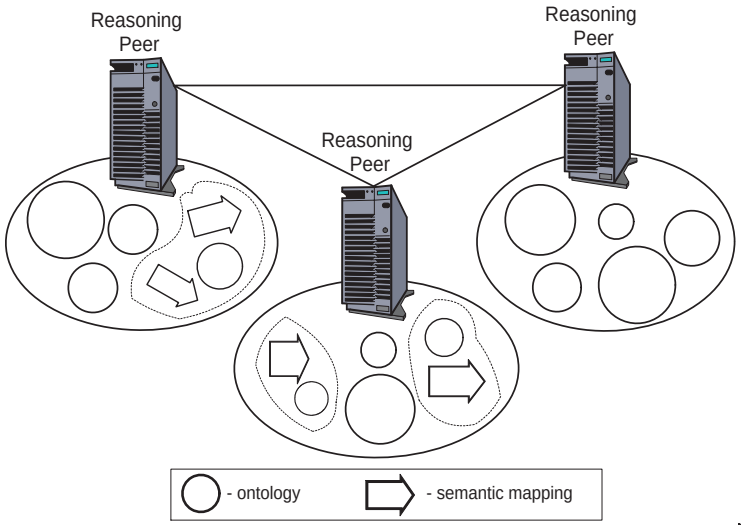

Fig. 4. DRAGO vision.

The role of a DRP is to provide reasoning services for ontologies with mappings registered to it as well as to request reasoning services of other DRPs when this is required for fulfilling a distributed reasoning task.

In order to register an ontology to a DRP the users specify a logical identifier for it, a URI, and give a physical location of ontology on the web, a URL. Besides that, it is possible to assign to the ontology semantic mappings, providing in the same manner their location on the web. As we discussed in the previous sections, attaching mappings to ontology enriches its knowledge due to the subsumption propagation mechanism. To prevent the possibility of attaching malicious mappings that can obstruct or falsify reasoning services, only the user that registered this ontology is allowed to add mappings to it.

When users or applications want to perform reasoning with a registered ontology they refer to the corresponding DRP and invoke its reasoning services giving the URI to which the ontology was bound.

4.2 Architecture

A DRP constitutes the basic element of DRAGO. The major components of a DRP are depicted in Figure 5.

A DRP has two interfaces which can be invoked by users or applications:

- A *Registration Service* interface is meant for creating/modifying/deleting of registrations of ontologies and mappings assigned to them.
- A *Reasoning Services* interface enables the calls of reasoning services for registered ontologies. Among the reasoning services can be a possibility to check ontology consistency, build classification, verify concepts satisfiability and check entailment.

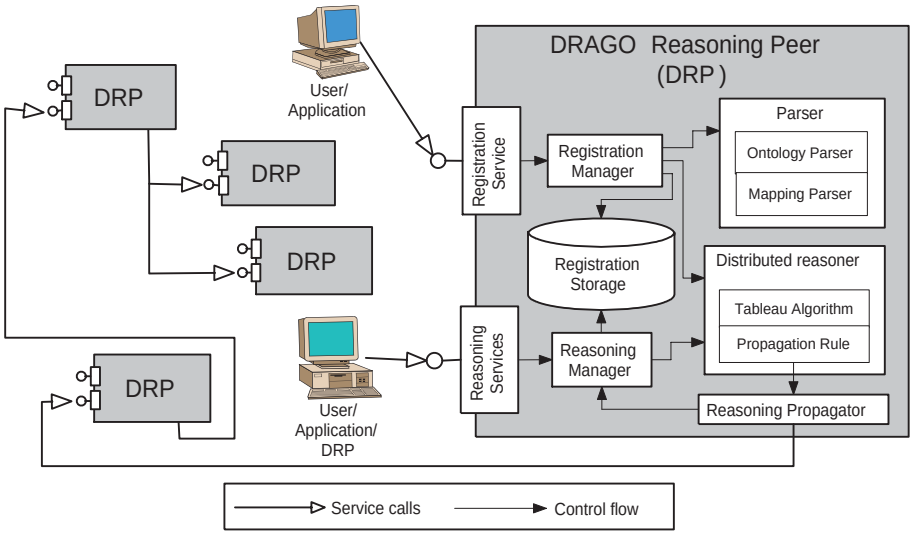

Fig. 5. DRAGO architecture.

All accessibility information about registered ontologies and mappings is stored by a DRP in its local *Registration Storage*.

In order to register an ontology with a collection of semantic mappings attached to it (both available on the web) a user or application invokes the Registration Service of a DRP and sends to it the following registration information:

- URI to which the ontology will be bound.
- URLs of ontology and semantic mappings attached to it, if any.
- If the semantic mappings connect this ontology with ontologies registered to external DRPs then additionally the URLs of these DRPs should be specified. This requirement is explained by the necessity to know who is responsible for reasoning with these ontologies.

The Registration Service interface is implemented by the *Registration Manager*. When the Manager receives a registration request, it (i) consults the Registration Storage and verifies if the URI has not occupied yet, (ii) if not it accesses ontologies and assigned mappings from their URLs, (iii) asks Parser component to process them, (iv) initializes the Distributed Reasoner with the parsed data, and (v) finally adds a new record to the Registration Storage.

The *Parser* component translates ontologies and mappings source files to the internal format used by the Distributed Reasoner. For doing that, the Parser consist from two sub components: the ontology parser, tailored on ontology language formats (for example, OWL[13]), and the mapping parser, tailored on mapping formats (for example, C-OWL[3]).

The *Reasoning Manager* component implements the Reasoning Services interface. When users, applications or other DRPs invoke this interface sending the URI of requested ontology, the Manager verifies with the Registration Storage whether the URI is registered to the DRP and, if yes, asks the Distributed Reasoner to execute corresponding reasoning task for that ontology.

The *Distributed Reasoner* represents a brain of a DRP. It realizes the distributed algorithm proposed in the Section 3 and reasons on ontologies with attached mappings that are registered to the DRP. The Distributed Reasoner is built on top of standard tableau reasoner whose algorithm was extended with a Propagation Rule in accordance with the distributed tableau algorithm. When the Propagation Rule is applied it analyses semantic mappings and possibly generates reasoning tasks that are required to be executed in the ontologies participating in mappings.

To dispatch reasoning tasks generated by a Distributed Reasoner to the responsible reasoners, the *Reasoning Propagator* component refers to the Reasoning Manager and either dispatches reasoning to the local Distributed Reasoner or sends out a request of reasoning service to the corresponding external DRP.

4.3 Implementation

The described DRAGO architecture was implemented by us for the case of OWL[1] ontology space. For expressing semantic mappings between OWL ontologies we use a C-OWL[3]. According to C-OWL, mapping consists of references to the source and target ontologies and a series of bridge rules relating classes between these ontologies. Due to the limitations of introduced distributed tableau algorithm (see Section 3) among the possible C-OWL bridge rule types DRAGO supports the use of $\equiv, \sqsubseteq, \sqsupseteq$ rules connecting atomic concepts.

A Distributed Reasoner was implemented as an extension to an open source OWL reasoner Pellet[12]. Originally, Pellet parses OWL ontology to a Knowledge Base (T-box/A-box). To satisfy the needs of DRAGO we extended a Pellet's Knowledge Base with a M-box containing parsed C-OWL mappings. Another extension of Pellet was done by adding a Propagation Rule to the core tableau algorithm in order to transform it to the distributed tableau. This rule is called for every node created by the core tableau algorithm and consist in finding such bridge rules in M-box that potentially capable of importing new subsumptions from mapping-related ontologies. The distributed tableau algorithm was implemented in a straightforward way without advanced optimization techniques as, for example, caching.

DRAGO is implemented to operate over HTTP and access ontologies and mappings published on the web. A DRP represents several java servlets that should be deployed to a java-enabled web server, for example Tomcat[13].

5 Related work

From a theoretical perspective, presented work is an extension of the results introduced in [2].

DDL inherited a lot of ideas from the other logics for distributed systems, among them Multi Context Systems (MCS) [7], the general framework for contextual reasoning, propositional MCS [8, 5, 19] and Distributed First Order Logics (DFOL) [6].

[12] `http://www.mindswap.org/2003/pellet`
[13] `http://jakarta.apache.org/tomcat`

In [15], it has been shown that DDL can be represented in a much richer theoretical framework for integrating different logics, called $\mathcal{E}$-connections. $\mathcal{E}$-connections allow to state relations between a set of logical frameworks using *n-ary link relations*. Bridge rules can be seen as a special case of binary link relations. In this case, the satisfiability problem for DDL can be reduced to the satisfiability problem for basic $\mathcal{E}$-connections.

The combined tableau algorithm for the restricted case of $\mathcal{E}$-connections has been proposed recently in [9]. In contrast to the distributed tableau algorithm proposed in this paper, as described in [9] implemented combined tableau for $\mathcal{E}$-connections is rather a selective global approach organized in a single reasoner, whereas in distributed tableau we combine different reasoning procedures for mapped ontologies.

The idea of having the system providing reasoning services for ontologies on the Semantic Web is not new. There are a number of reasoning servers based on the state of the art reasoners like RACER[10] or FaCT[12]. What makes DRAGO different from them is the capability of reasoning with ontologies coupled with semantic mappings using a distributed algorithm. While these reasoning servers are tightly connected with ontology repositories for achieving a higher level of optimization, DRAGO is a lightweight implementation which directly uses ontologies and mappings published on the web.

Also from the practical point of view, DRAGO architecture can be related to a variety of systems for mediation and integration of distributed heterogeneous sources like Piazza[11], OBSERVER[17] and others.

6 Conclusions and future work

In this paper, we have introduced DRAGO, a system which provides reasoning services for multiple OWL ontologies interconnected via C-OWL mappings[14].

The theoretical support of DRAGO is provided by the Distributed Description Logics framework. In this paper, we have described a sound and complete distributed tableau reasoning technique for DDL. According to it, a reasoning in DDL can be fulfilled by a suitable combination of existing local tableaux for Description Logics. Although the suggested reasoning algorithm has been considered for the limited case of DDL with acyclic distributed T-box without individuals and bridge rules connecting atomic concepts, we see a main benefit of the algorithm in its simplicity and easy implementation on top of existing tableau-based reasoning systems.

As promising paths for further research we plan to extend our results for general distributed T-boxes, investigate the use of more expressive mappings connecting complex concepts, and finally explore the caching techniques for improving the distributed algorithm.

References

1. G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In *Handbook on Ontologies in Information Systems*, pages 67–92, 2003.

[14] Evaluation version of DRAGO is available for download after the registration on the project home page `http://trinity.dit.unitn.it/drago`

2. A. Borgida and L. Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *Journal of Data Semantics*, pages 153–184, 2003.
3. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-owl: Contextualizing ontologies. In *Proc. of the 2d International Semantic Web Conference (ISWC2003)*, pages 164–179, 2003.
4. F.Baader, D.Calvanese, D.McGuinness, D.Nardi, and P.F.Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
5. C. Ghidini and F. Giunchiglia. Local Model Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.
6. C. Ghidini and L. Serafini. Distributed First Order Logics. In *Proc. of the Frontiers of Combining Systems*, pages 121–139, 2000.
7. F. Giunchiglia. Contextual Reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364, 1993.
8. F. Giunchiglia and L. Serafini. Multilanguage Hierarchical Logics (or: How we can do without modal logics). *Artificial Intelligence*, 65(1):29–70, 1994.
9. B. C. Grau, B. Parsia, and E. Sirin. Working with Multiple Ontologies on the Semantic Web. In *Proc. of the 3d International Semantic Web Conference (ISWC2004)*, 2004.
10. V. Haarslev and R. Moller. Racer System Description. In *Proc. of the International Joint Conference on Automated Reasoning (IJCAR2001)*, pages 701–706, 2001.
11. A. Halevy, Z. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *Proc. of the 12th International World Wide Web Conference (WWW 2003)*, 2003.
12. I. Horrocks and P. F. Patel-Schneider. FaCT and DLP. In *Proc. of the Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, pages 27–30, 1998.
13. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1):7–26, 2003.
14. I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for very Expressive Description Logics. *Logic Journal of IGPL*, 8(3):239–263, 2000.
15. O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. $\mathcal{E}$-connections of Abstract Description Systems. *Artificial Intelligence*, 156(1):1–73, 2004.
16. A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA - a Mapping Framework for Distributed Ontologies. In *Proc. of Knowledge Engineering and Knowledge Management (EKAW-02)*, volume 2473 of *Lecture Notes in Computer Science*. Springer, 2002.
17. E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. *International Journal on Distributed and Parallel Databases (DAPD), ISSN 0926-8782*, 8(2):223–272, April 2000.
18. B. Omelayenko. RDFT: A Mapping Meta-Ontology for Business Integration. In *Proc. of the Workshop on Knowledge Transformation for the Semantic for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW2002)*, pages 77–84, 2002.
19. L. Serafini and F. Giunchiglia. ML Systems: A Proof Theory for Contexts. *Journal of Logic, Language and Information*, 11(4):471–518, 2002.
20. L. Serafini, A. Tamilin, and A. Borgida. Distributed Description Logics. Technical report, ITC-IRST, 2004.