Towards Distributed Tableaux Reasoning Procedure for DDL with Increased Subsumption Propagation between Remote Ontologies

Martin Homola[1] **Luciano Serafini**[2]

[1]Faculty of Mathematics, Physics and Informatics,
Comenius University, Mlynska dolina, 842 48 Bratislava, Slovakia,
Email: homola@fmph.uniba.sk

[2] FBK-IRST, Via Sommarive 18, 38050 Povo, Trento, Italy
Email: serafini@fbk.eu

Abstract

Distributed Description Logics (DDL) enable reasoning with multiple ontologies interconnected by directional semantic mapping. In DDL semantic mapping is indicated by so called bridge rules. There are two kinds: into- and onto-bridge rules. Mappings between concepts allow subsumption to propagate from one ontology to another. However, in some cases, especially when more than two local ontologies are involved, subsumption does not always propagate as we would expect. In a recent study, an adjusted semantics has been introduced that is able to cope with more complex scenarios. In particular, subsumption propagates along chains of several bridge rules under this new semantics. This study makes use of so called compositional consistency requirement that has been employed before in Package-based description logics. While the results concerning subsumption propagation under the adjusted semantics are encouraging, we show in this paper that this semantics also has drawbacks. In certain situations it violates the directionality principle. We propose a weaker version of the semantics in this paper that is able to cope with chains of onto-bridge rules but it is not able to deal with chains of into-bridge rules. Furthermore we provide a sound and complete tableaux reasoning algorithm for this semantics.

1 Introduction and Motivation

Distributed Description Logics (DDL) have been introduced by Borgida and Serafini in (Borgida & Serafini 2003) and later developed in (Serafini & Tamilin 2004, Serafini et al. 2005). DDL are intended especially to enable reasoning over systems of multiple ontologies connected by directional semantic mapping, built upon the formal, logical and well established framework of Description Logics (DL) (Baader et al. 2003). DDL capture the idea of importing and reusing concepts between several ontologies. This idea combines well with the basic assumption of the Semantic Web that no central ontology but rather many ontologies with redundant knowledge will exist (Berners-Lee et al. 2001).

In DDL special syntactic constructs, dubbed bridge rules, are introduced for encoding semantic mapping between ontologies in a formal fashion. With bridge rules one is able to assert that some concept, say C, local to ontology $\mathcal{T}_1$, is mapped to an independent ontology $\mathcal{T}_2$ as a subconcept/superconcept of some $\mathcal{T}_2$-local concept, say D. Moreover, bridge rules are directed, and hence if there is a bridge rule with direction from $\mathcal{T}_1$ to $\mathcal{T}_2$, then $\mathcal{T}_2$ reuses knowledge from $\mathcal{T}_1$ but not necessarily the other way around. Semantic mapping encoded with bridge rules enables for knowledge reuse. In particular, we talk about subsumption propagation from one ontology to another. Consider the example depicted in Fig. 1. Let the ontology on the right be our local back-yard ontology. In this ontology we maintain knowledge about all animals in our backyard. Instead of complex modeling of all relations between our animals we reuse a far more complex ontology that contains classification of all animal species. We map MyCat to genus Felis, and we also create the concept DangerousAnimal and map it to Felidae, the family of all cats, since we also keep a hamster and we know that all cats hunt hamsters for food. Thanks to the semantics of DDL we are now able to derive that MyCat is a subconcept of DangerousAnimal, even if this relation is not derived in the original backyard ontology locally. We say that the subsumption between Felis and Felidae has propagated to the backyard ontology thanks to the mapping.

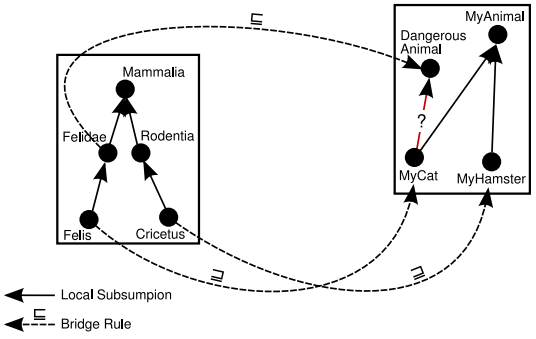

Figure 1: A DDL knowledge base with bridge rules. In the ontology on the right a query is marked up with "?"; we query whether MyCat is a subconcept of DangerousAnimal. This query is answered "yes" under the original DDL semantics.

Please note also that the direction of the mapping is from the species ontology to the backyard ontology, as we only intend to reuse the knowledge of the species ontology within the backyard ontology and not the other way around. The mapping depicted on Fig. 1 does not affect the knowledge within the species ontology in any way.

Subsumption propagation, as seen above, has been described as desired and one of the main features of DDL and it has been studied (cf. (Borgida & Serafini 2003, Serafini & Tamilin 2004, Serafini et al. 2005,

Homola 2007, 2008)). In (Homola 2008) we have argued that the original semantics of DDL behaves counter-intuitively in certain cases and we have provided an adjustment to the semantics in order to cope with this issue. For illustration consider Fig. 2 which extends the previously discussed example depicted in Fig. 1. In this case, the situation is more complex. We no longer map between concepts Felidae and DangerousAnimal, but instead these two concepts are connected indirectly via the concept Carnivore from yet another ontology. This third ontology deals with animal behaviour in contrast with the species ontology that merely provides classification. In case that there are such ontologies available, we suggest that this way of modelling is even more natural. As in the above example, again we would expect to derive that MyCat is a subconcept of DangerousAnimal within the backyard ontology. This relation is not derived under the original DDL semantics however.

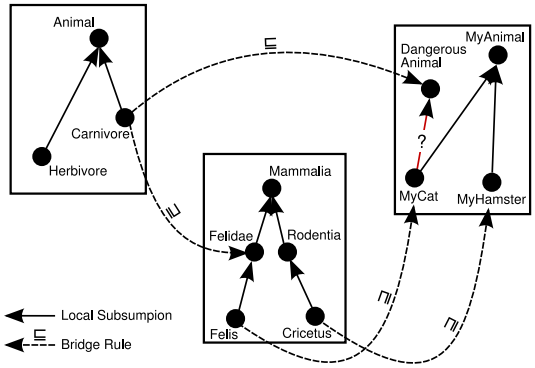

Figure 2: An example of complex concept mapping between three ontologies. In this case the query whether MyCat is a subconcept of DangerousAnimal is not true under the original DDL semantics.

In the example from Fig. 2 there is a so called chain of bridge rules. The onto-bridge rule between concepts Carnivore and Felidae and the other one between Felis and MyCat form a chain, as Felis is a subconcept of Felidae in the classification ontology. Similar chains are also possible with into-bridge rules. The adjusted semantics of (Homola 2008) allows subsumption propagation along such chains of bridge rules and hence the subsumption between MyCat and DangerousAnimal is entailed under this semantics.

In this paper we take steps forward in order to develop a distributed tableaux reasoning algorithm that would decide satisfiability of concepts and subsumption with respect to the adjusted semantics. The algorithm that is introduced in this paper handles chaining onto-bridge rules correctly, but it is unable to cope with chaining into-bridge rules. We provide precise characterization of the semantics that the algorithm actually implements. The algorithm works with acyclic DDL knowledge bases build on top of $\mathcal{ALC}$ as the local language. As in the case of the tableaux algorithm that is known for the original semantics (Serafini et al. 2005), the algorithm is truly distributed and it permits the scenario where every local ontology is governed by an autonomous reasoning service and these services communicate by passing queries. The local reasoner that has started the computation collects all the answers and makes the decision at the end. Besides the fact that each algorithm works under a different semantics, the main distinguishing feature of the newly introduced algorithm is that communication between local reasoners is divided into multiple messages. We believe that such behaviour may serve as a base for more fine-grained optimization in the future.

2 Distributed Description Logics

Distributed Description Logics have been introduced in (Borgida & Serafini 2003), as a formal framework for reasoning with multiple local ontologies interconnected with semantic mapping. Faithful to their namesake, DDL rely on Description Logics (Baader et al. 2003) as for the representation language of the local knowledge bases. DL provide a well established ontological representation language with formal semantics and with reasoning tasks and associated computational issues well understood. In the following we briefly introduce the $\mathcal{ALC}$ DL in oder to be able to build on it later in the paper. DDL, however, are also built over more expressive DL up to $\mathcal{SHIQ}$ (Horrocks et al. 1999).

A basic form of DL knowledge base is a TBox. We will denote TBoxes with $\mathcal{T}_i$ because we deal with several ontologies. Assume that i is an index from some index set. Each TBox is a set of subsumption axioms called general concept inclusions (GCIs), each of the form $i : C \sqsubseteq D$, where C and D are concepts. Concepts are either atomic or complex. Atomic concepts have no further structure. Complex concepts are composed from atomic concepts and roles using some of the DL concept constructors. For list of constructors and their meaning see Fig. 3. The semantics of DL is model-theoretic. Each ontology $\mathcal{T}_i$ is assigned an interpretation $\mathcal{I}_i$ consisting of a domain $\Delta^{\mathcal{I}_i}$ and an interpretation function $\cdot^{\mathcal{I}_i}$ which interprets each concept C by $C^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i}$. Each role R is interpreted by $R^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$. Complex concepts must satisfy further semantic constraints, see Fig. 3. There are also two special concepts $\bot$ and $\top$. The bottom concept ($\bot$) is always interpreted by $\bot^{\mathcal{I}_i} = \emptyset$, the top concept is always interpreted by $\top^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i}$, but they are formally defined as syntactic sugar: $\bot \equiv E \sqcap \neg E$ and $\top \equiv E \sqcup \neg E$, for some new concept name E. A concept C is in negation normal form (NNF) if negation ($\neg$) only appears in C directly in front of atomic concepts. For each concept, an equivalent concept that is in NNF exists (Baader et al. 2003), we denote NNF of C by $\mathrm{nnf}(C)$. Finally, a GCI axiom $i : C \sqsubseteq D$, is satisfied by $\mathcal{I}_i$ whenever $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. An interpretation $\mathcal{I}_i$ is a model of $\mathcal{T}_i$ if it satisfies all GCIs in $\mathcal{T}_i$. And, subsumption $i : C \sqsubseteq D$ is entailed by $\mathcal{T}_i$ if $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$ holds in all models of $\mathcal{T}_i$. Satisfiability of concepts and subsumption entailment are standard decision problems that are investigated in each DL. For more detailed introduction to DL please refer to (Baader et al. 2003, Horrocks et al. 1999).

E	$E^{\mathcal{I}_i}$
$\neg C$	$\Delta^{\mathcal{I}_i} \setminus C^{\mathcal{I}_i}$
$C \sqcap D$	$C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}$
$C \sqcup D$	$C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i}$
$\forall R.C$	$\{i \in \Delta^{\mathcal{I}_i} \mid (\forall j \in \Delta^{\mathcal{I}_i})\ (i,j) \in R^{\mathcal{I}_i} \implies j \in C^{\mathcal{I}_i}\}$
$\exists R.C$	$\{i \in \Delta^{\mathcal{I}_i} \mid (\exists j \in \Delta^{\mathcal{I}_i})\ (i,j) \in R^{\mathcal{I}_i} \wedge j \in C^{\mathcal{I}_i}\}$

Figure 3: Complex concepts in $\mathcal{ALC}$ and their semantics.

A DDL knowledge base, commonly dubbed distributed TBox $\mathfrak{T}$, includes a set of local TBoxes each over its own DL language and a set of bridge rules $\mathfrak{B}$ that provides mappings between local TBoxes. The local languages are commonly required to be under $\mathcal{SHIQ}$ (Horrocks et al. 1999) as it is not trivial to extend DDL with nominals (Serafini et al. 2005). In this work, we use the formal definition as follows.

Definition 1. *Assume a DL language $\mathcal{L}$ – a sub-language of $\mathcal{SHIQ}$, a non-empty index set I, a set of concept names $\mathrm{N_C} = \bigcup_{i\in I} \mathrm{N_{C}}i$ and a set of role names $\mathrm{N_R} = \bigcup_{i\in I} \mathrm{N_{R}}i$. A Distributed TBox over $\mathcal{L}$ is a pair $\mathfrak{T} = \langle \{\mathcal{T}_i\}_{i\in I}, \mathfrak{B}\rangle$ such that:*

1. *Each local TBox $\mathcal{T}_i$ is a collection of general concept inclusions (GCIs) over $\mathrm{N_C}i$ and $\mathrm{N_R}i$ in the local language of $\mathcal{T}_i$, a sub-language of $\mathcal{L}$. Each GCI is an axiom of the form*

$$i : C \sqsubseteq D \ .$$

2. *The set of bridge rules $\mathfrak{B}$ divides into sets of bridge rules $\mathfrak{B} = \bigcup_{i,j\in I, i\neq j} \mathfrak{B}_{ij}$. Each $\mathfrak{B}_{ij}$ is a collection of bridge rules in direction from $\mathcal{T}_i$ to $\mathcal{T}_j$ which are of two forms,* into-*bridge rules and* onto-*bridge rules (in the respective order):*

$$i : A \xrightarrow{\sqsubseteq} j : G \ , \qquad i : B \xrightarrow{\sqsupseteq} j : H \ .$$

As we have already mentioned in the introduction, the direction of bridge rules matters and hence $\mathfrak{B}_{ij}$ and $\mathfrak{B}_{ji}$ are possibly and expectedly distinct. The bridge graph $\mathrm{G}_{\mathfrak{T}} = \langle V, E\rangle$ of a distributed TBox $\mathfrak{T}$ is defined as follows: $V = I$ and $\langle i, j\rangle \in E$ if $\mathfrak{B}_{ij} \neq \emptyset$. We say that $\mathfrak{T}$ is acyclic if $\mathrm{G}_{\mathfrak{T}}$ is acyclic.

Given a TBox $\mathcal{T}$, a hole is an interpretation $\mathcal{I}^\epsilon = \langle \emptyset, \cdot^\epsilon\rangle$ with empty domain. Holes are used for fighting propagation of inconsistency. We use the most recent definition for holes, introduced in (Serafini et al. 2005). A distributed interpretation $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i\in I, i\neq j}\rangle$ of a distributed TBox $\mathfrak{T}$ consists of a set of local interpretations $\{\mathcal{I}_i\}_{i\in I}$ and a set of domain relations $\{r_{ij}\}_{i\in I, i\neq j}$. .or each $i \in I$, either $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$ is an interpretation of local TBox $\mathcal{T}_i$ or $\mathcal{I}_i = \mathcal{I}^\epsilon$ is a hole. Each domain relation r_{ij} is a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. We denote by $r_{ij}(d)$ the set $\{d' \mid \langle d, d'\rangle \in r_{ij}\}$ and by $r_{ij}(D)$ the set $\bigcup_{d\in D} r_{ij}(d)$.

Definition 2. *For every i and j, a distributed interpretation $\mathfrak{I}$ satisfies the elements of a distributed TBox $\mathfrak{T}$ (denoted by $\mathfrak{I} \models_\epsilon \cdot$) according to the following clauses:*

1. $\mathfrak{I} \models_\epsilon i : C \sqsubseteq D$ *if* $\mathcal{I}_i \models C \sqsubseteq D$.
2. $\mathfrak{I} \models_\epsilon \mathcal{T}_i$ *if* $\mathfrak{I} \models_\epsilon i : C \sqsubseteq D$ *for each* $C \sqsubseteq D \in \mathcal{T}_i$.
3. $\mathfrak{I} \models_\epsilon i : C \xrightarrow{\sqsubseteq} j : G$ *if* $r_{ij}\big(C^{\mathcal{I}_i}\big) \subseteq G^{\mathcal{I}_j}$.
4. $\mathfrak{I} \models_\epsilon i : C \xrightarrow{\sqsupseteq} j : G$ *if* $r_{ij}\big(C^{\mathcal{I}_i}\big) \supseteq G^{\mathcal{I}_j}$.
5. $\mathfrak{I} \models_\epsilon \mathfrak{B}$ *if* $\mathfrak{I}$ *satisfies all bridge rules in* $\mathfrak{B}$.
6. $\mathfrak{I} \models_\epsilon \mathfrak{T}$ *if* $\mathfrak{I} \models_\epsilon \mathfrak{B}$ *and* $\mathfrak{I} \models_\epsilon \mathcal{T}_i$ *for each* i.

If $\mathfrak{I} \models_\epsilon \mathfrak{T}$ then we also say that $\mathfrak{I}$ is a (distributed) model of $\mathfrak{T}$.

The two standard decision problems in DL, satisfiability of concepts and entailment of subsumption, play prominent rôle also in context of DDL. Formally, the decision problems are defined as follows.

Definition 3. *Given a distributed TBox $\mathfrak{T}$, an i-local concept C is satisfiable with respect to $\mathfrak{T}$ if there exists a distributed model $\mathfrak{I}$ of $\mathfrak{T}$ such that $C^{\mathcal{I}_i} \neq \emptyset$.*

Definition 4. *Given a distributed TBox $\mathfrak{T}$ and two i-local concepts C and D, it is said that C is subsumed by D with respect to $\mathfrak{T}$ if $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$ in every distributed model $\mathfrak{I}$ of $\mathfrak{T}$. We also sometimes say that the subsumption formula $i : C \sqsubseteq D$ is entailed by $\mathfrak{T}$ and denote this by $\mathfrak{T} \models_\epsilon i : C \sqsubseteq D$.*

It is a well known result that in most DL subsumption and unsatisfiability are inter-reducible. It follows rather straight forward, that this reduction is also valid for DDL, as follows in Theorem 1.

Theorem 1. *Assume a distributed TBox $\mathfrak{T}$ and two i-local concepts C and D. $\mathfrak{T} \models_\epsilon i : C \sqsubseteq D$ if and only if the concept $\neg C \sqcap D$ is unsatisfiable with respect to $\mathfrak{T}$. Also, C is satisfiable with respect to $\mathfrak{T}$ if and only if the subsumption formula $i : C \sqsubseteq \bot$ is not entailed by $\mathfrak{T}$.*

Below in this paper we present a distributed tableaux reasoning algorithm that decides satisfiability of concepts with respect to a distributed TBox for the adjusted semantics for DDL of (Homola 2008). Thanks to this reduction the algorithm provides a decision procedure for both satisfiability and subsumption entailment.

Among the properties of DDL we find the characterization of subsumption propagation, which formally describes the mechanism of knowledge reuse of DDL. Theorem 2 (Serafini & Tamilin 2004) constitutes the most basic version of this property: thanks to a pair of one into-bridge rule and one onto-bridge rule local subsumption relationship is propagated from the source ontology of these bridge rules to the target ontology.

Theorem 2 (Simple subsumption propagation)**.** *In each distributed TBox $\mathfrak{T}$ with two bridge rules $i : C \xrightarrow{\sqsupseteq} j : G \in \mathfrak{B}$ and $i : D \xrightarrow{\sqsubseteq} j : H \in \mathfrak{B}$ the following holds:*

$$\mathfrak{T} \models_\epsilon i : C \sqsubseteq D \implies \mathfrak{T} \models_\epsilon j : G \sqsubseteq H \ .$$

Study of subsumption propagation in more complex cases appears in the literature (Serafini et al. 2005, Homola 2007). These cases are not as much interesting for this study, since they do not extend the case captured by Theorem 2 in that aspect that only two local ontologies that are directly connected with bridge rules are studied. In this work we concentrate on indirectly connected ontologies. Various other properties of DDL have showed that DDL constitute a monotonic logic, effect of bridge rules is directional, and that if some of the local ontologies is inconsistent, it does not necessarily pollute the whole distributed system. The reader is kindly redirected to (Borgida & Serafini 2003, Serafini & Tamilin 2004, Serafini et al. 2005, Homola 2007) for all details and discussion.

3 Towards Improved Subsumption Propagation in DDL

In our recent paper (Homola 2008) we have adjusted the original semantics of DDL in order to improve subsumption propagation between remote ontologies, in cases when local ontologies are only connected indirectly, as in the example from Fig. 2. The adjustment exploits the *compositional consistency* requirement, that is known from Package-based Description Logics (P-DL) (Bao et al. 2006). In a nutshell, this requirement only allows transitive domain relations in distributed interpretations. Formal definition follows in Definition 5.

Definition 5. *Given a distributed interpretation $\mathfrak{I}$ with domain relation r, we say that r (and also $\mathfrak{I}$) satisfies compositional consistency if for each $i, j, k \in I$ and for each $x \in \Delta^{\mathcal{I}_i}$ with $r_{ij}(x) = D$ we have $r_{jk}(D) = r_{ik}(x)$.*

Now the adjusted semantics is simply obtained from the original DDL semantics by allowing only distributed interpretations that satisfy compositional

consistency. In accordance we often use the wording "DDL under compositional consistency" when referring to this semantics. The adjusted semantics actually extends the original one, in the sense that if some subsumption formula Φ is entailed by a distributed TBox $\mathfrak{T}$ in the original semantics, then it is also entailed by $\mathfrak{T}$ under compositional consistency. The only difference is that in the adjusted semantics possibly some more subsumption formulae are entailed in addition. This is formally stated in the following theorem (see (Homola 2008) for a proof).

Theorem 3. *Given a distributed TBox $\mathfrak{T}$ and a subsumption formula Φ, if $\mathfrak{T} \models_\epsilon \Phi$ according to the original semantics, then $\mathfrak{T} \models_\epsilon \Phi$ also holds in DDL under compositional consistency.*

To demonstrate the mechanism of improved subsumption propagation within the adjusted semantics, let us revisit the example from Fig. 2 formally.

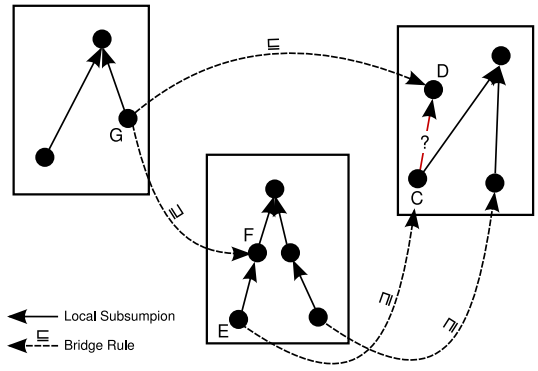

Figure 4: Renaming of concepts from Fig. 2 employed in Example 1. The three local ontologies are referred to as $\mathcal{T}_b$ – behaviour ontology, $\mathcal{T}_c$ – classification ontology, and $\mathcal{T}_y$ – backyard ontology.

Example 1. *Recall the example depicted in Fig. 2. Let's simplify the notation by renaming the concepts as follows: $C :=$ MyCat, $D :=$ DangerousAnimal, $E :=$ Felis, $F :=$ Felidae, $G :=$ Carnivore. Remaining concepts are of no interest (see Fig. 4). Assume the index set $I = \{b, c, y\}$, where b represents the behaviour ontology (on the left), c represents the classification ontology (in the middle) and y represents the backyard ontology (on the right). There are various axioms in $\mathfrak{T} = \langle\{\mathcal{T}_b, \mathcal{T}_c, \mathcal{T}_y, \}, \mathfrak{B}\rangle$ but of the GCIs only $c : E \sqsubseteq F$, actually matters to us, and there are three bridge rules in $\mathfrak{B}$:*

$$b : G \overset{\exists}{\rightarrow} c : F \ , \qquad c : E \overset{\exists}{\rightarrow} y : C \ ,$$
$$b : G \overset{\sqsubseteq}{\rightarrow} y : D \ .$$

We query whether it holds that $\mathfrak{T} \models_\epsilon y : C \sqsubseteq D$. Assume a distributed interpretation $\mathfrak{I}$. Let us first assume that $\mathfrak{I}$ contains no hole. From Definition 2 we get $r_{cy}(E^{\mathcal{I}_c}) \supseteq C^{\mathcal{I}_y}$. We also have $r_{bc}(G^{\mathcal{I}_b}) \supseteq F^{\mathcal{I}_c}$, but since $r_{cy}(\cdot)$ is a mapping we get that $r_{cy}(r_{bc}(G^{\mathcal{I}_b})) \supseteq r_{cy}(F^{\mathcal{I}_c})$. And from compositional consistency we get $r_{cy}(r_{bc}(G^{\mathcal{I}_b})) = r_{by}(G^{\mathcal{I}_b})$ and so $r_{by}(G^{\mathcal{I}_b}) \supseteq r_{cy}(F^{\mathcal{I}_c})$. From the GCI $c : E \sqsubseteq F$ we get $F^{\mathcal{I}_c} \supseteq E^{\mathcal{I}_c}$ and from properties of mapping we again get $r_{cy}(F^{\mathcal{I}_c}) \supseteq r_{cy}(E^{\mathcal{I}_c})$. Putting this all together we derive:

$$r_{by}(G^{\mathcal{I}_b}) \supseteq r_{cy}(F^{\mathcal{I}_c}) \supseteq r_{cy}(E^{\mathcal{I}_c}) \supseteq C^{\mathcal{I}_y} \ ,$$

and that amounts to $r_{by}(G^{\mathcal{I}_b}) \supseteq C^{\mathcal{I}_y}$. On the other hand, from the into-bridge rule between $\mathcal{T}_b$ and $\mathcal{T}_y$ we derive $r_{by}(G^{\mathcal{I}_b}) \subseteq D^{\mathcal{I}_y}$. And so we finally get $C^{\mathcal{I}_y} \subseteq D^{\mathcal{I}_y}$.

For the case with holes assume for instance that $\mathcal{I}_b = \mathcal{I}^\epsilon$. In that case $G^{\mathcal{I}_b} = \emptyset$. Thanks to the constraint generated by bridge rules and the GCI we easily derive that also $F^{\mathcal{I}_c} = \emptyset$, $E^{\mathcal{I}_c} = \emptyset$, and also $C^{\mathcal{I}_y} = \emptyset$. In such a case, however, $C^{\mathcal{I}_y} \subseteq D^{\mathcal{I}_y}$ holds trivially. If we substitute other local interpretations for holes, we get the very same result similarly.

Summing up, in every model of $\mathfrak{T}$ we have $C^{\mathcal{I}_y} \subseteq D^{\mathcal{I}_y}$ and hence $\mathfrak{T} \models_\epsilon y : C \sqsubseteq D$. Recall that we have actually used the compositional consistency requirement in our argumentation. Without it we would not be able to establish the result: we would not be able to prove that $r_{by}(G^{\mathcal{I}_b}) \supseteq r_{cy}(F^{\mathcal{I}_c})$. In fact, the original DDL semantics allows models that violate this inclusion and hence $\mathfrak{T} \models_\epsilon y : C \sqsubseteq D$ does not hold under the original semantics.

Theorem 4 below provides a more general characterization of cases when subsumption propagates to remote ontologies. This characterization generalizes the setting from Figs. 2-4. For a proof, please refer to (Homola 2008).

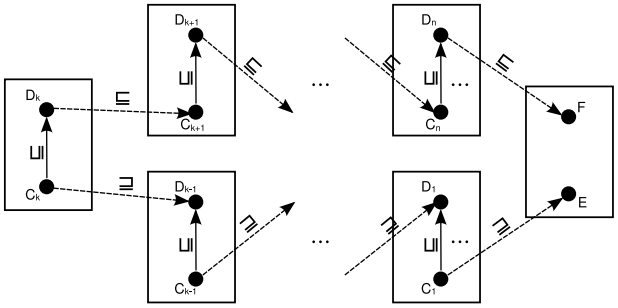

Figure 5: Depiction of the distributed TBox from Theorem 4. Local subsumption is indicated by solid arrows and bridge rules by dashed arrows.

Theorem 4. *Given a distributed TBox, as illustrated in Fig. 5, with index set I and set of bridge rules $\mathfrak{B}$, that features $n+1$ local TBoxes $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n$ with concepts $E, F \in \mathcal{T}_0$, and $C_i, D_i \in \mathcal{T}_i$, for $1 \leq i \leq n$, and k with $1 \leq k \leq n$ such that:*

1. $\mathfrak{T} \models_\epsilon i : C_i \sqsubseteq D_i$, *for* $1 \leq i \leq n$,
2. $i+1 : C_{i+1} \overset{\exists}{\rightarrow} i : D_i \in \mathfrak{B}$, *for* $1 \leq i < k$,
3. $i : D_i \overset{\sqsubseteq}{\rightarrow} i+1 : C_{i+1} \in \mathfrak{B}$, *for* $k \leq i < n$,
4. $1 : C_1 \overset{\exists}{\rightarrow} 0 : E \in \mathfrak{B}$ *and* $n : D_n \overset{\sqsubseteq}{\rightarrow} 0 : F \in \mathfrak{B}$.

In DDL under compositional consistency it follows that $\mathfrak{T} \models_\epsilon 0 : E \sqsubseteq F$.

In a nutshell, Theorem 4 basically says that the effect of bridge rules is now transitive, and hence subsumption propagates even between remote ontologies within the system. Unfortunately, current notation used with DDL does not allow us to formally state this in a more elegant and easier to read fashion.

4 DDL under the Transitivity Condition

Contrary to the claims of (Homola 2008), the adjusted semantics suffers from some drawbacks. The example below demonstrates that the directionality property is violated in this new setting.

Example 2. *Consider a distributed TBox $\mathfrak{T}$ with three local TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ and two bridge rules:*

$$1 : A \overset{\sqsupseteq}{\Rightarrow} 2 : \top \ , \qquad 1 : A \overset{\sqsubseteq}{\Rightarrow} 3 : \bot \ .$$

Assuming that $\mathcal{T}_2$ is consistent we have $\top^{\mathcal{I}_2} \neq \emptyset$ and hence there must be some $y \in \top^{\mathcal{I}_2}$. In such a case the first bridge rule requires that there also is $x \in A^{\mathcal{I}_1}$ with $y \in r_{12}(x)$. However, the compositional consistency condition requires that $r_{32}(r_{13}(x)) = r_{12}(x)$ and hence, there must also be some $z \in \Delta^{\mathcal{I}_3}$ such that $z \in r_{13}(x)$. But for each such z the second bridge rule requires that $z \in \bot^{\mathcal{I}_3}$, which violates the condition that $\bot^{\mathcal{I}_3} = \emptyset$. Hence inconsistency is induced within $\mathcal{T}_2$.

This violates directionality because, as there is no directed path from 2 to 3 in $\mathrm{G}_{\mathfrak{T}}$, there should be no change in the semantics of $\mathcal{T}_3$ if we remove $\mathcal{T}_2$ from $\mathfrak{T}$. But indeed, if we remove $\mathcal{T}_2$, then $\mathcal{T}_3$ is no longer inconsistent.

From the example we conclude that the compositional consistency requirement is perhaps too strong. It implies all the constraints that are necessary in order to increase subsumption propagation in the amount we have described as desired but it also has some more consequences that are possibly destructive as we have seen in Example 2. In the following we study a relaxed version of the compositional consistency condition, which basically is just transitivity of the domain relation in distributed models. We will show that subsumption propagation is more limited under this weaker condition, but on the other hand we will be able to introduce a distributed tableaux reasoning algorithm that decides satisfiability for this semantics. Formally, the weaker condition is defined as follows.

Definition 6. *Given a distributed interpretation $\mathfrak{I}$ with domain relation r, we say that r (and also $\mathfrak{I}$) satisfies the transitivity condition if for each $i, j, k \in I$ and for each $x \in \Delta^{\mathcal{I}_i}$ with $r_{ij}(x) = D$ we have $r_{jk}(D) \subseteq r_{ik}(x)$.*

Apparently, a distributed interpretation satisfies the transitivity condition if and only if its domain relation is transitive. The semantics obtained by allowing only distributed interpretations that satisfy the transitivity condition is hence called DDL under transitivity or DDL with transitive domain relation. The first observation for the new semantics is that the proposition of Theorem 4 does not hold any more if we relax from compositional consistency to transitivity. We demonstrate this problem by an example.

Example 3. *Consider a distributed TBox $\mathfrak{T}$ with three local TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, with local concepts C_1, D_1 in $\mathcal{T}_1$, E_2 in $\mathcal{T}_2$ and F_3 in $\mathcal{T}_3$, and with bridge rules:*

$$2 : E_2 \overset{\sqsubseteq}{\Rightarrow} 3 : F_3 \ , \qquad 3 : F_3 \overset{\sqsubseteq}{\Rightarrow} 1 : D_1 \ ,$$
$$2 : E_2 \overset{\sqsupseteq}{\Rightarrow} 1 : C_1 \ .$$

This distributed TBox is clearly a very simple instance of the setting assumed by Theorem 4. According to this theorem, we should be able to derive $\mathfrak{T} \models_{\mathrm{d}} 1 : C_1 \sqsubseteq D_1$. This is not true, however, as a distributed model of $\mathfrak{T}$ exists in which ${C_1}^{\mathcal{I}_1} \not\subseteq {D_1}^{\mathcal{I}_1}$. This distributed model $\mathfrak{I}$ has three local domains $\Delta^{\mathcal{I}_1} = \{c_1, d_1\}$, $\Delta^{\mathcal{I}_2} = \{e_2\}$, and $\Delta^{\mathcal{I}_3} = \{f_3\}$. Local concepts are interpreted as follows: ${C_1}^{\mathcal{I}_1} = \{c_1, d_1\}$, ${D_1}^{\mathcal{I}_1} = \{d_1\}$, ${E_2}^{\mathcal{I}_2} = \{e_2\}$, and ${F_3}^{\mathcal{I}_3} = \{f_3\}$. Finally, the domain relation is the following: $r_{21} = \{\langle e_2, c_1\rangle, \langle e_2, d_1\rangle\}$, $r_{23} = \{\langle e_2, f_3\rangle\}$, $r_{31} = \{\langle f_3, d_1\rangle\}$, $r_{12} = r_{32} = r_{13} = \emptyset$. It is easily verified that r is transitive, hence the transitivity condition is satisfied, and hence Theorem 4 does not stand if compositional consistency is relaxed to transitivity. At the same time, this example does not invalidate Theorem 4 under compositional consistency, as the stronger condition is not satisfied by $\mathfrak{I}$ because $r_{21}(e_2) = \{c_1, d_1\} \not\subseteq \{d_1\} = r_{31}(r_{23}(e_2))$.

So we see that DDL with transitive domain relation has problems with "chaining" into-bridge rules. Luckily enough the problem does not repeat with chaining onto-bridge rules – if we reinspect Example 1 we will see that transitivity is enough to guarantee the two chaining onto-bridge rules in this example to propagate the subsumption relationship as expected. This observation holds in general an is formally established by Theorem 5 below. This theorem is a weaker version of Theorem 4 and provides characterization of the semantics of DDL under the transitivity condition.

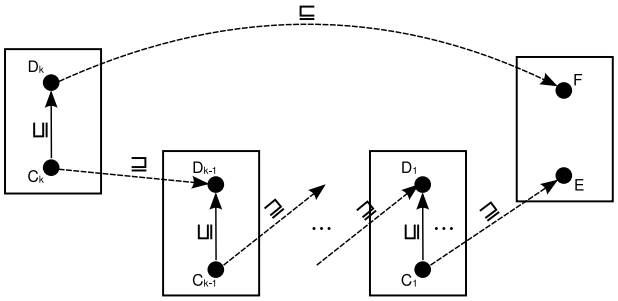

Figure 6: Depiction of the distributed TBox from Theorem 5. Local subsumption is indicated by solid arrows and bridge rules by dashed arrows.

Theorem 5. *Given a distributed TBox, as illustrated in Fig. 6, with index set I and set of bridge rules $\mathfrak{B}$, that features $k+1$ local TBoxes $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_k$ with concepts $E, F \in \mathcal{T}_0$, and $C_i, D_i \in \mathcal{T}_i$, for $1 \leq i \leq k$, such that:*

1. *$\mathfrak{T} \models_\epsilon i : C_i \sqsubseteq D_i$, for $1 \leq i \leq k$,*
2. *$i+1 : C_{i+1} \overset{\sqsupseteq}{\Rightarrow} i : D_i \in \mathfrak{B}$, for $1 \leq i < k$,*
3. *$1 : C_1 \overset{\sqsupseteq}{\Rightarrow} 0 : E \in \mathfrak{B}$ and $k : D_k \overset{\sqsubseteq}{\Rightarrow} 0 : F \in \mathfrak{B}$.*

In DDL under the transitivity condition it follows that $\mathfrak{T} \models_\epsilon 0 : E \sqsubseteq F$.

Proof. The theorem is proved by the following chain of inclusions:

$$E^{\mathcal{I}_0} \overset{1}{\subseteq} r_{k0}\big({C_k}^{\mathcal{I}_k}\big) \overset{2}{\subseteq} r_{k0}\big({D_k}^{\mathcal{I}_k}\big) \overset{3}{\subseteq} F^{\mathcal{I}_0} \ .$$

Inclusion 3 is a direct consequence of the into-bridge rule $k : D_k \overset{\sqsubseteq}{\Rightarrow} 0 : F \in \mathfrak{B}$. As for Inclusion 2, $\mathfrak{T} \models_{\mathrm{d}} k : C_k \sqsubseteq D_k$ implies ${C_k}^{\mathcal{I}_k} \subseteq {D_k}^{\mathcal{I}_k}$ and since $r_{k0}(\cdot)$ we also have $r_{k0}\big({C_k}^{\mathcal{I}_k}\big) \subseteq r_{k0}\big({D_k}^{\mathcal{I}_k}\big)$. Inclusion 1 divides into:

$$E^{\mathcal{I}_0} \subseteq r_{10}\big(r_{21}\big(\cdots r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{k0}\big({C_k}^{\mathcal{I}_k}\big) \ .$$

Both these inclusions ought to be proved by mathematical induction. The first inclusion is proved by induction on k, base case is $E^{\mathcal{I}_0} \subseteq r_{10}\big({C_1}^{\mathcal{I}_1}\big)$. This is a direct consequence of the onto-bridge rule $1 : C_1 \overset{\sqsupseteq}{\Rightarrow} 0 : E \in \mathfrak{B}$. In the induction step,

by the induction hypothesis we have that $E^{\mathcal{I}_0} \subseteq r_{10}\big(r_{21}\big(\cdots r_{k-1k-2}\big(C_{k-1}{}^{\mathcal{I}_{k-1}}\big)\cdots\big)\big)$. From the assumptions of the theorem we derive:

$$C_{k-1}{}^{\mathcal{I}_{k-1}} \subseteq D_{k-1}{}^{\mathcal{I}_{k-1}} \subseteq r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big) \quad .$$

The composition $r_{10}(r_{21}(\cdots r_{k-1k-2}(\cdot)\cdots))$ is still a mapping, and so $r_{10}\big(r_{21}\big(\cdots r_{k-1k-2}\big(C_{k-1}{}^{\mathcal{I}_{k-1}}\big)\cdots\big)\big)$ is a subset of $r_{10}\big(r_{21}\big(\cdots r_{k-1k-2}\big(r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big)\big)\cdots\big)\big)$. Together with the induction hypothesis this amounts to the inclusion we wanted to prove.

As for the second inclusion, we shall prove a slightly more general proposition:

$$r_{n+1n}\big(r_{n+2n+1}\big(\cdots r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{kn}\big(C_k{}^{\mathcal{I}_k}\big) \quad ,$$

where $0 \leq n \leq k-2$. The inclusion we ought to prove is then derived by setting $n = 0$. The proposition is proved by mathematical induction on $k - n$. As for the base case consider $k - n = 2$, and so $n = k - 2$. That means we have to show $r_{k-1k-2}\big(r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big)\big) \subseteq r_{kk-2}\big(C_k{}^{\mathcal{I}_k}\big)$, which indeed holds because $\mathfrak{I}$ satisfies the transitivity condition. The induction step is for $k - n = j > 2$, and so $n = k - j$. From the induction hypothesis we get $r_{n+2n+1}\big(r_{n+3n+2}\big(\cdots r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{kn+1}\big(C_k{}^{\mathcal{I}_k}\big)$. And since $r_{n+1n}(\cdot)$ is a mapping, we also get $r_{n+1n}\big(r_{n+2n+1}\big(\cdots r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{n+1n}\big(r_{kn+1}\big(C_k{}^{\mathcal{I}_k}\big)\big)$. Since $\mathfrak{I}$ satisfies the transitivity condition, we get $r_{n+1n}\big(r_{kn+1}\big(C_k{}^{\mathcal{I}_k}\big)\big) \subseteq r_{kn}\big(C_k{}^{\mathcal{I}_k}\big)$. By combining the last two inclusions we directly get:

$$r_{n+1n}\big(r_{n+2n+1}\big(\cdots r_{kk-1}\big(C_k{}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{kn}\big(C_k{}^{\mathcal{I}_k}\big) \quad .$$

By setting $n = 0$ we derive the inclusion we wanted to prove, and hence the Theorem. □

And thus we are able to conclude this section by comparing the three semantics of DDL with respect to the amount of subsumption propagation they allow along chaining bridge rules. The original semantics of DDL in general does not guarantee subsumption propagation between remote ontologies along chaining bridge rules. A limited case when subsumption propagates between remote ontologies under the original semantics is described in (Homola 2008). Semantics of DDL with transitive domain relation ensures subsumption propagation along chaining onto-bridge rules in scenarios that instantiate Theorem 5. Finally, in DDL under computational consistency subsumption propagates along chaining onto-bridge rules and along chaining into-bridge rules in addition, as established by Theorem 4.

5 Distributed Tableaux Algorithm for DDL with Transitive Domain Relation

In this section, we introduce a distributed tableaux algorithm for deciding satisfiability of concepts with respect to an acyclic distributed TBox for DDL over $\mathcal{ALC}$ under the transitivity requirement. As in the algorithm of (Serafini et al. 2005), also in our approach, local reasoners are run independently. We keep precise track of the domain relation r however, and the communication between local reasoners is divided into multiple queries. Hence while the original algorithm can be seen as working with autonomous local tableaux by passing queries, our algorithm can be seen as working with a truly distributed tableau.

Definition 7. *Assume a distributed TBox $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \mathfrak{B}\rangle$ over $\mathcal{ALC}$ with index set I, concept names $\mathrm{N_C} = \bigcup_{i\in I}\mathrm{N_{C}}i$ and role names $\mathrm{N_R} = \bigcup_{i\in I}\mathrm{N_{R}}i$. Let CC_i be the set of all (atomic and complex) concepts over $\mathrm{N_{C}}i$ and $\mathrm{N_{R}}i$ in negation normal form. A distributed completion tree $T = \{T_i\}_{i\in I}$ is a set of labeled trees $T_i = \langle V_i, E_i, \mathcal{L}_i, r_i\rangle$, such that for each $i \in I$:*

1. *the members of $\{V_i\}_{i\in I}$ are mutually disjoint;*
2. *the members of $\{E_i\}_{i\in I}$ are mutually disjoint;*
3. *the labeling function $\mathcal{L}_i$ labels each node $x \in V_i$ with $\mathcal{L}(x) \subseteq 2^{\mathrm{CC}_i}$ and each edge $\langle x, y\rangle \in E_i$ with $\mathcal{L}(\langle x, y\rangle) \in \mathrm{N_{R}}i$;*
4. *the labeling function r_i labels each node $x \in V_i$ with a set of references to its r-images $r_i(x) \subseteq \{j : y \mid j \in I \wedge y \in V_j\}$.*

During the run of the tableaux algorithm, tableaux expansion rules are applied on the completion tree and the tree is expanded by each rule application. If no more rules are applicable any more, we say that the completion tree is *complete*. There is a *clash* in the completion tree T if for some $x \in V_i$ and for some $C \in \mathrm{N_{C}}i$ we have $\{C, \neg C\} \subseteq \mathcal{L}_i(x)$. If there is no clash in T then we say that T is *clash-free*. In order to assure termination we use standard subset blocking that is common for $\mathcal{ALC}$. Given a distributed completion tree $T = \{T_i\}_{i\in I}$, a node $x \in V_i$ is blocked, if it has an ancestor $y \in V_i$ such that $\mathcal{L}_i(x) \subseteq \mathcal{L}_i(y)$. In such a case we also say that x is blocked by y. Node $y \in V_i$ is said to be an *R-successor* of $x \in V_i$, if $\langle x, y\rangle \in E_i$ and $\mathcal{L}_i(\langle x, y\rangle) = R$.

The distributed tableaux algorithm for deciding satisfiability of concepts with respect to a distributed TBox takes a distributed TBox $\mathfrak{T}$, a concept C in NNF and $i \in I$ as its inputs. The algorithm then continues in three steps:

1. *Initialization.* Create new completion tree $T = \{T_j\}_{j\in I}$ such that $T_j = \langle\{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \emptyset\rangle$ for $j = i$ and $T_j = \langle\emptyset, \emptyset, \emptyset, \emptyset\rangle$ for $j \neq i$.
2. *Tableau expansion.* Apply the tableaux expansion rules of Fig. 7 exhaustingly.
3. *Answer.* If none of the tableaux expansion rules in Fig. 7 is applicable any more (i.e., the completion tree is now complete), answer "C is satisfiable" if a clash-free completion tree has been constructed. Answer "C is unsatisfiable" otherwise.

Below we present a formal correctness proof for the newly introduced algorithm. The proof is based on the classic proof for $\mathcal{ALC}$ as in fact the only new thing to prove here is that the algorithm uses the $\sqsupseteq$-rule and the $\sqsubseteq$-rule to combine several autonomous local $\mathcal{ALC}$ reasoners correctly. The proof is done in three parts. We first prove termination: on every input the algorithm always terminates and never ends up in an infinite loop; then soundness: if the algorithm answers that C is satisfiable with respect to $\mathfrak{T}$ for some i-local concept C and a distributed TBox $\mathfrak{T}$ then there actually exists some model of $\mathfrak{T}$ that supports this; and finally we prove completeness: for every concept C that is satisfiable with respect to $\mathfrak{T}$ the algorithm indeed gives a correct answer.

Theorem 6. *Given a distributed TBox $\mathfrak{T}$ over $\mathcal{ALC}$ with acyclic bridge graph $\mathrm{G}_{\mathfrak{T}}$ and an i-local concept C on the input, the distributed tableaux algorithm for deciding satisfiability of concepts with respect to a distributed TBox over $\mathcal{ALC}$ for DDL with transitive domain relation always terminates and it is sound and complete.*

$\sqcap$-rule:
If $C_1 \sqcap C_2 \in \mathcal{L}_i(x)$ for some $x \in V_i$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}_i(x)$, and x is not blocked,
then set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{C_1, C_2\}$.

$\sqcup$-rule:
If $C_1 \sqcup C_2 \in \mathcal{L}_i(x)$ for some $x \in V_i$ and $\{C_1, C_2\} \cap \mathcal{L}_i(X) = \emptyset$, and x is not blocked,
then either set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{C_1\}$ or set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{C_2\}$.

$\forall$-rule:
If $\forall R.C \in \mathcal{L}_i(x)$ for some $x \in V_i$, and there is R-successor y of x s.t. $C \notin \mathcal{L}_i(y)$, and x is not blocked,
then set $\mathcal{L}_i(y) = \mathcal{L}_i(y) \cup \{C\}$.

$\exists$-rule:
If $\exists R.C \in \mathcal{L}_i(x)$, for some $x \in V_i$ with no R-successor y s.t. $C \in \mathcal{L}_i(y)$, and x is not blocked,
then add new node z to V_i, add the edge $\langle x, z\rangle$ to E_i, and set $\mathcal{L}_i(z) = \{C\}$ and $\mathcal{L}_i(\langle x, z\rangle) = \{R\}$.

$\mathcal{T}$-rule:
If $C \sqsubseteq D \in \mathcal{T}$ and for some $x \in V_i$ $\mathrm{nnf}(\neg C \sqcup D) \notin \mathcal{L}_i(x)$, and x is not blocked,
then set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{\mathrm{nnf}(\neg C \sqcup D)\}$.

$\stackrel{\exists}{\Rightarrow}$-rule:
If $G \in \mathcal{L}_j(x)$ for some $x \in V_j$, $i : C \stackrel{\exists}{\Rightarrow} j : G \in \mathfrak{B}$, and there is no $y \in V_i$ s.t. $C \in \mathcal{L}_i(y)$ and $j : x \in r_i(y)$, and x is not blocked,
then add new node y to V_i and set $\mathcal{L}_i(y) = \{C\}$, and set $r_i(y) = \{j : x\} \cup r_j(x)$.

$\stackrel{\sqsubseteq}{\Rightarrow}$-rule:
If $D \in \mathcal{L}_i(x)$ for some $x \in V_i$, $i : D \stackrel{\sqsubseteq}{\Rightarrow} j : H \in \mathfrak{B}$ and there is $y \in V_j$ s.t. $j : y \in r_i(x)$ and $H \notin \mathcal{L}_j(y)$
then set $\mathcal{L}_j(y) = \mathcal{L}_j(y) \cup \{H\}$.

Figure 7: Tableaux expansion rules for DDL over $\mathcal{ALC}$ under the transitivity requirement. First five rules are standard $\mathcal{ALC}$ tableaux rules. Note that the $\sqcup$-rule is non-deterministic. The final two rules are new and are triggered by bridge rules.

Proof. Termination. Given a distributed TBox $\mathfrak{T}$ with local TBox $\mathcal{T}_i$ and an i-local concept C, we ought to prove that the algorithm, once started with $\mathfrak{T}$, C and i on input, eventually terminates. The algorithm initializes the completion tree to $T = \{T_j\}_{j \in I}$ such that $T_j = \langle \{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \emptyset\rangle$ for $j = i$ and $T_j = \langle \emptyset, \emptyset, \emptyset, \emptyset\rangle$ for $j \neq i$. The computation then continues by expanding $\mathcal{T}_i$. If there are no onto-bridge rules ingoing into $\mathcal{T}_i$ the algorithm eventually terminates thanks to subset blocking, this result is known for $\mathcal{ALC}$. If there are some ingoing onto-bridge rules then possibly in some $x \in V_i$ such that $C \in \mathcal{L}_i(x)$ and C appears on a right hand side of an onto-bridge rule, say $k : D \stackrel{\exists}{\Rightarrow} i : C$, then the $\stackrel{\exists}{\Rightarrow}$-rule is applied and computation is triggered in T_k. By structural subsumption we assume that this computation eventually terminates (the length of the longest incoming path of into-bridge rules decreased for $\mathcal{T}_k$ compared to $\mathcal{T}_i$, and all such paths are finite because of acyclicity). During this process we possibly get some new concepts in $\mathcal{L}_i(x)$ that are introduced thanks to incoming into-bridge rules that trigger the $\stackrel{\sqsubseteq}{\Rightarrow}$-rule – but only finitely many. The computation now continues in x and its descendants and possibly the $\stackrel{\exists}{\Rightarrow}$-rule is triggered again in some $y \in V_i$, a descendant of x. But thanks to subset blocking, this happens only finitely many times. Hence the algorithm eventually terminates.

Soundness. Now that we know that the algorithm terminates, we shall prove that if it answers "C is satisfiable" on input $\mathfrak{T}$, C and i, then it also holds that C is satisfiable in $\mathcal{T}_i$ with respect to $\mathfrak{T}$, that is we must show that in such a case there exists a distributed model $\mathfrak{I}$ of $\mathfrak{T}$ such that $C^{\mathcal{I}_i} \neq \emptyset$. Given $\mathfrak{T}$, C and i and let T be the complete and clash-free completion tree that the algorithm has constructed to support the decision. Let us construct the distributed interpretation $\mathfrak{I}$ as follows:

1. Let $\Delta^{\mathcal{I}_i} = V_i$, for each $i \in I$.
2. Let $x \in A^{\mathcal{I}_i}$, for each atomic concept $A \in \mathcal{L}_i(x)$, for each $x \in V_i$ and each $i \in I$.
3. Let $\langle x, y\rangle \in R^{\mathcal{I}_i}$ for each $\langle x, y\rangle \in E_i$ such that $\mathcal{L}_i(\langle x, y\rangle) = R$, for each $i \in I$, if y is not blocked.
4. Let $\langle x, z\rangle \in R^{\mathcal{I}_i}$ for each $\langle x, y\rangle \in E_i$ such that $\mathcal{L}_i(\langle x, y\rangle) = R$, for each $i \in I$, in case that y is blocked by z.
5. Let $r_{ij}(x) = y$ for each $x \in V_i$ and for each $j : y \in r_i(x)$.

Please observe that if computation was never triggered within some T_j of T during the run of the algorithm, then $\mathcal{I}_j = \mathcal{I}^\epsilon$. It remains to show that $\mathfrak{I}$ is in fact a model of $\mathfrak{T}$ and $C^{\mathcal{I}_i} \neq \emptyset$. We will first prove the following proposition:

> Given any $i \in I$, for each $E \in \mathcal{L}_i(x)$ (i.e., also for complex concepts) we have $x \in E^{\mathcal{I}_i}$.

This is proved by induction on the structure of E. We need to consider the following cases:

1. E is atomic. We know that $x \in E^{\mathcal{I}_i}$ from the construction.
2. $E = \neg E_1$, E_1 atomic. Since T is clash-free, $E_1 \notin \mathcal{L}_i(x)$ and by construction $x \notin {E_1}^{\mathcal{I}_i}$. In that case however $x \in E^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \setminus {E_1}^{\mathcal{I}_i}$.
3. $E = E_1 \sqcap E_2$. Since T is complete, the $\sqcap$-rule is not applicable and hence also $E_1 \in \mathcal{L}_i(x)$ and $E_2 \in \mathcal{L}_i(x)$. By induction we now have that $x \in {E_1}^{\mathcal{I}_i}$ and $x \in {E_2}^{\mathcal{I}_i}$ and hence also $x \in E^{\mathcal{I}_i}$.
4. $E = E_1 \sqcup E_2$. Since T is complete, the $\sqcup$-rule is not applicable and hence either $E_1 \in \mathcal{L}_i(x)$ or $E_2 \in \mathcal{L}_i(x)$. By induction we now either have $x \in {E_1}^{\mathcal{I}_i}$ or we have $x \in {E_2}^{\mathcal{I}_i}$. In either case however also $x \in E^{\mathcal{I}_i}$.
5. $E = \exists R.E_1$. Since T is complete, the $\exists$-rule is not applicable and hence there must be $y \in V_i$, an R-successor of x such that $E_1 \in \mathcal{L}_i(y)$. By induction $y \in {E_1}^{\mathcal{I}_i}$ and by construction of $\mathfrak{I}$ $\langle x, y\rangle \in R^{\mathcal{I}_i}$. But that means that $x \in E^{\mathcal{I}_i}$.
6. $E = \forall R.E_1$. Since T is complete, the $\forall$-rule is not applicable and hence for every $y \in V_i$, an R-successor of x, we have $E_1 \in \mathcal{L}_i(y)$. By induction $y \in {E_1}^{\mathcal{I}_i}$ and by construction of $\mathfrak{I}$ $\langle x, y\rangle \in R^{\mathcal{I}_i}$. But that means that $x \in E^{\mathcal{I}_i}$.

Thus we have verified that the local interpretation $\mathcal{I}_i$ is indeed an $\mathcal{ALC}$ interpretation and that C has an instance in this interpretation (since $C \in \mathcal{L}_i(s_0)$). In addition we have in fact also proved that each i-local GCI axiom $E_1 \sqsubseteq E_2$ is satisfied by $\mathcal{I}_i$ – from the completeness of T, $\mathrm{nnf}(\neg E_1 \sqcup E_2) \in x$, for each $x \in V_i$, and hence $x \in (\mathrm{nnf}(\neg E_1))^{\mathcal{I}_i} \cup {E_2}^{\mathcal{I}_i}$. It follows that $x \in {E_1}^{\mathcal{I}_i}$ implies $x \in {E_2}^{\mathcal{I}_i}$.

It remains to show that $\mathfrak{I}$ is indeed a distributed model of $\mathfrak{T}$ in the adjusted semantics. First, all the bridge rules must be satisfied. Given an onto-bridge rule $k : E_1 \stackrel{\exists}{\Rightarrow} l : E_2 \in \mathfrak{B}$, we ought to show that $r_{kl}\big({E_1}^{\mathcal{I}_k}\big) \supseteq {E_2}^{\mathcal{I}_l}$. So let $x \in {E_2}^{\mathcal{I}_l}$, that is $x \in V_l$ and $E_2 \in \mathcal{L}_l(x)$. But T is complete, $\stackrel{\exists}{\Rightarrow}$-rule is not applicable, and hence there must be $y \in V_k$ with $E_1 \in \mathcal{L}_k(y)$ and $l : x \in r_k(y)$. That means however that $y \in {E_1}^{\mathcal{I}_k}$ and $\langle x, y\rangle \in r_{kl}$, and so $x \in r_{kl}\big({E_1}^{\mathcal{I}_k}\big)$. Therefore the bridge rule is satisfied by $\mathfrak{I}$.

Given an into-bridge rule $k : E_1 \stackrel{\sqsubseteq}{\Rightarrow} l : E_2 \in \mathfrak{B}$, we ought to show that $r_{kl}(E_1^{\mathcal{I}_k}) \subseteq E_2^{\mathcal{I}_l}$. Let $x \in r_{kl}(E_1^{\mathcal{I}_k})$. Then there is $y \in V_k$ such that $l : x \in r_l(y)$. But since T is complete, it must be the case that $E_2 \in \mathcal{L}_l(x)$ and hence $x \in E_2^{\mathcal{I}_l}$. Therefore the bridge rule is satisfied by $\mathfrak{I}$.

The last thing in order to verify that $\mathfrak{I}$ is indeed a model of $\mathfrak{T}$ is to show that $\mathfrak{I}$ satisfies the transitivity requirement. We ought to show that for each $k, l, m \in I$, if $y \in r_{kl}(x)$ and $z \in r_{lm}(y)$ then also $z \in r_{km}(x)$. Given the two assumptions we know by construction that $l : y \in r_k(x)$ and $m : z \in r_l(y)$. That means that $x \in V_k$ was initialized after several "chained" applications of the $\stackrel{\sqsupseteq}{\Rightarrow}$-rule starting from $y \in V_l$ and y in turn after several "chained" $\stackrel{\sqsupseteq}{\Rightarrow}$-rule applications starting from $z \in V_m$. In that case however $r_l(y) \subseteq r_k(x)$. Hence $m : z \in r_k(x)$ and so $z \in r_{km}(x)$.

And thus $\mathfrak{I}$ is a distributed model of $\mathfrak{T}$ that satisfies the transparency condition and $C^{\mathcal{I}_i} \neq \emptyset$ – in other words, C is satisfiable in $\mathcal{T}_i$ with respect to $\mathfrak{T}$.

Completeness. Given $\mathfrak{T}$ with index set I, a concept C and $i \in I$ such that C is satisfiable in $\mathcal{T}_i$ with respect to $\mathfrak{T}$, we shall prove that the algorithm answers "C is satisfiable", if run on input $\mathfrak{T}$, C and i. Let $\mathfrak{I}$ be a distributed model of $\mathfrak{T}$ with $C^{\mathcal{I}_i} \neq \emptyset$, we know that one must exist. We will simulate the run of the algorithm. During the initialization T_i is set to $\langle \{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \emptyset \rangle$ and all the other T_j, $i \neq j$, are set to $\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$. There is no clash in T. We will show by induction (on the number of tableau expansion steps) that after each tableau expansion step the completion tree T is expanded in such a way that no clash is introduced. In order to demonstrate this, we inductively construct an auxiliary mapping $\pi : \bigcup_{i \in I} V_i \to \bigcup_{i \in I} \Delta^{\mathcal{I}_i}$ to track the relation between $\mathfrak{I}$ and T. This mapping will keep the property $(*)$:

> For each node $x \in V_i$, for each $i \in I$: if $C \in \mathcal{L}_i(x)$ then $\pi(x) \in C^{\mathcal{I}_i}$.

Let x be arbitrary member of $C^{\mathcal{I}_i}$, place $\pi(s_0) = x$. So, if a tableaux expansion rule is triggered in some $x \in V_i$ of the clash-free completion tree T (from induction hypothesis), we consider several cases, based on the kind of rule that was triggered:

1. $\sqcap$-rule is triggered in x because $E_1 \sqcap E_2 \in \mathcal{L}_i(x)$. Then by induction hypothesis $\pi(x) \in E_1 \sqcap E_2^{\mathcal{I}_i}$. In that case also $\pi(x) \in E_1^{\mathcal{I}_i}$ and $\pi(x) \in E_2^{\mathcal{I}_i}$, and hence the property $(*)$ is maintained after the algorithm adds E_1 and E_2 to $\mathcal{L}_i(x)$.

2. $\sqcup$-rule is triggered in x because $E_1 \sqcup E_2 \in \mathcal{L}_i(x)$. Then by induction hypothesis $\pi(x) \in E_1 \sqcup E_2^{\mathcal{I}_i}$. In that case however either $\pi(x) \in E_1^{\mathcal{I}_i}$ or $\pi(x) \in E_2^{\mathcal{I}_i}$. Without loss of generality let it be the case that $\pi(x) \in E_1^{\mathcal{I}_i}$. Without loss of generality we assume that the algorithm adds E_1 to $\mathcal{L}_i(x)$, as non-deterministic decision takes place. Hence the property $(*)$ is maintained after this step.

3. $\exists$-rule is triggered in x because it has an R-successor y, and $\exists R.E_1 \in \mathcal{L}_i(x)$. Then $\pi(x) \in \exists R.E_1^{\mathcal{I}_i}$ and so there is some $y' \in \Delta^{\mathcal{I}_i}$ such that $y' \in E_1^{\mathcal{I}_i}$ and $\langle \pi(x), y' \rangle \in R^{\mathcal{I}_i}$. When the algorithm generates new node y in this step we set $\pi(y) = y'$ and the property $(*)$ is maintained.

4. $\forall$-rule is triggered in x because it has an R-successor y, and $\forall R.E_1 \in \mathcal{L}_i(x)$, as a consequence E_1 is added to $\mathcal{L}_i(y)$. We know that y was created by an application of the $\exists$-rule and hence $\pi(y)$ is an R-successor of x in $\mathcal{I}_i$. But $\mathcal{I}_i$ is a model of $\mathcal{T}_i$ and hence $\pi(y) \in y^{\mathcal{I}_i}$ since from induction hypothesis we know that $\pi(x) \in \forall R.E_1^{\mathcal{I}_i}$. Hence $(*)$ is maintained after this step.

5. $\mathcal{T}$-rule is triggered in x, resulting to adding $\mathrm{nnf}(\neg E_1 \sqcup E_2)$ into $\mathcal{L}_i(x)$. Then $(*)$ is maintained as $\mathcal{I}_i$ is a model of $\mathcal{T}_i$ and so it holds that $\pi(x) \in (\mathrm{nnf}(\neg E_1 \sqcup E_2))^{\mathcal{I}_i}$.

6. $\stackrel{\sqsupseteq}{\Rightarrow}$-rule is triggered in x because $E_2 \in \mathcal{L}_i(x)$ and $j : E_1 \stackrel{\sqsupseteq}{\Rightarrow} i : E_2 \in \mathfrak{B}$. From induction hypothesis, $\pi(x) \in E_2^{\mathcal{I}_i}$, and hence $\pi(x) \in r_{ji}(y')$ for some $y' \in E_1^{\mathcal{I}_j}$. Set $\pi(y) = y'$ for the node y that is newly created in V_i as the result of this expansion step. The label $\mathcal{L}_j(y)$ has been set to E_1 but since $y' \in E_1^{\mathcal{I}_j}$ then $(*)$ is maintained.

7. $\stackrel{\sqsubseteq}{\Rightarrow}$-rule is triggered in x because of $E_1 \in \mathcal{L}_i(x)$, $j : y \in r_i(x)$ and $i : E_1 \stackrel{\sqsubseteq}{\Rightarrow} j : E_2 \in \mathfrak{B}$, resulting to adding E_2 into $\mathcal{L}_j(y)$. From induction hypothesis have $\pi(x) \in E_1^{\mathcal{I}_i}$. Observe that the definition of $\stackrel{\sqsupseteq}{\Rightarrow}$-rule and the inductive construction of $\pi(\cdot)$ in previous steps also assure that $j : y \in r_i(x)$ implies $\pi(y) \in r_{ij}(\pi(x))$. This is because initialization of new $y \in V_j$ always follows incoming r-edge and r is transitive because of computational consistency. It follows that $\pi(y) \in E_2^{\mathcal{I}_j}$, since $\mathfrak{I}$ is a distributed model of $\mathfrak{T}$ and the bridge rule assures this. Hence $(*)$ is maintained even after this step.

We already know that the algorithm always terminates. Once this happens, it follows that T is now complete, because no rule is applicable, and clash-free, because the property $(*)$ is maintained all the way up to this point. Hence the algorithm answers "C is satisfiable" and hence the theorem. □

The algorithm for the original DDL semantics of (Serafini et al. 2005) is obviously truly distributed in that sense that it supports a scenario in which several autonomous reasoning services run independently, one for each local ontology, and communicate by passing queries. While this is also the case for the newly introduced algorithm, it is not necessarily that obvious. In order ti clarify this, we provide a message protocol that handles $\stackrel{\sqsupseteq}{\Rightarrow}$-rule and $\stackrel{\sqsubseteq}{\Rightarrow}$-rule execution in a truly distributed fashion, and it also collects the information that the completion tree is complete within the reasoner that has initialized the computation. The algorithm employs three kinds of messages $\mathsf{querySat}(x, r, C)$, $\mathsf{answerSat}(x, C, \mathit{answer})$, and $\mathsf{pushConcept}(x, C)$. In the message based version, the algorithm starts by initializing the local completion tree T_j when asked for satisfiability of some j-local concept C with respect to $\mathfrak{T}$. More local completion trees are initialized during the runtime by passing a $\mathsf{querySat}(x, r, C)$ message every time the $\stackrel{\sqsupseteq}{\Rightarrow}$-rule is fired. When $\stackrel{\sqsubseteq}{\Rightarrow}$-rule is fired, the consequences of the into-bridge rule are propagated by passing a $\mathsf{pushConcept}(x, C)$ message. Once a local completion tree T_k is complete this is announced by passing a $\mathsf{answerSat}(x, C, \mathit{answer})$ message to the local reasoner that has triggered the computation in $\mathcal{T}_k$. Detailed specification of the protocol messages is given in Fig. 8.

6 Related Work

A distributed tableaux reasoning algorithm for the original semantics of DDL has been introduced in (Serafini & Tamilin 2005, Serafini et al. 2005) and implemented in system DRAGO. This algorithm is

querySat(x, r, C):
Send: if $G \in \mathcal{L}_j(x)$ for some $x \in V_j$, $i : C \stackrel{\sqsupseteq}{\to} j : G \in \mathfrak{B}$, and x is not blocked, pass the message querySat$(x, r_j(x), C)$ to T_i.
Receive: upon receipt of a message querySat(x, r, C) from T_i, create new completion tree $\langle \{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \{s_0 \mapsto \{i : x\} \cup r\}\rangle$ within T_j and start the computation. Record the sent message in S_j.

answerSat$(x, C, answer)$:
Send: if the local completion tree for concept C within T_j, that has been initialized due the message querySat(x, r, C) previously received from T_i, is now complete, and there are no outgoing messages recorded in S_j, send the message answerSat$(x, C, answer)$ to $\mathcal{T}_i$ with $answer$ set to true if T_j is clash-free and to false otherwise.
Receive: upon receipt of a message answerSat$(x, C, answer)$ from $\mathcal{T}_i$, remove the message querySat(x, r, C) from S_j, that has been previously send to T_i. If $answer$ is false then add $E \sqcup \neg E$ to $\mathcal{L}_j(x)$ for some new concept name E.

pushConcept(x, C):
Send: if D has been added to $\mathcal{L}_j(x)$ in the previous step, for some $x \in V_j$, then for each $j : D \stackrel{\sqsubseteq}{\to} i : H \in \mathfrak{B}$ s.t. $i : y \in r_j(x)$ send the message pushConcept(y, H) to T_i.
Receive: upon receipt of a message pushConcept(x, C) from $\mathcal{T}_i$, add C to $\mathcal{L}_j(x)$.

Figure 8: The message protocol for the newly introduced distributed tableaux algorithm. For each kind of message we specify when the message is sent from some local reasoning service T_j and also what happens when the message is received in some local reasoning service T_j (T_j is always the local reasoning service). An auxiliary data structure S_j is introduced in each T_j to track messages that have been sent in order to assure termination.

based on a fix-point characterization of the original DDL semantics. Given a set of bridge rules $\mathfrak{B}_{ij}$ between $\mathcal{T}_i$ and $\mathcal{T}_j$, define the operator $\mathfrak{B}_{ij}(\cdot)$ as follows: $\mathfrak{B}_{ij}(\mathcal{T}_i) = \{G \sqsubseteq \bigsqcup_{k=1}^{n} H_k \mid \mathcal{T}_i \models A \sqsubseteq \bigsqcup_{k=1}^{n} B_k, i : A \stackrel{\sqsupseteq}{\to} j : G \in \mathfrak{B}_{ij}, i : B_k \stackrel{\sqsubseteq}{\to} j : H_k \in \mathfrak{B}_{ij}, 1 \leq k \leq n\}$. Then the $\mathfrak{B}$ operator is defined: $\mathfrak{B}(\{\mathcal{T}_i\}_{i\in I}) = \{\mathcal{T}_i \cup \bigcup_{j\neq i} \mathfrak{B}_{ji}(\mathcal{T}_j)\}$. As given in (Serafini et al. 2005), the $\mathfrak{B}$ operator always has a fix-point $\mathfrak{B}^*(\mathfrak{T})$ when repeatedly applied on a distributed TBox $\mathfrak{T}$. Moreover, $\mathfrak{T} \models_\epsilon i : \phi$ if and only if the i-th component TBox of $\mathfrak{B}^*(\mathfrak{T})$ locally entails ϕ. Consequently, the standard $\mathcal{SHIQ}$ tableaux reasoning algorithm (Horrocks et al. 1999) is used with one additional bridge rule (see Fig. 9). The unsatisfiability check called by $\mathfrak{B}_{ij}$-rule is done by running the same algorithm again for the i-concept that is being checked. The algorithm assumes distributed TBoxes with acyclic bridge graph to insure termination. Compared to the algorithm introduced in this paper, this algorithm does not keep track of the domain relation r and does not in fact construct a true distributed tableau that would correspond to a particular distributed model of the input concept. Instead it cleverly uses the fix-point characterization and constructs multiple local tableaux in order to guarantee the existence of such a model. Most prominently, all consequences that are added to the triggering node if the unsatisfiability check is successful are estimated prior to the unsatisfiability check is executed. In contrast, in our approach computation in remote ontology is triggered by one message and consequences are announced back to the triggering node once they are computed, possibly by several independent messages. We believe that keeping precise track of all model structures, including the domain relation and dividing the communication into more fine-grained messages may provide better grounds for future optimization. Finally, it is worth noting that the newly introduced algorithm is easily adjusted to correspond to the original DDL semantics (by setting $r_i(y)$ to $\{j : x\}$ and not to $\{j : x\} \cup r_j(x)$ in the $\stackrel{\sqsupseteq}{\to}$-rule).

$\mathfrak{B}_{ij}$**-rule:**
If $G \in \mathcal{L}_j(x)$, $i : A \stackrel{\sqsupseteq}{\to} j : H \in \mathfrak{B}_{ij}$,
$BH \subseteq \{\langle B_k, H_k\rangle \mid i : B_k \stackrel{\sqsubseteq}{\to} j : H_k \in \mathfrak{B}_{ij}\}$,
$B = \{B \mid \langle B, X\rangle \in BH\}$, $H = \{H \mid \langle Y, H\rangle \in BH\}$, $H \notin \mathcal{L}_j(x)$
and $A \sqcap \neg \bigsqcup B$ is unsatisfiable w.r.t $\mathfrak{T}$ in $\mathcal{T}_i$
then set $\mathcal{L}_j(x) = \mathcal{L}_j(x) \cup \{\bigsqcup H\}$.

Figure 9: The tableaux expansion rule used in the original DDL algorithm (Serafini & Tamilin 2005).

Another distributed ontology framework is $\mathcal{E}$-connections (Cuenca Grau et al. 2004). Here, inter-ontology roles (called links) are employed instead of concept mapping. A dedicated set ϵ_{ij} of symbols is used for (directed) links between $\mathcal{T}_i$ and $\mathcal{T}_j$. Links are then used in existential and value restrictions, and complex concepts involving links are formed (by instance the i-concept $\exists E.C$ is composed using the link $E \in \epsilon_{ij}$ and the j-concept C). Semantically, a combined interpretation consists of local interpretations $\mathcal{I}_i$ with non-empty domains $\Delta^{\mathcal{I}_i}$ and each link $E \in \epsilon_{ij}$ is interpreted by $E^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. Reasoning support for $\mathcal{E}$-connections is provided by extending the tableaux reasoning algorithm for $\mathcal{SHIF}(D)$. The two tableaux rules that are essential to handle links are depicted in Fig. 10. There is a notable correspondence between the $\exists_{link}$-rule and our $\stackrel{\sqsupseteq}{\to}$-rule and also between the $\forall_{link}$-rule and our $\stackrel{\sqsubseteq}{\to}$-rule. This is not that surprising, given the known correspondence between DDL and $\mathcal{E}$-connections (Kutz et al. 2004). Given the nature of E-connections the $\forall_{link}$-rule works exactly the other way around, compared to our $\stackrel{\sqsubseteq}{\to}$-rule, and in this respect the instantiation of a j-tree by one application of the $\exists_{link}$-rule and possibly multiple applications of the $\forall_{link}$-rule resembles an unsatisfiability check call from the $\mathfrak{B}_{ij}$-rule of the original DDL algorithm. Remarkably, the exact mechanism how this happens within the $\mathcal{E}$-connections algorithm seems to be more transparent and prone to optimization. The algorithm introduced in this paper in addition keeps track on the domain relation, which is needed to handle subsumption propagation along chains of bridge rules. There is no domain relation within $\mathcal{E}$-connections, and hence no need for such a feature.

$\exists_{link}$**-rule:**
If $\exists E.C \in \mathcal{L}_i(x)$, $E \in \epsilon_{ij}$, x is not blocked and x has no E-successor y with $\mathcal{L}(\langle x, y,\rangle) = \{E\}$,
then create a new E-successor (a j-node) y of x with $\mathcal{L}_j(y) = \{C\}$.
(The new j-tree rooted in y will not be expanded until no more rules apply to the i-tree.)

$\forall_{link}$**-rule:**
If $\forall E.C \in \mathcal{L}_i(x)$, $E \in \epsilon_{ij}$, x is not blocked and there is an E-successor y of x such that $C \notin \mathcal{L}_j(y)$
then set $\mathcal{L}_j(y)$ to $\mathcal{L}_j(y) \cup \{C\}$.

Figure 10: The tableaux expansion rules of the algorithm for $\mathcal{E}$-connections (Cuenca Grau et al. 2004).

Yet another point of view on distributed ontologies is Package-based description logics, or P-DL (Bao et al. 2006). In P-DL, the intuition of importing is pursued: every concept name belongs to some local ontology, but can be also imported to and used by other ontologies in the system. The P-DL semantics uses domain relations. In contrast to other approaches, only one-to-one domain relations are allowed and also compositional consistency (that we have borrowed and applied on DDL) is required. Local domains in P-DL semantics are viewed as partially overlapping, and not disjoint. As a result, some of the problems known for DDL, such as restricted knowledge propagation between remote ontologies, that we

also address in this paper, are not an issue for P-DL according to (Bao et al. 2006). On the other hand, in P-DL reasoning over a distributed ontology is always equivalent to reasoning over the union of local ontologies, which is not a goal of DDL. The distributed tableaux algorithm for P-DL is introduced in (Bao et al. 2006). It uses $\mathcal{ALC}$ as local language and requires acyclic importing relation. It uses message-based protocol, similar to the ours, and keeps track of the domain relation. Since imported concepts may appear anywhere in the importing ontology, messages are invoked directly from $\mathcal{ALC}$-tableaux rules.

A completely different approach to distributed ontology reasoning is taken in (Schlicht & Stuckenschmidt 2008), where distributed reasoning algorithm based on resolution techniques is introduced. Yet another distributed ontology framework called Integrated Distributed Description Logics (IDDL) is introduced in (Zimmermann 2007), where also an incomplete decision procedure is outlined.

7 Conclusion and Future Work

In a recent paper (Homola 2008) we have proposed an adjusted semantics for DDL, dubbed "DDL under compositional consistency", that features improved subsumption propagation between remote ontologies. More specifically, subsumption propagates along chains of bridge rules that span throughout multiple ontologies. In this paper we take steps forward in order to develop a distributed tableaux reasoning algorithm that would decide satisfiability of concepts and subsumption with respect to the adjusted semantics. The algorithm that is introduced in this paper handles chaining onto-bridge rules correctly, but it is unable to cope with chaining into-bridge rules. We provide precise characterization of the semantics that the algorithm actually implements.

As in the case of the tableaux algorithm that is known for the original semantics (Serafini et al. 2005), the newly introduced algorithm is also truly distributed and it permits the scenario where every local ontology is governed by an autonomous reasoning service and these services communicate by passing queries. The local reasoner that has started the computation collects all the answers and makes the decision at the end. To demonstrate this fact more clearly, we have provided a message-based protocol for the algorithm. Besides the fact that each of these algorithms works under a different semantics, the main distinguishing feature of the newly introduced algorithm is that communication between local reasoners is divided into multiple messages. Computation is sparked in a remote reasoner at some point and both reasoners continue to run independently. If subsumption propagation is proved by the remote reasoner, the local reasoner is acknowledged by a message. This possibly repeats several times, if subsumption is proved between different pairs of concepts. We believe that such behaviour may serve as a base for more fine-grained optimization in future.

The two most prominent open issues put forward by this paper are: extending the algorithm so that it would handle the DDL under compositional consistency semantics in full extent, that is, dealing with chaining into-bridge rules; and, extending the algorithm towards more expressive DL, since the current version only supports $\mathcal{ALC}$ as the local representation language and requires acyclic concept mapping. Computational handling of distributed knowledge bases in which the concept mapping is not necessarily acyclic is an interesting problem, which is to our best knowledge still unresolved also for the original DDL framework. As is the problem of extending the DDL framework with nominals. We would like to address these issues in the near future. Remaining research problems that are closely related to this work include practical evaluation of the algorithm by implementation, complexity analysis for the decision problems, and combining, within a unified framework, the current approach with that of (Homola 2007), which has addressed the problem of interaction between bridge-rules in DDL.

Acknowledgement

Support from the VEGA project of Slovak Ministry of Education and Slovak Academy of Sciences no. 1/0173/03 and from the grant GUK no. UK/365/2008 awarded by Comenius University is gratefully acknowledged. Special thanks to Ján Kľuka for advanced LaTeX symbol hacking.

References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P., eds (2003), *The Description Logic Handbook*, Cambridge University Press.

Bao, J., Caragea, D. & Honavar, V. G. (2006), A distributed tableau algorithm for package-based description logics, *in* 'Procs. of CRR 2006'.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001), 'The semantic web', *Scientific American* **284**(5), 34–43.

Borgida, A. & Serafini, L. (2003), 'Distributed description logics: Assimilating information from peer sources', *Journal of Data Semantics* **1**.

Cuenca Grau, B., Parsia, B. & Sirin, E. (2004), Working with multiple ontologies on the semantic web., *in* 'Procs. of ISWC2004', Vol. 3298 of *LNCS*, Springer.

Homola, M. (2007), Distributed description logics revisited, *in* 'Procs. of DL-2007', Vol. 250 of *CEUR-WS*.

Homola, M. (2008), Subsumption propagation between remote ontologies in distributed description logic, *in* 'Procs. of DL2008', Vol. 353 of *CEUR-WS*.

Horrocks, I., Sattler, U. & Tobies, S. (1999), Practical reasoning for expressive description logics, *in* 'Procs. of LPAR'99', number 1705 *in* 'LNAI', Springer, pp. 161–180.

Kutz, O., Lutz, C., Wolter, F. & Zakharyaschev, M. (2004), '$\mathcal{E}$-connections of abstract description systems', *Artificial Intelligence* **156**(1), 1–73.

Schlicht, A. & Stuckenschmidt, H. (2008), Distributed resolution for alc, *in* 'Procs. of DL2008', Vol. 353 of *CEUR-WS*.

Serafini, L., Borgida, A. & Tamilin, A. (2005), Aspects of distributed and modular ontology reasoning, *in* 'Procs. of IJCAI'05', pp. 570–575.

Serafini, L. & Tamilin, A. (2004), Local tableaux for reasoning in distributed description logics, *in* 'Procs. of DL'04', CEUR-WS.

Serafini, L. & Tamilin, A. (2005), DRAGO: Distributed reasoning architecture for the semantic web, *in* 'Procs. of ESWC'05', Vol. 3532 of *LNCS*, Springer-Verlag, pp. 361–376.

Zimmermann, A. (2007), Integrated distributed description logics, *in* 'Procs. of DL-2007', Vol. 250 of *CEUR-WS*.