Generalized Link Properties for Expressive $\mathcal{E}$-Connections of Description Logics *

Bijan Parsia
University of Maryland, MIND Lab
8400 Baltimore Avenue,
College Park, MD, 20742 USA
bparsia@isr.umd.edu

Bernardo Cuenca Grau
Departamento de Informatica, Univ. Valencia
Av. Vicente Andres Estelles s/n
Burjassot, Valencia, Spain
bernardo@mindlab.umd.edu

Abstract

$\mathcal{E}$-Connections are a robust framework for combining in a decidable way several families of decidable logics, including Description Logics (DLs), Modal Logics, and many logics of time and space. $\mathcal{E}$-Connections have also proved to be useful for supporting modular, distributed modeling such as is becoming common on the Semantic Web. In this paper, we present an extension to $\mathcal{E}$-Connections of DLs that provides more flexibility in the way link properties can be defined and used in a combination of ontologies. We also provide means for defining transitive relations across domains and for simulating some of the expressivity of the transitive closure operator. Finally, we provide a tableau-based decision procedure for two relevant $\mathcal{E}$-Connection languages involving the influential DLs $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHIO}$, which are at the basis of the Web Ontology Language (OWL)

Motivation

For many years, the modal logic community has pursued various techniques for robustly combining logics, including fusion, fibring and multi-dimensional modal logics. Recently, a striking framework, $\mathcal{E}$-Connections (Kutz *et al.* 2004), has been proposed with a number of desirable properties. $\mathcal{E}$-Connections provide a framework for combining in a decidable way several families of decidable logics

Robust decidability is achieved by imposing strong restrictions in the way the logics can be combined. For example, in the case of $\mathcal{E}$-Connections of DLs, each component is strictly disjoint from all the others and interpreted in a disjoint domain. Each component can contain *link properties* which connect individuals in one domain with individuals in another. Classes in a component can be built up out of restrictions on link properties. These restrictions turn out to work well for a number of modeling situations.

However, there are further restrictions imposed by the original $\mathcal{E}$-Connections framework that do inhibit useful modeling and turn out to be inessential, at least in combinations involving only DLs.

In this paper, we provide syntax, semantics, and a decision procedure for extended $\mathcal{E}$-Connections of the logics $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHIO}$ (Horrocks, Sattler, & Tobies 2000) (Horrocks & Sattler 2001), in which some important constraints in the way link properties can be defined and used are lifted. Such an extension provides modeling benefits in relevant applications of DLs, such as Semantic Web (Cuenca-Grau, Parsia, & Sirin 2005) as well as Knowledge Representation in biological, medical and manufacturing domains, in which partinomy relations are crucial.

Overview of the Extensions

Link Properties Defined in and Pointing to Multiple Ontologies

Suppose we are building a KB with information about people, pets and leisure activities, which includes the following axioms:

$$Lover \equiv Person \sqcap \exists loves.Person$$
$$PetLover \equiv Person \sqcap \exists loves.Pet$$
$$UnfriendlyPet \equiv Pet \sqcap \neg(\exists loves.Person)$$
$$FilmLover \equiv Person \sqcap \exists loves.Cinema$$

For modularity reasons, we would like to use $\mathcal{E}$-Connections to represent the knowledge about people, pets and leisure activities into distinct, yet connected, components, say $\mathcal{K}_1, \mathcal{K}_2$ and $\mathcal{K}_3$ respectively. However, although the separation seems natural (the domains modeled by the different components are disjoint), the current formalism does not allow to represent such a combination, since:

- A link property cannot be defined in two different components, i.e. the property *loves* cannot be used in $\mathcal{K}_1$ and $\mathcal{K}_2$ at the same time for defining, for example, a *pet lover* and an *unfriendly pet*. On the other hand, it is not possible to define in a certain component a link property pointing to several different domains, i.e. we cannot use in $\mathcal{K}_1$ the link property *loves* to define *pet lover* and *film lover*, since the concepts *pet* and *cinema* belong to different components of the $\mathcal{E}$-Connection.
- The same property cannot be defined both as a role and a link property. In other words, we cannot use the same

*We would like to thank Evren Sirin for useful discussions. Mindswap funding is provided by: Fujitsu Labs of America, Lockheed Martin Advanced Tech. Lab., NTT Corp., Kevric Corp., SAIC, NSF, National Geospatial-Intelligence Agency, DARPA, US Army Research Laboratory, NIST, Other DoD sources. The second author is supported by a FPU Scholarship from the Spanish Government and is currently a visitor at the MIND Lab

property *loves* to represent the relation of a person loving another person and of a person loving a pet.

In each of these cases, the current framework would either force us to coin new properties or to merge different components. The former would result in an unnecessary proliferation of properties, while the latter constrains the way knowledge can be modularized. These limitations constrain the flexibility of the formalism from a modeling perspective.

The main idea of the extension presented here is to explicitly point out the target domain of a property whenever it is used in a certain component. For example, in $\mathcal{K}_1$ we would define:

$$Lover \equiv Person \sqcap \exists loves^{(1)}.Person$$
$$PetLover \equiv Person \sqcap \exists loves^{(2)}.Pet$$
$$FilmLover \equiv Person \sqcap \exists loves^{(3)}.Cinema$$

A concept like $\exists loves^{(1)}.Person$, when used in $\mathcal{K}_1$, would represent the set of people who love a person (i.e, here the property would be acting as a role). However, when used in $\mathcal{K}_2$:

$$UnfriendlyPet \equiv Pet \sqcap \neg(\exists loves^{(1)}.Person)$$

the concept would represent the set of pets who love a person (i.e., *loves* would act as a link property from $\mathcal{K}_2$ to $\mathcal{K}_1$). Hence, the way a property is interpreted depends on the component it is used and on its explicit superscript. Note that, since the disjointness between the interpretation domains of different components is preserved, the logical interpretation of a property P used in $\mathcal{K}_i$ when its target domain is $\mathcal{K}_j$ is disjoint with the interpretation of the same property when used in $\mathcal{K}_k$ pointing to $\mathcal{K}_m$ (for either $k \neq i$ or $j \neq m$). The actual interpretation of P in the $\mathcal{E}$-Connection would then be the (disjoint) union of its interpretation in each of its different "contexts" (see Equation 1 in Definition 1).

Transitive Relations in $\mathcal{E}$-Connections

We propose an extension of $\mathcal{E}$-Connections that allows to define transitive relations across domains. The advantages of our approach are the following:

1. It provides a fine-grained control over the entailments caused by the transitivity of a relation. In such a framework it is possible to "switch on and off" transitivity when moving from one component to another within an $\mathcal{E}$-Connection
2. It allows to reproduce the expressivity of the transitive closure operator in many cases without paying a computational cost

Combining Ontologies using transitive Link Properties Transitive roles are a key component of modern DL languages (Sattler 1996),(Horrocks, Sattler, & Tobies 2000), since they provide a key expressivity for many applications at a "low" computational cost. When a role is defined to be transitive, it is interpreted as a transitive relation. For example, in a KB about wines, if we describe a *RiojaWinery* as a winery that is *locatedIn* Rioja, which is a region *locatedIn* Spain, and define *locatedIn* to be transitive, a reasoner would infer that a *RiojaWinery* is *locatedIn* Spain.

We would like to use $\mathcal{E}$-Connections to represent the knowledge on wineries and the knowledge about regions in different components, say $\mathcal{K}_1$ and $\mathcal{K}_2$ respectively. In such a representation, the two components would be connected by the property *locatedIn*. However, $\mathcal{E}$-Connection do not allow to define a link property to be transitive and the KBs could not be separated without losing the entailment $RiojaWinery \sqsubseteq \exists locatedIn.SpainRegion$. We propose to extend the $\mathcal{E}$-Connections formalism with a new kind of axiom which "points out" in which cases a property must be interpreted as a transitive relation. In the example above, we could split the knowledge as follows:

$$\mathcal{K}_1 : RiojaWinery \sqsubseteq$$
$$Winery \sqcap \exists locatedIn^{(2)}.RiojaRegion$$
$$\mathcal{K}_2 : RiojaRegion \sqsubseteq SpanishRegion$$
$$Trans(locatedIn; (1,2), (2,2))$$

Intuitively the last axiom states that, for x, y, z arbitrary regions and w an arbitrary winery, if $locatedIn(w, x)$ and $locatedIn(x, y)$, then a reasoner should infer that $locatedIn(w, y)$ holds. Analogously, if $locatedIn(x, y)$ and $locatedIn(y, z)$ then we can conclude $locatedIn(x, z)$. The numbers in the transitivity axiom, analogously to the superscripts in link properties, refer to components of the $\mathcal{E}$-Connection. A formal semantics for transitivity assertions their most general form will be provided later.

Controlling transitivity Suppose that in a DL KB the relation *partOf* is defined to be transitive. Suppose that we have the following ABox:

$$Person(joe); Finger(joeFinger); Hand(joeHand);$$
$$ResearchLab(lab); University(univ);$$
$$partOf(joeFinger, joeHand); partOf(joeHand, joe);$$
$$partOf(joe, lab); partOf(lab, univ);$$

A DL reasoner would not only infer that Joe's finger is a part of Joe and that Joe is a part of *univ*, but also that Joe's finger and hand are a part of both *lab* and *univ*. In DLs with transitive roles, it is not possible to keep the desired entailments while ruling out the undesirable ones without coining a new property that would replace *partOf* in some cases.

Suppose now that we use an $\mathcal{E}$-Connection. A component $\mathcal{K}_1$ would contain information about human body parts, such as hand and finger; a second component $\mathcal{K}_2$ would represent information about people, such as Joe, and finally a third component $\mathcal{K}_3$ would model the domain of academic institutions, such as research labs and universities. Then, the property *partOf* is defined as a role in $\mathcal{K}_1$ relating different body parts, a link property from $\mathcal{K}_1$ to $\mathcal{K}_2$ relating body parts to persons, a link property from $\mathcal{K}_2$ to $\mathcal{K}_3$ relating people to institutions and a role in $\mathcal{K}_3$, relating institutions to other institutions. In order to rule out the undesired entailments, we must distinguish two different *transitivity chains*: the first one would propagate transitivity within $\mathcal{K}_1$ and from $\mathcal{K}_1$ to $\mathcal{K}_2$, while the second one would propagate transitively *partOf* relations starting in $\mathcal{K}_2$ and within $\mathcal{K}_3$. For such a purpose, we introduce in the $\mathcal{E}$-Connection the following axioms:

$$Trans(partOf;(1,1),(1,2))$$
$$Trans(partOf;(2,3),(3,3))$$

The first axiom allows to infer that Joe's finger is a part of Joe, provided that his finger is a part of his hand and his hand a part of him. Analogously, the second axiom would entail that Joe is a part of *univ*, provided that he is a part of the *lab* and the *lab* is a part of *univ*. Please, note that we are not imposing any transitivity along chains starting in $\mathcal{K}_1$ and ending in $\mathcal{K}_3$. In other words, the above two axioms are *not* equivalent to the following assertion:

$$Trans(partOf;(1,1),(1,2),(2,3),(3,3))$$

This allows us to rule out the undesired entailments.

Simulating transitive closure The DL $\mathcal{SHOIN}$ provides powerful means for representing different different kinds of part-whole relationships, such as transitive and functional roles, role inversion and role hierarchies. However, sometimes the expressivity provided by $\mathcal{SHOIN}$ does not suffice. For example, suppose that devices, batteries and engines are disjoint concepts and we want to describe devices whose parts can only be batteries or engines and such that all their direct parts are engines and must have *at some level of decomposition* a battery. The logic $\mathcal{SHOIN}$ does not provide the expressivity for distinguishing direct from indirect parts. For such a purpose, a logic including the *transitive closure* operator "$^+$" would be required (Sattler 1996). This operator, when applied to property P, represents the smallest transitive relation containing P. The devices mentioned before could be described as follows:

$$Device \sqcap \forall hasPart.Engine \sqcap \exists hasPart^+.Battery \sqcap \forall hasPart^+.(Engine \sqcup Battery)$$

Although the transitive closure operator provides key expressive power for many applications, it's inclusion in a DL is not without a price, since it complicates its implementation, increases the computational complexity of a logic and brings it beyond First Order Logic (Sattler 1996).

Let us consider again the example about devices. Since devices and batteries are disjoint concepts, we can place them in different components of an $\mathcal{E}$-Connection. Suppose we represent devices in $\mathcal{K}_1$, and we represent *only* engines in $\mathcal{K}_2$ and *only* Batteries in $\mathcal{K}_3$, i.e. $Device \in \mathcal{K}_1$, $Engine \in \mathcal{K}_2$, and $Battery \in \mathcal{K}_3$, with $\top_2 \equiv Engine$ and $\top_3 \equiv Battery$, where $\top_2$ and $\top_3$ stand for the universal concept in $\mathcal{K}_2$ and $\mathcal{K}_3$ respectively. Then, the following concept:

$$Device \sqcap \neg(\exists hasPart^{(1)}.\top_1) \sqcap \neg(\exists hasPart^{(3)}.Battery) \sqcap \exists hasPart^{(2)}.Engine \sqcap \exists hasPart^{(2)}.(\exists hasPart^{(3)}.Battery)$$

Together with the transitivity axiom:

$$Trans(hasPart;(1,2),(2,3),(2,2))$$

adequately simulate the effect of transitive closure. The conjuncts $\neg(\exists hasPart^{(3)}.Battery)$ and $\neg(\exists hasPart^{(1)}.\top_1)$ ensure that the device does not have a direct part that is *not* an engine; the conjunct $\exists hasPart^{(2)}.Engine$ guarantees that the device has a direct part which is an engine; the conjunct $\exists hasPart^{(2)}.(\exists hasPart^{(3)}.Battery)$ makes sure that at least one of the engines of the device has a part which is a battery. Finally, the transitivity axiom states that, if the device has a part which is an engine and that engine has a battery, then such a battery is a part of the device.

However, it is not possible to simulate transitive closure in all cases. As an example, let us define a queen as follows (Sattler 1996):

$$Queen \equiv Woman \sqcap \forall hasChild.(Princess \sqcup Prince) \sqcap \forall hasChild^+.Noble$$

In this case, we cannot separate the concepts into different components since queens, princes and princesses are all nobles.

Syntax and Semantics

In this section, we define the language $\mathcal{C}^{\epsilon}_{\mathcal{HQ}^+}(\mathcal{SHIQ},\mathcal{SHOQ},\mathcal{SHIO})$, which allows to define $\mathcal{SHIQ},\mathcal{SHOQ}$ and $\mathcal{SHIO}$ KBs and combine them using link properties, which can be applied in existential, universal and number restrictions as well as organized in hierarchies.

Definition 1 *($\mathcal{C}^{\epsilon}_{\mathcal{HQ}^+}(\mathcal{SHIQ},\mathcal{SHOQ},\mathcal{SHIO})$ Syntax and Semantics)*

Let $\{m_1,m_2,...,m_n\}$ be a set of constants where each m_i, $i=1,...,n$ is either equal to '$\mathcal{SHIQ}$', '$\mathcal{SHOQ}$' or '$\mathcal{SHIO}$'. These indexes indicate the logic in which each component is written.For $i\in\{1,2,...,n\}$ and $j\in\{1,...,n\}$ let $V_{C_i}^{m_i}, V_{I_i}^{m_i}$ be disjoint sets of concept and individual names respectively. Let $\epsilon_{ij}^{m_i}$ be sets of property names, not necessarily pair-wise disjoint, but disjoint with respect to the other sets of names. For $i,j=1,...,n$, the set ρ_{ij} of ij-properties is defined as follows:

- *If $i=j$ and $m_i\in\{$'$\mathcal{SHIQ}'$,'$\mathcal{SHIO}'\}$, then $\rho_{ij}=\epsilon_{ij}^{m_i}\cup\{Inv(P_{(ji)})|P\in\epsilon_{ji}^{m_j}\}$. If $m_i=$'$\mathcal{SHOQ}'$, $\rho_{ij}=\epsilon_{ij}^{m_i}$*
- *If $i\neq j$, then $\rho_{ij}=\epsilon_{ij}^{m_i}$.*

For $P_1,P_2,P\in\rho_{ij}$, an ij-property axiom is an assertion of the form $P_1\sqsubseteq P_2$. An ij-property box $\Re_{ij}$ is a finite set of ij-property axioms. A transitivity axiom is of the form:

$$Trans(P;(i_1,j_1),...,(i_p,j_p))$$

where P must be defined in each of $\epsilon_{i_k,j_k}^{m_{i_k}},\forall k=1,...,p$. We say that P is transitive for $(i_1,j_1),...,(i_p,j_p)$ The combined property box $\Re$ contains each of the property boxes plus all transitivity axioms. An ij-property P is called simple if for $\sqsubseteq^*$ the transitive reflexive closure of $\sqsubseteq$ on $\Re_{ij}$, and for each $S\in\rho_{ij}$, $S\sqsubseteq^* P$ implies that S is not transitive in (i,j) for any transitivity axiom.

The sets of i-concepts are defined by simultaneous induction as the smallest sets such that each concept name $A\in V_{C_i}^{m_i}$ and $\top_i$, are i-concepts and, for $o\in V_{I_i}^{m_i}$ for $m_i\in\{$'$\mathcal{SHOQ}$','$\mathcal{SHIO}'\}$, $P,S\in\rho_{ij}$, S simple and such that, if $i=j$, then $m_i\in\{$'$\mathcal{SHIQ}$','$\mathcal{SHOQ}'\}$, C,C_1,C_2 i-concepts, and Z a j-concept, the constructions shown in Figure 1 are also valid i-concepts.

Syntax and Semantics
$A \in V_{C_i}^{m_i}; A^{\mathcal{M}_i} \subseteq W_i$
$\top_i^{\mathcal{M}_i} = W_i$
$\bot_i^{\mathcal{M}_i} = \emptyset$
$P \in \rho_{ij}; P^{\mathcal{M}_{ij}} \subseteq W_i \times W_j$
$(Inv(Q_{(ji)}))^{\mathcal{M}_{ij}} = \{(x,y) \in O_i \times O_j \mid (y,x) \in Q^{\mathcal{M}_{ji}}\}$
$\{o\}^{\mathcal{M}_i} \subseteq W_i, \|\{o\}^{\mathcal{M}_i}\| = 1$
$(\neg C)^{\mathcal{M}_i} = W_i / C^{\mathcal{M}_i}$
$(C \sqcap D)^{\mathcal{M}_i} = C^{\mathcal{M}_i} \cap D^{\mathcal{M}_i}$
$(C \sqcup D)^{\mathcal{M}_i} = C^{\mathcal{M}_i} \cup D^{\mathcal{M}_i}$
$(\exists P^{(j)}.Z)^{\mathcal{M}_i} = \{x \in W_i \mid \exists y \in W_j, (x,y) \in P^{\mathcal{M}_{ij}}, y \in Z^{\mathcal{M}_j}\}$
$(\forall P^{(j)}.Z)^{\mathcal{M}_i} = \{x \in W_i \mid \forall y \in W_j, (x,y) \in P^{\mathcal{M}_{ij}} \rightarrow y \in Z^{\mathcal{M}_j}\}$
$(\geq nS^{(j)}.Z)^{\mathcal{M}_i} = \{x \in W_i, \|y \in W_j, (x,y) \in S^{\mathcal{M}_{ij}}, y \in Z^{\mathcal{M}_{ij}}\| \geq n\}$
$(\leq nS^{(j)}.Z)^{\mathcal{M}_i} = \{x \in W_i, \|y \in W_j, (x,y) \in S^{\mathcal{M}_{ij}}, y \in Z^{\mathcal{M}_{ij}}\| \leq n\}$

Table 1: Semantics of i-concepts

A combined TBox is a tuple $\mathcal{K} = (\mathcal{K}_1, \mathcal{K}_2, ..., \mathcal{K}_n)$, with $\mathcal{K}_i$ the set of assertions of the form $C \sqsubseteq D$, with C, D i-concepts. A combined KB is a pair $\Sigma = (\mathcal{K}, \Re)$.

The semantics is given by means of an interpretation:

$$\mathcal{M} = (\{\mathcal{M}_i,\}_{1 \leq i \leq n}, \{\mathcal{M}_{ij}\}_{1 \leq i,j \leq n})$$

Where $\mathcal{M}_i = (W_i, .^{\mathcal{M}_i}), \mathcal{M}_{ij} = (W_i, W_j, .^{\mathcal{M}_{ij}})$ and where the interpretation domains are disjoint: $W_i \cap W_j = \emptyset, \forall i \neq j$. Let $P \in \rho_{i_1,j_1}, ..., P \in \rho_{i_p,j_p}$, then:

$$P^{\mathcal{M}} = \biguplus_{k=1,...,p} P^{\mathcal{M}_{i_k j_k}} \quad (1)$$

An i-concept C is interpreted as a subset of W_i (see Table 1). An interpretation satisfies an ij-property axiom $P_1 \sqsubseteq P_2$ iff $P_1^{\mathcal{M}_{ij}} \subseteq P_2^{\mathcal{M}_{ij}}$ and it satisfies the ij-property box $\Re_{ij}$ iff it satisfies all the assertions in it. An interpretation satisfies a transitivity axiom $Trans(P; (i_1, j_1), ..., (i_p, j_p))$ iff $(P^{\mathcal{M}_{i_1 j_1}} \cup ... \cup P^{\mathcal{M}_{i_p j_p}})$ is transitive. Finally, an interpretation satisfies the combined property box $\Re$ iff it satisfies all its component property boxes and all its transitivity axioms.

If C, D are i-concepts, $\mathcal{M} \models (C_1 \sqsubseteq C_2)$ iff $C_1^{\mathcal{M}_i} \subseteq C_2^{\mathcal{M}_i}$, and satisfies the combined TBox $\mathcal{K}$ iff it satisfies all the axioms in each of its component $\mathcal{K}_i$. Finally, $\mathcal{M} \models \Sigma$ iff it satisfies both $\mathcal{K}, \Re$. An i-concept C is satisfiable w.r.t. a combined knowledge base Σ iff there is an interpretation $\mathcal{M}$ s.t $\mathcal{M} \models \Sigma$, and $C^{\mathcal{M}_i} \neq \emptyset$.

Note that Equation 1 establishes that the interpretation of a property in the $\mathcal{E}$-Connection is obtained as the union of its interpretation in each of the components of the combination.

A Tableau for $\mathcal{C}^{\epsilon}_{\mathcal{HQ}^+}(\mathcal{SHIQ}, \mathcal{SHOQ}, \mathcal{SHIO})$

We assume all concepts written in Negation Normal Form (NNF). To transform an i-concept into NNF, negation is pushed inwards using De Morgan's laws and the following equivalences:

$$\neg\exists P^{(j)}.Z \equiv \forall P^{(j)}.\neg Z \qquad \neg\forall P^{(j)}.Z \equiv \exists P^{(j)}.\neg Z$$
$$\neg \leq nS^{(j)}.Z \equiv \geq (n+1)S^{(j)}.Z$$
$$\neg \geq (n+1)S^{(j)}.Z \equiv \leq nS^{(j)}.Z \qquad \neg(\geq 0S^{(j)}.Z) \equiv \bot_i$$

Let X be an i-concept, $\Re$ a combined property box. We define for $i = 1, ..., n$, the set $sub_i(X, \Re)$ of i-subconcepts of X as follows:

- If $X \in V_{C_i}^{M_i}$, the negation of a concept name $V_{C_i}^{M_i}$, or a nominal $\{o\}$, then $sub_i(X, \Re) = \{X\}$, and $sub_j(X, \Re) = \emptyset, \forall j \neq i$
- If X is of the form $C_1 \sqcap C_2$, or $C_1 \sqcup C_2$ where C_1 and C_2 are i-concepts, then $sub_i(X, \Re) = \{X\} \cup sub_i(C_1, \Re) \cup sub_i(C_2, \Re)$, $sub_j(X, \Re) = sub_j(C_1, \Re) \cup sub_j(C_2, \Re)$, $\forall j \neq i$
- If X is of the form $\exists P^{(k)}.Z$, or $\forall P^{(k)}.Z$, or $\leq nS^{(k)}.Z$, or $\geq nS^{(k)}.Z$ then $sub_i(X, \Re) = \{X\} \cup sub_i(Z, \Re)$; $sub_j(X, \Re) = sub_j(Z, \Re), \forall j \neq i$

For X and a combined KB $\Sigma = (\mathcal{K}, \Re)$, we define, for $i = 1, ..., n$:

$$clos_i(X, \Sigma) = clos_i(X, \Re) \bigcup_{j=1,...,n} clos_i(C_{\mathcal{K}_j}, \Re)$$

Where $clos_i(X, \Re)$ includes both the concepts in $sub_i(X, \Re)$ and their NNF. The concept $C_{\mathcal{K}_j}$, defined as follows:

$$C_{\mathcal{K}_j} = \sqcap_{C \sqsubseteq D \in \mathcal{K}_j} (\neg C \sqcup D)$$

verifies that $\mathcal{M} \models \mathcal{K}_j$ iff $\mathcal{M} \models (\top_j \equiv C_{\mathcal{K}_j})$.

Definition 2 *Let X be an i-concept in NNF, and $\Sigma = (\mathcal{K}, \Re)$ a combined knowledge base. A **combined tableau** $\mathcal{T}$ for X w.r.t. Σ is a tuple $\mathcal{T} = (\{O_i\}, \{L_i\}, \{\alpha_{ij}\}), i, j = 1, ..., n$, where O_i is a set of individuals, $L_i : O_i \rightarrow 2^{clos_i(X,\Sigma)}$ maps each individual to a set of concepts in $clos_i(X, \Sigma)$, and $\alpha_{ij} : \rho_{ij} \rightarrow 2^{O_i \times O_j}$ maps each ij-property to a set of pairs of individuals.*

There must exist n individuals $o_p \in O_p$ such that $\{X, C_{\mathcal{K}_p}\} \subseteq L_p(o_p)$ if $p = i$ and $\{C_{\mathcal{K}_p}\} \subseteq L_p(o_p)$ otherwise, $\forall p = 1, ..., n$. Furthermore, $\forall p = 1, ..., n$ and $\forall s \in O_p$, then $C_{\mathcal{K}_p} \in L_p(s)$.

For all $i = 1, ..., n; j = 1, ..., n$, all $s \in O_i, t \in O_j, u \in O_k, C, D \in clos_i(X, \Sigma)$, $P, S, P_1, P_2 \in \rho_{ij}, Z \in clos_j(X, \Sigma)$, S simple and:

$$S_{ij}^{\mathcal{T}}(s, C) = \{t \in O_j \mid (s,t) \in \alpha_{ij}(S), and\ C \in L_j(t)\}$$

the following conditions hold:

1. *If $C \in L_i(s)$, then $\neg C \notin L_i(s)$*
2. *If $(C \sqcap D) \in L_i(s)$, then $C \in L_i(s)$, and $D \in L_i(s)$*
3. *If $(C \sqcup D) \in L_i(s)$, then $C \in L_i(s)$, or $D \in L_i(s)$*
4. *If $\forall P^{(j)}.Z \in L_i(s)$, and $(s,t) \in \alpha_{ij}(P)$, then $Z \in L_j(t)$*
5. *If $\exists P^{(j)}.Z \in L_i(s)$, then there is some $t \in O_j$ such that $(s,t) \in \alpha_{ij}(P)$, and $C \in L_j(t)$*
6. *If $(s,t) \in \alpha_{ij}(P)$, $(t,u) \in \alpha_{jk}(P)$ and $Trans(P; ...(i,j), ..., (j,k), ...) \in \Re$, then $(s,t) \in \alpha_{ik}(P)$*
7. *If $(s,t) \in \alpha_{ij}(P_1)$ and $(P_1 \sqsubseteq^* P_2) \in \Re_{ij}$, then $(s,t) \in \alpha_{ij}(P_2)$*
8. *If $(\leq nS^{(j)}.Z) \in L_i(s)$, then $\|(S_{ij}^T(s, Z))\| \leq n$*
9. *If $(\geq nS^{(j)}.Z) \in L_i(s)$, then $\|(S_{ij}^T(s, Z))\| \geq n$*

10. *If* $\{(\geq nS^{(j)}.Z), (\leq nS^{(j)}.Z)\} \cap L_i(s) \neq \emptyset$, *and* $(s,t) \in \alpha_{ij}(P)$, *then* $\{C, \sim C\} \cap L_j(t) \neq \emptyset$
11. $(s,t) \in \alpha_{ij}(P)$ *iff* $(t,s) \in \alpha_{ji}(Inv(P_{ij}))$
12. *If* $\{o\} \in L_i(s) \cap L_i(t)$, *then* $s = t$

Lemma 1 *An i-concept X in NNF is satisfiable w.r.t. a combined KB* $\Sigma = (\mathcal{K}, \Re)$ *iff X has a combined tableau w.r.t.* Σ.

Detailed proofs for all the results presented in this paper are available online [1]

A Tableau Algorithm for $\mathcal{C}^{\epsilon}_{\mathcal{HQ}^+}(\mathcal{SHIQ}, \mathcal{SHOQ}, \mathcal{SHIO})$

Let $\Sigma = (\mathcal{K}, \Re)$ be a combined KB and X an i-concept written in NNF. A **combined completion graph** $\mathcal{G}$ for X w.r.t. Σ is defined such that it contains n kinds of nodes, called i-nodes $i = 1, ..., n$.An i-node x is labeled with a set:

$$L_i(x) \subseteq clos_i(X, \Sigma) \cup \{\uparrow (P^{(j)}, \{o\})\}$$

for $P \in \rho_{ij}$, and $o \in V_{I_i}^{m_j}\}$. For each $o \in V_{I_i}^{m_i}$, with $m_i \in \{\text{`}\mathcal{SHOQ}', \text{`}\mathcal{SHIO}'\}$, there is a distinguished i-node x_o, with $\{o\} \in L_i(x_o)$. We keep track of inequalities between nodes using symmetric binary relationships $\neq_i$.

Edges $< x, y >$, where x is an i-node and y a j-node are labeled with a set $L(< x, y >)$. We use $\uparrow (P^{(j)}, \{o\}) \in L_i(y)$ to represent an $P^{(j)}$-labeled edge from the i-node y to the distinguished j-node x_o.

If the i-node x is connected by an edge $< x, y >$ to the j-node y, then y is a *successor* of x and x is a *predecessor* of y. *Ancestor* is the transitive closure of predecessor. A j-node y is a $P^{(j)}$-successor of an i-node x if, for some P' with $P' \sqsubseteq^* P^{(j)}$ in $\Re_{ij}$, either y is a successor of x and $P'^{(j)} \in L(< x, y >)$ or $(\uparrow P'^{(j)}, \{o\}) \in L_i(x)$ and $y = x_o$. A j-node y is a $P^{(j)}$-neighbor of an i-node x if either y is a successor of x and $P^{(j)} \in L(< x, y >)$, or if y is a predecessor of x and $Inv(P_{(ij)}) \in L(< y, x >)$. A j-node y is a P-ancestor of an i-node x, if y,x are connected by a chain of nodes $y, z_1, ..., z_p, x$ such that z_k is a P-successor of z_{k-1}, $\forall k = 1, .., p$. If an edge is **not** added by the $\rightarrow TRANS$ rule it is called a direct edge [2]. We define direct successor and direct predecessor analogously. Note that direct successors/predecessors are also successors/predecessors according to the definition.

An i-node is *blocked* iff it is directly blocked or indirectly blocked. An i-node x is directly blocked by an i-node y iff none of its ancestors is blocked and:

- If x is successor of a j-node x', with $j \neq i$, y is an ancestor of x such that $L_i(x) \subseteq L_i(y)$
- If x is a direct successor of a j-node x', with $j = i$, and: $m_i =$ '$\mathcal{SHIQ}$', x has ancestors x', y, y' such that x is a direct successor of x' and y a direct successor of y', and $L_i(x) = L_i(y)$, and $L_i(x') = L_i(y')$, and $L(< x', x >) = L(< y', y >)$, **or** $m_i =$ '$\mathcal{SHOQ}$' and $L_i(x) \subseteq L_i(y)$, **or** $m_i =$ '$\mathcal{SHIO}$' and $L_i(x) = L_i(y)$.

An i-node is indirectly blocked iff one of its ancestors is blocked. For an i-node x, $L_i(x)$ contains a *clash* iff, either of the following conditions holds:

- $\{A, \neg A\} \subseteq L_i(x)$
- For some i-concept C, with $m_i \in \{\text{`}\mathcal{SHIQ}', \text{`}\mathcal{SHOQ}'\}$, some ij-Property S, with $i = j$, and some $n \in \aleph$, $(\leq nS^{(j)}.C) \in L_i(x)$ and there are $(n+1)$ $S^{(j)}$-neighbors $y_0, ..., y_n$ of x such that $C \in L_i(y_k)$ and $y_k \neq_i y_l$, for all $0 \leq k < l \leq n$
- For some $\{o\} \in L_i(x), x \neq_i x_o$

The graph $\mathcal{G}$ is *clash-free* iff none of its nodes contains a clash and it is *complete* iff none of the expansion rules is applicable.

For an input X, Σ (X an i-concept), the algorithm starts with n nodes x_j, $j = 1, ..., n$ with x_j a j-node x with $L_j(x) = \{X\}$ if $i = j$ and $L_j(x) = \emptyset$ if $i \neq j$. The algorithm also creates $l = \sum_{i=1}^{n} |V_{I_i}^{m_i}|$ nodes $x_{o_1}, ..., x_{o_l}$ with $L_i(x_{o_j}) = \{\{o_j\}\}, \forall o_j \in V_{I_i}^{m_i}$. Finally, the $\neq_i$ relations are initialized to the empty relation.

In DLs , transitivity is typically handled by using the $\rightarrow_{\forall^+}$ rule (Horrocks, Sattler, & Tobies 2000), which propagates universal restrictions across transitive edges. However, such a rule fails when the transitive relation relates different domains and is replaced by the $\rightarrow TRANS$ rule in Figure 1. Termination, soundness and completeness are a consequence of the following lemma:

Lemma 2 *Let* $\Sigma = (\mathcal{K}, \Re)$ *be a combined knowledge base the* $\mathcal{E}$*-connection* $\mathcal{C}^{\epsilon}_{\mathcal{HQ}^+}(\mathcal{SHIQ}, \mathcal{SHOQ}, \mathcal{SHIO})$ *and let* X *be an* i*-concept in the language of the* $\mathcal{E}$*-connection. Then:*

1. *The algorithm terminates when applied to* X *and* Σ*, and runs in the worst-case in 2NExpTime w.r.t. the sizes of* X *and* Σ
2. *The rules can be applied such that they generate a clash-free and complete graph* $\mathcal{G}$ *iff X is satisfiable w.r.t.* Σ

Note that the tableau algorithms for the logics $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHIO}$ also run in 2NExpTime in the worst-case (see for example (Horrocks, Sattler, & Tobies 2000)), and hence reasoning with the $\mathcal{E}$-Connections presented in this paper is not harder than reasoning with the component DLs themselves.

The Language $\mathcal{C}^{\epsilon}_{\mathcal{IH}^+}(\mathcal{SHIQ}, \mathcal{SHOQ}, \mathcal{SHIO})$

The language $\mathcal{C}^{\epsilon}_{\mathcal{IH}^+}(\mathcal{SHIQ}, \mathcal{SHOQ}, \mathcal{SHIO})$ allows the use of inverses of ij-properties relating different domains and disallows the use of number restrictions on those properties. The set of ij-properties for $i \neq j$ is the set $\epsilon_{ij}^{m_i} \cup \{Inv(P_{(ji)}) | P \in \epsilon_{ji}\}$ and, if Z is a j-concept, then $\geq nP^{(j)}.Z$ and $\leq nP^{(j)}.Z$, for P simple, are valid i-concepts iff $i = j$ and $m_i \in \{\text{`}\mathcal{SHIQ}', \text{`}\mathcal{SHOQ}'\}$.

The definition of a combined tableau is analogous to Definition 2. With respect to the tableau algorithm, the only change w.r.t. the previous section is the need of a slightly more sophisticated blocking technique. If an i-node x is a $P^{(i)}$-successor of a j-node x', for $i \neq j$ subset blocking does

[1] http://www.mindswap.org/2004/multipleOnt/papers/Extension.pdf

[2] An edge is added **either** by a generating rule or by the $\rightarrow TRANS$ rule

-$\longrightarrow \sqcap$ **rule**: If $C_1 \sqcap C_2 \in L_i(x)$, x is not blocked, and $\{C_1, C_2\} \cap L_i(x) = \emptyset$, **then** $L_i(x) \leftarrow L_i(x) \cup \{C_1, C_2\}$

- $\rightarrow \sqcup$ **rule**: If $C_1 \sqcup C_2 \in L_i(x)$, x is not blocked, and $\{C_1, C_2\} \cap L_i(x) = \emptyset$, **then** $L_i(x) \leftarrow L_i(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$

-$\longrightarrow CE$ **rule**: If $C_{\mathcal{K}_i} \notin L_i(x)$, **then** $L_i(x) \leftarrow L_i(x) \cup \{C_{\mathcal{K}_i}\}$

-$\longrightarrow TRANS$ **rule**: If x is an i-node and it has a P-ancestor y such that x, y are connected by a chain $y, z_1, ..., z_p, x$ s.t. P is transitive for each of $(y, z_1), (z_1, z_2), ..., (z_{p-1}, z_p), (z_p, x)$ within a single axiom, then add to the graph the edge $< y, x >$, with $L(< y, x >) = \{P^{(i)}\}$ if it didn't exist before.

- $\rightarrow \forall$ **rule**: **If** $\forall P^{(j)}.Z \in L_i(x)$, x is not indirectly blocked and x has a $P^{(j)}$-neighbor y with $Z \notin L_j(y)$, **then** $L_j(y) \rightarrow L_j(y) \cup \{Z\}$

-$\longrightarrow \exists$ **rule**: **If** $\exists P^{(j)}.Z \in L_i(x)$, x is not *blocked* and x has no $P^{(j)}$-neighbor y with $\{Z\} \in L_j(y)$ **then** create a new j-node y with $L(< x, y >) = P^{(j)}$ and $L_j(y) = \{Z\}$

-$\longrightarrow \geq$ **rule**: **If** $\geq nS^{(j)}.C \in L_i(x)$, x is not blocked and there are no n $S^{(j)}$-neighbors $y_1, ..., y_n$ of x with $C \in L_j(y_k)$, and $y_k \neq_j y_l$, for $1 \leq k < l \leq n$, **then** create n new j-nodes $y_1, ..., y_n$ with $L(< x, y >) = \{S^{(j)}\}$ and $y_k \neq_j y_l$, for $1 \leq k < l \leq n$, $L_j(y_k) = \{Z\}$

$\rightarrow \leq$ **rule**: **If** $\leq nS^{(j)}.Z \in L_i(x)$, x is not indirectly blocked and has $n+1$ $S^{(j)}$-neighbors $y_0, ..., y_n$ with $Z \in L_j(y_k)$, for each $0 \leq k \leq n$, and there exist $k \neq l$ s.t. not $y_k \neq_j y_l$, for $1 \leq k < l \leq n$, and if one of y_k,y_l is distinguished, it is y_k, **then**:

1)$L_j(y_k) = L_j(y_k) \cup L_j(y_l)$

2)Add $y \neq_j y_k$ for each y s.t. $y \neq_j y_l$

3)$L(< x, y_k >) = L(< x, y_k >) \cup L(< x, y_l >)$

4)Remove y_l and all the edges leading to y_l

-$\longrightarrow$ *choose* **rule**: **If** $\{\geq nS^{(j)}.Z, \leq nS^{(j)}.Z\} \cap L_i(x) \neq \emptyset$, x is not blocked and y is a $S^{(j)}$-neighbor of x, and $\{Z, \sim Z\} \cap L_j(y) = \emptyset$, **then** $L_j(y) = L_j(y) \cup \{X\}$, for some $X \in \{Z, \sim Z\}$

-$\longrightarrow O$ **rule**: If $\{o\} \in L_i(x)$, x is neither blocked, nor distinguished, and not $x \neq_i x_o$, then for z distinguished with $\{o\} \in L_i(z)$, do:

1)$L_i(z) = L_i(z) \cup L_i(x)$

2)If x has a predecessor x', then $L_i(x') = L_i(x') \cup \{\uparrow (P^{(i)}, \{o\}) | P^{(i)} \in L(< x', x >))\}$

4) Add $y \neq_i z$ for each y with $y \neq_i x$ and remove x and all edges leading to x

Figure 1: Expansion Rules

not suffice anymore due to the possible presence of inverses. In such case, x is directly blocked by an i-node y if y is an ancestor of x such that $L_i(x) = L_i(y)$.

We have not used in this paper the language $\mathcal{C}^{\epsilon}_{\mathcal{IHQ}^+}(\mathcal{SHIQ}, \mathcal{SHOQ}, \mathcal{SHIO})$, since in the presence of inverses and number restrictions on links and nominals in any of the component logics, the separation between domains is broken, since nominals can be exported from one component to another (Kutz *et al.* 2004).

Conclusion and Future Work

$\mathcal{E}$-Connections provide a *well-founded logical framework* for combining logics and KBs, which offers computational guarantees and modeling guidelines. However, when first introduced in (Kutz *et al.* 2004), some important issues were overlooked. In this paper, we have presented an extension of the $\mathcal{E}$-Connections formalism that solves some important limitations of the original framework. We have provided a tableau-based decision procedure for two extended $\mathcal{E}$-Connection languages involving the logics $\mathcal{SHIQ}, \mathcal{SHOQ}$ and $\mathcal{SHIO}$, which are at the basis of OWL (Patel-Schneider, Hayes, & Horrocks 2004).

In the future, we aim to explore up to what extent our extension can be generalized to Abstract Description Systems, which involve many families of logical formalisms other than expressive DLs. Finally, we are planning to implement the decision procedures presented in this paper as an extension of our DL reasoner Pellet, which already provides support for $\mathcal{E}$-Connections (Cuenca-Grau, Parsia, & Sirin 2004) (Cuenca-Grau, Parsia, & Sirin 2005).

References

Cuenca-Grau, B.; Parsia, B.; and Sirin, E. 2004. Working with multiple ontologies on the semantic web. In *Proc. of the 3thrd International Semantic Web Conference (ISWC 2004)*, 620–634. Springer.

Cuenca-Grau, B.; Parsia, B.; and Sirin, E. 2005. Combining OWL ontologies using $\mathcal{E}$-Connections. *Journal of Web Semantics*. Forthcoming.

Horrocks, I., and Sattler, U. 2001. Ontology reasoning in the $\mathcal{SHOQ}(\mathcal{D})$ description logic. In *Proc. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, 199–204. Morgan Kaufmann.

Horrocks, I.; Sattler, U.; and Tobies, S. 2000. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL* 8(3):239–263.

Kutz, O.; Lutz, C.; Wolter, F.; and Zakharyaschev, M. 2004. $\mathcal{E}$-Connections of abstract description systems. *Artificial Intelligence 156(1):1-73*.

Patel-Schneider, P.; Hayes, P.; and Horrocks, I. 2004. Web ontology language (OWL) abstract syntax and semantics. *W3C Recommendation*.

Sattler, U. 1996. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in LNAI. Springer.