Working with Multiple Ontologies on the Semantic Web

Bernardo Cuenca Grau[1,2], Bijan Parsia[1], and Evren Sirin[1]

[1] Maryland Information and Network Dynamics Laboratory, USA
[2] Department of Computer Science, University of Valencia, Spain

Abstract. The standardization of the second generation Web Ontology Language, OWL, leaves a crucial issue for Web-based ontologies unsatisfactorily resolved: how to represent and reason with multiple distinct, but linked, ontologies. OWL provides the owl:imports construct which, roughly, allows Web ontologies to include other Web ontologies, but only by merging all the linked ontologies into a single logical "space." Recent work on multidimensional logics, fusions and other combinations of modal logics, distributed and contextual logics, and the like have tried to find formalisms wherein knowledge bases (and their logic) are kept more distinct but yet affect each other. These formalisms have various degrees of robustness in their computational complexity, their modularity, their expressivity, and their intuitiveness to modelers. In this paper, we explore a family of such formalisms, grounded in $\mathcal{E}$-connections as extensions to OWL, with emphasis on a novel sub-formalism that seems very straightforward to implement on existing tableau OWL reasoners, as witnessed by our implementation of this formalism in the OWL reasoner Pellet. We discuss how to integrate those formalisms into OWL, as well as some of the issues that modelers have to face when using such formalisms in the context of a large number of heterogeneous, independently developed, richly interconnected ontologies that we expect to be the norm on the Semantic Web.

1 Introduction

Combining ontologies in a controlled and scalable way is crucial for the success of the Semantic Web. However, at the current stage, the means provided by the Web Ontology Language (OWL) for such a purpose are clearly insufficient. OWL provides a construct, 'owl:imports'[1], that allows to include by reference in a knowledge base the axioms contained in another ontology, presumably retrievable from the Web and identified by a URI. However, the functionality provided by owl:imports is unsatisfactory for a number of reasons.

First, it does not support information hiding or filtering, which means that everything in (the transitive closure of the imports of) the imported ontologies gets into the original ontology. Thus, it relies on developer discipline to avoid

[1] Throughout, we abbreviate the URI, http://www.w3.org/TR/2004/REC-owl-semantics-20040210/#*owl_imports*, with the pseudo-qname 'owl:imports'.

S.A. McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp. 620–634, 2004.

unmanageable blow up of ontology size. Indeed, as ontology reuse via owl:imports increases, so does the chance than an innocuous owl:imports will bring into the local ontology a large chunk of the Semantic Web. Second, none of the imported axioms or facts retain their context. While it is possible to track down the originator(s) of some assertions by inspection of the imported ontology, OWL reasoning does not take such context into account. Finally, the use of a "foreign" URI reference without a corresponding owl:imports brings *nothing* from any of the foreign owner's ontologies

Hence, with owl:imports, OWL can let in either all the axioms of a foreign ontology, or none. The primary distinction, in practical terms, between using owl:imports and directly adding all the assertions in the imported ontology is that with owl:imports, the modeler can break an ontology into different documents. This is a very important distinction, but it is not quite the distinction many hope for.

2 Logical Formalisms for Combining Knowledge Bases

Fortunately, there is a growing body of work about combining multiple ontologies in ways that strike a middle ground between importing all and nothing. In this paper, we consider two formalisms, which are natural extensions of OWL-DL: Distributed Description Logics (DDLs) [2], and $\mathcal{E}$-connections [4].

2.1 Distributed Description Logics

Distributed Description Logics (DDL) [2] is a formalism for combining different DL knowledge bases in a loosely coupled information system. The idea of the combination is to preserve the "identity" and independence of each local ontology. The coupling is established by allowing a new set of inter-ontology axioms, called bridge rules. A bridge rule is an expression of one of the following forms[2]

$$C_i \xrightarrow{\sqsubseteq} C_j; \qquad ; C_i \xrightarrow{\sqsupseteq} C_j \; ; \qquad a_i \rightarrow b_j$$

Where, C_i, C_j are classes and a_i,b_j individuals in the ontologies O_i,O_j respectively.

From the modeling point of view, bridge rules have been conceived for establishing directional ("view dependent") subsumption relationships between classes and correspondences between individuals in different ontologies. The motivation for bridge rules, hence, covers a wide variety of intuitive modeling scenarios on the Semantic Web. For example, suppose that in ontology O_1 we have a class "Beer" defined as a subclass of "GermanProduct" and which doesn't exist in ontology O_2. However, in ontology O_2 there is a class "Drink", but there is no class modeling the concept of "Beer". Both ontologies could be linked by

[2] We will consider bridge rules without complete individual correspondences along the paper

using a bridge rule stating that "Beer" in ontology O_1 is a subclass of "Drink" in O_2 .

These kind of modeling scenarios are appealing, because they nicely fit in the vision of the Semantic Web in which the notion of a universal upper ontology is abandoned by the notion of a web of distributed, linked and independently developed ontologies. Researchers in the Semantic Web community are starting to realize the importance of extending OWL with a suitable formalism that provides inter-ontology mappings with a precise logical semantics. A first attempt in this direction resulted in C-OWL [3], a syntax and semantic extension of OWL that adds the DDL formalism to the language.

However, C-OWL, as presented in [3] has some difficulties. First, no reasoning support is provided for the language. Extending the existing tableau reasoners to deal with such an extension is certainly a crucial issue. Second, some expressive features that are beyond DDLs (and also beyond $\mathcal{E}$-connections), like the inclusion of inter-ontology role subsumption statements, are included in the language without the required discussion. This kind of expressive power could even make the reasoning undecidable. Finally, from the modeling perspective, no distinction is made between the idea of partially importing concepts from a foreign KB and the idea of linking ontologies through the use of inter-ontology subsumption relationships. It is unclear whether the semantics of DDLs capture that idea of linking, as we will discuss in section 5.

2.2 $\mathcal{E}$-Connections

The $\mathcal{E}$-connections technique [4] is a method for combining logical formalisms that are expressible in the Abstract Description System (ADS) framework. ADSs are a generalization of description, modal and epistemic logics and many logics of time and space (see [1] for a detailed discussion).

The advantages of this technique are twofold: On one hand, $\mathcal{E}$-connections provide a quite expressive way for combining knowledge bases written in a wide variety of logical languages, while, on the other hand, the coupling between the combined logical formalisms is loose enough for ensuring the decidability of the combined formalism. Hence, $\mathcal{E}$-connections provide a trade-off between the tightness of the coupling between the component logical languages and the computational robustness of the combination.

The $\mathcal{E}$-connection method can be applied, for example, to the following setting. Assume that we have n disjoint domains $D_1, ..., D_n$ and two languages l_1 and l_2 for talking about them, and suppose that l_1 and l_2 are two propositionally closed description logics (two extensions of ALC) with decidable satisfiability (and hence subsumption) problem and such that they can be expressed as an Abstract Description System. Assume we want to define a combined formalism $C(l_1, l_2)$ which talks about the domains and also about the relationships among them. First, we define countable and disjoint sets ϵ_{ij} with $i, j = 1, ..., n; i \neq j$ of link names. A link $E \in \epsilon_{ij}$ can be seen as a property that relates elements from the domain D_i to elements of the domain D_j, as opposed to object properties, which relate elements of the same interpretation domain.

For example, let K_1 and K_2 be two knowledge bases written in l_1 and l_2 respectively. The knowledge base K_1 deals with people, while K_2 describes the domain of Pets; then we can define the links $Owns \in \epsilon_{12}$, and $LovesToPlayWith \in \epsilon_{21}$. Intuitively, the link $Owns$ represent the relationships of a person to own a pet, while the link $lovesToPlayWith$ would represent the relation of pets liking to play with people. In its basic form, the syntax of the combination is specified by extending the syntax of l_1 and l_2 with two new constructors: existential and value restrictions on links. For example, in our ontologies K_1, K_2 it is possible to define in K_1 a "DogOwner" as a person who owns at least one dog and in K_2 an "unfriendly pet" as a pet that doesn't like to play with people:

$$DogOwner \equiv Person \sqcap \exists Owns.Dog$$
$$UnfriendlyPet \equiv Pet \sqcap \forall LovesToPlayWith.(\neg Person)$$

Finally, we can also use inverses on the links and define an "Unhappy Cat" in K_2 as a cat that is *owned by* a "Dog Owner":

$$UnhappyCat \equiv Cat \sqcap \exists Owns^{-}(DogOwner)$$

Given n ontologies, the semantics of the combined KB, written in $C(l_1, l_2)$, is given by a combined interpretation $I = (\{I_i\}_{i=1}^{n} \epsilon_{ij}^{I})$, where $I_i = (W_i, .^{I_i})$ is an interpretation and $W_i \cap W_j = \emptyset, \forall i \neq j$. A link $E \in \epsilon_{ij}$ is interpreted as a cross-domain relation $E^I \subseteq W_i \times W_j$, and its inverse E^- as the relation $(E^-)^I = \{(a,b) \in W_j \times W_i | (b,a) \in E^I\}$. Finally, the expressions $\exists E.C$, and $\forall E.C$, where $E \in \epsilon_{ij}$ and C a concept in the jth KB, are interpreted as subsets of W_i, as follows:

$$(\exists E.C)^I = \{x \in W_i | \exists y \in W_j, (x,y) \in E^I, y \in C^I\}$$
$$(\forall E.C)^I = \{x \in W_i | \forall y \in W_j, if\ (x,y) \in E^I, \rightarrow y \in C^I\}$$

$\mathcal{E}$-connections are strictly more expressive than DDLs (without individual correspondences), and in [4] it is shown how DDLs can be translated into axioms in an $\mathcal{E}$-connection.

3 Integrating $\mathcal{E}$-Connections into OWL

In this section, we define a syntax and semantic extension of OWL that integrates the $\mathcal{E}$-connections formalism into the language in a compact and natural way. The extension is based on the definition of a new set of properties, called links, that stand for the inter-ontology relations in the $\mathcal{E}$-connections framework. The OWL-DL[3] language is then enriched with a new set of constructors that basically allow to define new classes by placing restrictions on the link properties.

[3] We are considering OWL-DL/OWL-Lite along the paper because OWL-Full is beyond the ADS framework, and hence $\mathcal{E}$-connections cannot be used to combine OWL-Full with OWL-DL KBs

3.1 Abstract Syntax

The syntax presented in this section is an extension of the normative OWL abstract syntax, as described in the OWL abstract syntax and semantics recommendation [5]. We use the same Extended BNF syntax as in the normative document. A new set of properties, called link properties, are named using a URI, and hence a *linkID* is defined to be a $URIreference$

In order to ensure the separation of vocabularies, a URI-reference cannot be both a linkID, an individual-valued property ID, a datatype property ID, an annotation property ID or an ontology property ID. Intuitively, while individual-valued properties relate individuals to other individuals in the same ontology, while link properties relate individuals corresponding to different interpretation domains. Thus, link properties act as a bridge between different ontologies, which remain separate, and keep their own identity.

Our proposal extends OWL-DL with a new kind of property axiom, by adding the following production rule:

axiom::= 'Link('**linkID**['Deprecated'] { **annotation** }
['inverseOf(' linkID ')']
{ ' domain (' **Description**')' }
{ ' range ('**Description**')' }
'ForeignOntology (' **OntologyID**')

Informally, link properties are used to bring or import class descriptions from a foreign OWL ontology to the local context, without having to import the whole external knowledge base. The addition of inverses in link properties is a distinguishing feature of $\mathcal{E}$-connections, which has important consequences, both conceptually and in practice.

Link properties can be used to define new classes by means of restrictions on them. Intuitively, a restriction on a link property is applied to a concept in a foreign ontology, and returns a concept defined over the local interpretation domain.

restriction::= 'Restriction('**linkID**
linkRestrictionComponent
{ **linkRestrictionComponent** } ')'

linkRestrictionComponent::= 'allValuesFrom('**Description**)'
| 'someValuesFrom('**Description**)'

The intuition is that the classes specified in link restriction properties are evaluated according to the semantics of the foreign ontology specified in the definition of the link property. Finally, the set of facts is defined by extending the set of facts in OWL in order to provide the means for instantiating link properties. This is achieved by simply adding to the OWL abstract syntax a new alternative to the non-terminal "value"

value ::= 'value(' linkID individualID ')'

Essentially, link properties are similar to Datatype properties. In OWL DL, datatypes constitute a domain disjoint from the domain of classes. Similarly, classes defined in foreign ontologies belong to a domain separate from the domain of classes defined in the importing ontology. The reasoning about datatypes is done by a so-called "Datatype Oracle" and this process is a black-box operation for the rest of the reasoning procedure. Analogously, reasoning about link properties are done with respect to the foreign ontology in an isolated process.

3.2 Direct Model-Theoretic Semantics

The semantics starts with the notion of vocabulary.

Definition 1. *(Combined OWL Vocabulary)*
A combined OWL vocabulary V consists of a tuple $V = \{V_1, ..., V_n, V_{link}\}$, where $V_1, ..., V_n$ is a set of OWL vocabularies, as defined in [5], and V_{link} is a set whose elements are triples of the form:

$$(O_i, O_j, L_k),\ i, j = 1, ..., n, i \neq j$$

Where O_j, O_i are ontologyIDs with $i \neq j$, L_k a linkID (a URI), and where none of the linkIDs are contained in any of the $V_i, i = 1..., n$

Definition 2. *(Combined OWL interpretation)*
A combined OWL interpretation with vocabulary $V = \{V_1, ..., V_n, V_{link}\}$ is a tuple of the form:

$$I = (\{I_i\}_{i=1}^{n}, \epsilon_{ij} = (O_i, O_j, L_k)^I), i, j = 1, ..., n; i \neq j;$$

Where $I_i, i = 1, ..., n$ is an abstract OWL interpretation with vocabulary V_i, and with respect to a datatype map D_i, and domain W_i, and $(O_i, O_j, L_k)^I = W_i \times W_j$, for $(O_i, O_j, L_k) \in V_{link}$.

Class and property axioms are interpreted as in OWL-DL, with the remark that now the concepts built using the new constructors can appear in class axioms and the semantics applied to them must be the semantics of table 1.

Definition 3. *($\mathcal{E}-$Connection of OWL ontologies)*
A combined OWL ontology is a combined KB $\Sigma = \{O_1, ..., O_n\}$ written in the extended abstract syntax, such that it is closed under 'owl:imports' and for every ontology in Σ, all the link properties it contains refer to a foreign ontology which is also contained in Σ.

Finally, we say that a combined interpretation I satisfies Σ, iff it satisfies all the axioms of all the ontologies in it.

Table 1. Semantics

Abstract Syntax	EC Syntax	Semantics
$Link(L_k)$ defined in O_i $ForeignOntology(O_j)$	L_k^{ij}	$(L_k^{ij})^I \subseteq W_i \times W_j$
inverseOf((E_m))	$L_k^{ij} \equiv (E_m^{ji})^-$	$((L_k)ij)^I = ((E_m^{ji})^I)^-$
domain(C)		$(L_k^{ij})^I \subseteq C^I \times W_j$
range(D)		$(L_k^{ij})^I \subseteq W_i \times D^I$
restriction(L_k $someValuesFrom(C)$)	$\exists L_k^{ij}(C)$	$(\exists L_k^{ij}(C))^I = \{x \in W_i \vert \exists y \in W_j, [x,y] \in (L_k^{ij})^I \ and \ y \in C^I\}$
restriction(L_k $allValuesFrom(C)$)	$\forall L_k^{ij}(C)$	$(\forall L_k^{ij}(C))^I = \{x \in W_i \vert \forall y \in W_j, \ if \ [x,y] \in (L_k^{ij})^I \rightarrow y \in C^I\}$
value (t L_k o)	$(t,o) : L_k^{ij}$	$(t^I, o^I) : (L_k^{ij})^I$

4 Reasoning with $\mathcal{E}$-Connections on the Semantic Web

Finding efficient and easy to implement algorithms for reasoning on $\mathcal{E}$-connections is a challenging task. Two possible ways of simplifying the problem are, on one hand, defining subformalisms of $\mathcal{E}$-connections which still preserve most of the expressive power of $\mathcal{E}$-connections, and, on the other hand, restricting to particular ADS of special practical relevance.

In this paper we do both. We define Perspectival $\mathcal{E}$-connections (PECs), an expressive sublanguage of $\mathcal{E}$-connections, and we restrict ourselves to combinations of OWL-Lite (SHIF(D)) ontologies without ABoxes. We will use the notation $C^{\epsilon}(SHIF(D))$, for the PEC language that enables the combination of OWL-Lite ontologies.

The main motivation for introducing PECs is that they can be implemented in a quite straightforward way by extending the existing tableaux-based DL reasoners, at least for logics that do not include nominals (hence the restriction to SHIF(D)/OWL-Lite).

4.1 Perspectival $\mathcal{E}$-Connections

Perspectival $\mathcal{E}$-connections (PECs) are obtained from $\mathcal{E}$-connections by disallowing the use of inverses on link properties. This implies that links cannot be "navigated" in both directions anymore. In the abstract syntax, PECs would be obviously defined by suppressing the inverse clause in the definition of link properties. In our example of persons and pets, we would still be able to define the concept of a "dog owner" and "unhappy pet owner" in K_1, but not the concept of "unhappy cat" in K_2.

As a consequence of the restrictions imposed to the formalism, PECs are strictly less expressive than $\mathcal{E}$-connections, but remain strictly more expressive than DDLs.

The syntax and semantics of the combined language $C^{\epsilon}(SHIF(D))$ is defined as follows:

Definition 4. *Let $C_i, R_i, \delta_i, \epsilon_{ij}$ be disjoint sets, for $i, j = 1...n; i \neq j$, where the $C_i, R_i, \delta_i, \epsilon_{ij}$ are respectively sets of concept, role, datatype properties, and link*

Table 2. Semantics of i-concepts

Syntax	Semantics
R^-	$(R^-)^I = \{(a,b)\|(b,a) \in R^I\}$
$\neg C$	$(\neg C)^I = W_i - C^I$
$C \sqcap D$	$(C \sqcap D)^I = C^I \cap D^I$
$C \sqcup D$	$(C \sqcup D)^I = C^I \cup D^I$
$\exists R.C$	$(\exists R.C)^I = \{x \in W_i \| \exists y \in W_i, (x,y) \in R^I, y \in C^I\}$
$\forall R.C$	$(\forall R.C)^I = \{x \in W_i \| \forall y \in W_i, (x,y) \in R^I \rightarrow y \in C^I\}$
$\leq 1S$	$(\leq 1S)^I = \{x \in W_i \| \forall y,z, if\ (x,y) \in S^I \wedge (x,z) \in S^I \rightarrow y = z\}$
$\geq 2S$	$(\geq 2S)^I = \{x \in W_i \| \exists y,z : (x,y) \in S^I, (x,z) \in S^I, \wedge y \neq z\}$
$\exists U.d$	$(\exists U.d)^I = \{x \in W_i \| \exists y \in \Delta_D, (x,y) \in U^I, y \in d^D\}$
$\forall U.d$	$(\forall U.d)^I = \{x \in W_i \| \forall y \in \Delta_D, (x,y) \in U^I \rightarrow y \in d^D\}$
$\exists G.Z$	$(\exists G.Z)^I = \{x \in W_i \| \exists y \in W_j, (x,y) \in G^I, y \in Z^I\}$
$(\forall G.Z)$	$(\forall G.Z)^I = \{x \in W_i \| \forall y \in W_j, if\ (x,y) \in G^I, \rightarrow y \in Z^I\}$

names. The set of i-roles is the set $R_i \cup \{R^- | R \in R_i\}$. *We will use the notation* $Inv(R) = S$ *if* $R = S^-$.

For R, S *i-roles, a role inclusion axiom is an assertion of the form* $R \sqsubseteq S$. *An i-role axiom is either a role inclusion axiom, an assertion of the form* $Inv(R) = S$ *or a transitivity axiom* $Trans(R)$ *, with* $R \in R_i$. *An i-role box* $\Re_i$ *is a finite set of i-role axioms.The combined role box is given by the tuple* $\Re = (\Re_1, ..., \Re_n)$. *An i-role* R *is called simple if for* $\sqsubseteq^*$ *the transitive reflexive closure of* $\sqsubseteq$ *on* $\Re_i$, *and for each i-role* S, $S \sqsubseteq^* R$ *implies* $Trans(R) \notin \Re_i$. *The set of ij-links is the set of atomic links* ϵ_{ij}, *and the set of i-datatype properties is the set* δ_i.

Finally, the set of i-concepts in $C^\epsilon(SHIF(D))$ *is recursively defined, for* $i = 1, ..., n$ *as the smallest set such that each concept name* $A \in C_i$, *is an i-concept and, for* R *an i-role,* U *an i-datatype property,* S *a simple i-role,* $G \in \epsilon_{ij}$ C, D *i-concepts, and* Z *a* j*-concept, the following constructions are also valid i-concepts:*

- $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $(\leq 1S)$, $(\geq 2S)$, $\forall U.d$, $\exists U.d$
- $\exists G.Z$, $\forall G.Z$

A combined TBox is defined as $K = (K_1, ..., K_n)$, *where* K_i *is a set of assertions of the form* $C \sqsubseteq D$, *with both* C, D *i-concepts. The semantics is given by means of an interpretation* $I = (\{I_i\}\epsilon_{ij}^I)$, $i, j = 1, ..., n; i \neq j$ *where* $I_i = (W_i, .^{I_i})$, *with* $W_i \cap W_j = \emptyset$. *The interpretation function maps every atomic concept* $A \in N_{C_i}$ *to* $A^I \subseteq W_i$, *every role* $R \in N_{R_i}$ *to* $R^I \subseteq W_i \times W_i$, *every link* $G \in \epsilon_{ij}$ *to* $G^I \subseteq W_i \times W_j$, *every i-datatype property* U *to* $U^I \subseteq W_i \times \Delta_D$, *and every datatype* d *to* $d^D \subseteq \Delta_D$. *Table 1 shows the extension of the interpretation function to complex descriptions. In the table,* C, D *are i-concepts,* R *is an i-role,* S *is an simple i-role,* U *is an i-datatype property,* Z *is a* j*-concept,* $G \in \epsilon_{ij}$, *and* d *is a datatype.*

An interpretation satisfies an i-role inclusion axiom $R \sqsubseteq S$, *if* $R^{I_i} \subseteq S^{I_i}$, *where* R, S *are both i-roles and the axiom is in* $\Re_i$, *and satisfies a transitivity*

axiom $Trans(R)$, *iff* $R^{I_i} = (R^{I_i})^+$. *An interpretation satisfies the combined role box* $\Re$ *iff it satisfies all the axioms in each* $\Re_i$. *If* C, D *are i-concepts, an interpretation satisfies the axiom* $C \sqsubseteq D$ *if* $C^I \subseteq D^I$, *and satisfies the combined TBox* K *iff it satisfies all the axioms in each* K_i. *A combined knowledge base is defined as* $\Sigma = (T, \Re)$, *where* K *is a combined TBox and* $\Re$ *a combined role box. An interpretation satisfies* Σ *if it satisfies both* K *and* $\Re$. *An i-concept* C *is satisfiable wrt* Σ *iff there is an interpretation* I *such that it satisfies* Σ, *and* $C^I \neq \emptyset$. *Such an interpretation is called a model of* C *w.r.t.* Σ. *An i-concept* C *is subsumed by another i-concept* D *wrt* Σ *iff* $C^I \subseteq D^I$, *for each interpretation satisfying* Σ.

We can transform each component K_i *of a combined TBox* K *into a single equivalent concept equation* $\top_i \equiv C_{K_i}$, *where:*

$$C_{K_i} = \sqcap_{C_j^i \sqsubseteq D_j^i \in K_i} (\neg C_j^i \sqcup D_j^i)$$

An interpretation I *will satisfy the TBox* K *iff* $\top_i = C_{K_i}^I$, *for* $i = 1, ..., n$.

5 A Combined Tableau for $C^{\epsilon}(\boldsymbol{SHIF}(\boldsymbol{D}))$

We will assume concepts written in NNF. To transform an i-concept into NNF, we can push negation inwards using De Morgan's laws and the following equivalences:

For R an i-role, U an i-datatype role S an i-simple role, $E \in \epsilon_{ij}, i, j = 1, ..., n; i \neq j$, C an i-concept, Z a j-concept and d a datatype:

$$\neg\exists R.C \equiv \forall R.\neg C \qquad \neg\forall R.C \equiv \exists R.\neg C \qquad \neg \geq 2S \equiv \leq 1S \qquad \neg \leq 1S \equiv \geq 2S$$
$$\neg\exists U.d \equiv \forall U.\neg d \qquad \neg\forall U.d \equiv \exists U.\neg d \qquad \neg\exists E.Z \equiv \forall E.\neg Z \qquad \neg\forall E.Z \equiv \exists D.\neg Z$$

We will use the notation $\sim C$ for the NNF of $\neg C$.

Let X be an i-concept, then the set $sub_i(X, \Re)$ of i-subconcepts of X w.r.t. the combined role box $\Re$ is defined as follows, for all $i, j = 1, ..., n; i \neq j$:

- If X is an atomic primitive i-concept or its negation, then $sub_i(X, \Re) = \{X\}$, $sub_j(X, \Re) = \emptyset$
- If X is of the form $\exists R.C$ where $\exists R.C$ is an i-concept, then $sub_i(X, \Re) = \{X\} \cup sub_i(C, \Re)$, $sub_j(X, \Re) = sub_j(C, \Re)$
- If X is of the form $\forall R.C$, where $\forall R.C$ is an i-concept then $sub_i(X, \Re) = \{X\} \cup sub_i(C, \Re) \bigcup_k \{\forall S_k.C\}$ for every $S_k \sqsubseteq^* R$, with $Trans(S_k)$ in $\Re_i$. Again, $sub_j(X, \Re) = sub_j(C, \Re)$
- 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 $sub_i(X) = \{X, \Re\} \cup sub_i(C_1, \Re) \cup sub_i(C_2, \Re)$, $sub_j(X, \Re) = sub_j(C_1, \Re) \cup sub_j(C_2, \Re)$
- If X is of the form $\leq nS$ or $\geq nS$, where S is a simple role, then $sub_i(X, \Re) = \{X\}$ and $sub_j(X, \Re) = \emptyset$
- If X is a concept of the form $\exists U.d$, or $\forall U.d$, where U is a datatype property and d a datatype, then $sub_i(X, \Re) = \{X\} \cup \{d\}$ and $sub_j(X, \Re) = \emptyset$

– If X is an i-concept of the form $\exists G.D$ or $\forall G.D$, where $G \in \epsilon_{ij}$, D a j-concept, then $sub_i(X, \Re) = \{X\} \cup sub_i(D, \Re)$, $sub_j(X, \Re) = sub_j(D, \Re)$

For X and a combined knowledge base Σ, we define $sub_i(X, \Sigma)$ as the union of $sub_i(X, \Re)$ and all the $sub_i(C_{T_k}, \Re), k = 1, ..., n$. From now on, we will call ρ_i and δ_i to the set of i-roles and i-datatype properties respectively occurring in X,Σ, and ϵ_{ij} will be the set of ij-links appearing in X, Σ

Definition 5. *(Combined Tableau)*

Let X be a valid i-concept (written in NNF), and Σ a combined KB. A ***combined tableau*** *T for X wrt Σ is defined as a tuple $T = (\{T_i\}, \epsilon_X)$, $i = 1, ..., n$ where:*

– $T_i = (O_i, L_i, \alpha_i, \mu_i)$, *with O_i a set of individuals, and:*
  - $L_i : O_i \rightarrow 2^{sub_i(X,\Sigma)}$
  - $\alpha_i : \rho_i \rightarrow 2^{O_i \times O_i}$
  - $\mu_i : \delta_i \rightarrow 2^{O_i \times \Delta_D}$
– $\epsilon_X : \epsilon_{ij} \rightarrow 2^{O_i \times O_j}$

In a combined tableau for an i-concept X, there must exist n individuals $o_j \in O_j$ such that $\{X, C_{K_j}\} \subseteq L_j(o_j)$ if $j = i$ and $\{C_{K_j}\} \subseteq L_j(o_j)$ otherwise, for all $j = 1, ..., n$.

Moreover, for all $i, j = 1, ..., n; i \neq j$, $s_i, t_i \in O_i, C, D \in sub_i(X, \Sigma)$, $Z \in sub_j(X, \Sigma)$, $R, S \in \rho_i$, $E \in \epsilon_{ij}$ $U \in \delta_i$, and d a datatype, the following conditions must hold:

1. *If $C \in L_i(s_i)$, then $\neg C \notin L_i(s_i)$*
2. *If $(C \sqcap D) \in L_i(s_i)$, then $C \in L_i(s_i)$, and $D \in L_i(s_i)$*
3. *If $(C \sqcup D) \in L_i(s_i)$, then $C \in L_i(s_i)$, or $D \in L_i(s_i)$*
4. *If $\forall R.C \in L_i(s_i)$, and $(s_i, t_i) \in \alpha_i(R)$, then $C \in L_i(t_i)$*
5. *If $\exists R.C \in L_i(s_i)$, then there is some $t_i \in O_i$ such that $(s_i, t_i) \in \alpha_i(R)$, and $C \in L_i(t_i)$*
6. *If $\forall R.C \in L_i(s_i)$, and $(s_i, t_i) \in \alpha_i(S)$, for some $S \sqsubseteq^* R$ with $Trans(S)$, then $\forall S.C \in L_i(t_i)$*
7. *If $(\leq 1R) \in L_i(s_i)$, and $(s_i, t_i) \in \alpha_i(R)$, and $(s_i, t'_i) \in \alpha_i(R)$, then $t_i = t'_i$*
8. *If $(\geq 2R) \in L_i(s_i)$, then there are some $t_i, t'_i \in O_i$, such that $(s_i, t_i) \in \alpha_i(R)$, $(s_i, t'_i) \in \alpha_i(R)$, and $t_i \neq t'_i$*
9. *If $(s_i, t_i) \in \alpha_i(R)$ and $R \sqsubseteq^* S$, then $(s_i, t_i) \in \alpha_i(S)$*
10. *$(s, t) \in \alpha_i(R)$ iff $(t, s) \in \alpha_i(Inv(R))$*
11. *If $\forall U.d \in L_i(s_i)$, and $(s_i, t_i) \in \mu_i(U)$, then $t_i \in d^D$*
12. *If $\exists U.d \in L_i(s_i)$, then there is some $t_i \in \Delta_D$ such that $(s_i, t_i) \in \mu_i(U)$, and $t_i \in d^D$*
13. *If $\forall E.Z \in L_i(s)$ where $s \in O_i$, and $(s, t) \in \epsilon_X(E)$, then $Z \in L_j(t)$, $t \in O_j$*
14. *If $\exists E.Z \in L_i(s)$, where $s \in O_i$, then there exists some $t \in O_j$ such that $(s, t) \in \epsilon_X(E)$, and $Z \in L_j(t)$*

Lemma 1. *An i-concept X in $C^{\epsilon}(SHIF(D))$ is satisfiable wrt a combined knowledge base Σ iff X has a combined tableau wrt Σ.*

5.1 Description of the Algorithm

We suggest a tableaux-based reasoning algorithm for determining the satisfiability of i-concepts (i=1,2) wrt a combined knowledge base Σ in the language $C^{\epsilon}(SHIF(D))$.

From the structure of a combined TBox it is easy to see that the combined TBox K cannot be internalized into a single concept. This means that we cannot directly use a concept satisfiability algorithm with empty TBox as a decision procedure for the reasoning problems in the combined TBox. We add a rule $\rightarrow CE$ that ensures that every i-node contains the concept C_{K_i}.

The algorithm works on a *finite combined completion*, which is a forest of $SHIF(D)$ completion trees. The labels of the i-nodes in the model are restricted to subsets of $sub_i(X, \Sigma)$.

The algorithm tries to build a combined tableau for the input concept. If it succeeds, it returns "satisfiable", while if it fails to build the tableau, it returns "unsatisfiable".

The algorithm generates n kinds of nodes, called i-nodes, and can generate n different types of trees, called i-trees. An i-tree is expanded using the $SHIF(D)$ rules, plus the additional rules and can contain i-nodes and j-nodes, $i \neq j$. The root node of an i-tree is always an i-node and the j-nodes of the i-trees are always leaves. The algorithm will keep track of all the generated i-trees, together with the dependencies between them. If the algorithm is building an i-tree T and during the process generates a new j-tree, the algorithm will continue expanding T, potentially generating new j-trees on the way, which are all processed after T is finished. A j-node g in an i-tree contains a clash iff the corresponding j-tree contains a clash when expanded. Otherwise, it is satisfiable and g is marked with the label "visited".

The algorithm adds three new expansion rules:

- $\rightarrow \exists_{link}$ **Rule**: If $\exists E.C$ is in $L_i(x)$ $(E \in \epsilon_{ij})$, x is not blocked and has no E-successor y with $L(x, y) = \{E\}$, and $\{C\} \in L_j(y)$, then create a new E-successor (a j-node) y of x with $L_j(y) = \{C\}$. A new j-tree is created with root y labeled with $L_j(y)$. The j-tree won't be expanded until no more rules apply in the i-tree.
- $\rightarrow \forall_{link}$ **Rule**:If $\forall E.C$ is in $L_i(x)$ $(E \in \epsilon_{ij})$, x is not blocked and there exists an E-successor y of x, such that $C \notin L_j(y)$, then $L_j(y) = L_j(y) \cup \{C\}$.
- $\rightarrow CE$ **Rule**: If $C_{K_i} \notin L_i(x)$, then $L_i(x) \leftarrow L_i(x) \cup \{C_{K_i}\}$

However, the algorithm as we have described it so far may not terminate. In order to ensure termination an extra blocking condition is required. Each time the algorithm starts expanding a new i-tree T with root g, created as a successor of a certain j-node, the algorithm checks if there exists a node x in a not yet completed i-tree such that $L_i(g) \subseteq L_i(x)$. In that case the algorithm blocks the root of the new tree, which returns "satisfiable". Given as an input an i-concept X and a combined KB Σ the algorithm creates an i-tree with a single i-node x_i, labeled $L_i(x_i) = \{X\}$, and a j-tree for each $j = 1, ..., n; j \neq i$, each with a single j-node labelled with the emptyset.

A tree is **complete** when some node in it contains a (SHIF(D)) clash, or when none of the rules is applicable in the i-nodes and all the j-nodes it contains are marked as "visited".

If for the input i-concept X, the expansion rules can be applied to all the trees in such a way that each initial tree yields to a complete, clash-free completion tree, then the algorithm returns "X is satisfiable", and "X is unsatisfiable" otherwise.

The algorithm presented in this section is a decision procedure for satisfiability and subsumption of i-concepts w.r.t. a combined knowledge base Σ, as a direct consequence of the following lemma:

Lemma 2. *Let $\Sigma = (K, \Re)$ a combined knowledge base in the PEC $C^{\epsilon}(SHIF(D))$ and let X be an i-concept. Then:*

1. *The algorithm terminates when applied to Σ and X*
2. *The rules can be applied such that they generate a clash-free and complete combined completion iff X is satisfiable w.r.t. Σ*

This technique shows that the generated i-trees in a combined completion are independent, in the sense that they do not affect each other, and hence the decision procedure we have presented has a "black box" property. A a slight modification of existing DL reasoners suffices for implementing the algorithm. However, this technique cannot be straightforwardly extended to basic $\mathcal{E}$-connections. If the algorithm is naively applied to the basic $\mathcal{E}$-connections case, then it happens to be unsound.

As an example, consider the concept in the pet ontology:

$$Man \sqcap \exists Owns(\forall Owns^{-}(\neg Man))$$

The concept is clearly unsatisfiable; however, the algorithm will return the wrong answer. The unsoundness is caused by the presence of inverses on the links, which breaks the black box property of the algorithm.

Nominals (presence of the oneOf constructor) also cause unsoundness if the algorithm is naively extend to PECs whose component logics contain nominals. For example, a concept of the form:

$$\exists E.(\{o\} \sqcap A) \sqcap \exists F.(\{o\} \sqcap \neg A)$$

Would be unsatisfiable. However, a straightforward extension of the algorithm would return the wrong answer[4]. Similar considerations do apply to the $C^{\epsilon}(SHIF(D))$ combination when ABoxes are included.

[4] We would like to thank the anonymous reviewer of the DL-2004 workshop that pointed out this example to us in an earlier presentation of some of this work

5.2 Implementation

We have implemented the algorithm in the OWL reasoner Pellet, a tableau based DL reasoner specifically developed to work with OWL ontologies. The reasoner also supports XML Schema datatypes and includes a datatype oracle for this purpose.

In Pellet, an OWL ontology is normally parsed into one KB that has the associated TBox/ABox components. The effect of importing an OWL ontology results in the merging of these two KBs. For PECs, the component ontologies should be kept separate. Hence, each ontology is loaded into a disjoint KB during the parsing process. A global ontology manager stores this set of ontologies along with the cached results of satisfiability results for the different KBs. Please note that we are considering combinations of an arbitrary number of ontologies.

The modification to the existing tableau algorithm implies marking each node with the ontology it belongs to. When a successor node is being created due to a link property, the new node is marked as belonging to the ontology which defined that link. The clash detection for the nodes that belong to foreign ontologies is performed by the ontology manager.

This ensures termination and also provides some caching of the satisfiability results. For each ontology two different caches are stored: one for satisfiable concepts and the other for unsatisfiable concepts. After a satisfiability test succeeds, the labels of all the nodes in the resulting completion tree are added to the satisfiability cache of that ontology. In the case of an unsatisfiable concept, only the label of the root node is stored in the unsatisfiability cache. When a new satisfiability test is asked, these caches are first searched for the previous answers.

6 Difficulties When Modeling with C-OWL and DDLs

Consider the ontology Figure 1a. The ontology O defines two disjoint concepts $Flying$ and $NonFlying$, the concept $Bird$ and the concept $Penguin$. The axioms in O state that all birds fly and also that a penguin is a bird, but it doesn't fly. In this case, in which all the axioms are gathered in a single logical space, the concept $Penguin$ would be clearly unsatisfiable.

Now, imagine that we split the knowledge about the domain in two coupled ontologies, shown in Figure 1b. The ontology A states that the concepts $Flying$ and $NonFlying$ are disjoint and states that all birds fly. On the other hand, the ontology B defines the concept $Penguin$ and states, using DDL subsumption links, that a penguin doesn't fly and that a penguin is a bird. However, it is easy to see by direct application of the semantics that in this case the obvious (and relevant) contradiction is not detected, and $Penguin$ is satisfiable in the coupled system.

The reason for this is that what bridge rules actually do is to place restrictions on a link property, which is nothing but an "ordinary" $\mathcal{E}$-connection link. Intuitively, there's nothing contradictory in these bridge rules in the same way that

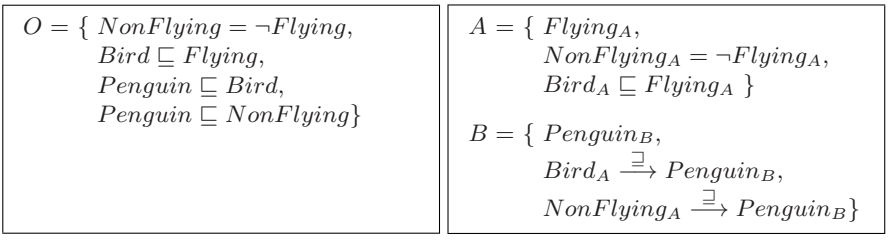

Fig. 1. Figure (a) shows a single ontology where concept *Penguin* is unsatisfiable. Figure (b) shows the same definition with bridge rules where contradiction cannot be detected anymore

there is no inconsistency between two axioms like $Father \sqsubseteq \exists hasChild.Male$ and $Father \sqsubseteq \exists hasChild.\neg Male$ in an ordinary ontology.

This result, together with the fact that inter-ontology subsumption links do not propagate transitively, shows that DDLs can be misleading and counterintuitive, and do not seem to capture the notion of subsumption links across a "Web of ontologies". The example suggests that a formalism for dealing with inter-ontology subsumption relationships is still lacking. $\mathcal{E}$-connections, though suitable for covering a wide variety of relevant modeling scenarios in the Semantic Web context, do not capture the idea of linking ontologies with subsumption relationships, but, as opposed to DDLs, were not conceived for such a purpose.

7 Conclusion and Future Work

One of the most important challenges for the Semantic Web is to understand how to represent and reason with multiple, distinct, independently developed, but linked ontologies in a distributed environment. In this paper, we have explored the $\mathcal{E}$-connections framework as a powerful formalism for combining OWL ontologies.

We argue that $\mathcal{E}$-connections capture the intuition behind many relevant modeling scenarios in which ontology developers "partially import" concepts from other ontologies and use them for the definition of new concepts without having to bring to the local logical space all the axioms in the foreign ontology. We have shown how to integrate the formalism into OWL in a natural and compact way. The resulting syntax and semantic extension of OWL is strictly more expressive than earlier proposals, like C-OWL.

We have defined an expressive subformalism of $\mathcal{E}$-connections, strictly more expressive than DDLs, which can be implemented in a quite straightforward way for the combination of OWL-Lite ontologies by extending existing tableaux-based reasoners. The resulting algorithm has been satisfactorily implemented in the OWL reasoner Pellet. Finally, we have shown how DDLs seem unsatisfactory, because on one hand they are too inexpressive for the modeling scenarios for which $\mathcal{E}$-connections are suitable, and on the other hand because they seem not

to capture the intuitive idea behind subsumption links across multiple ontologies. This suggests further work in defining a new semantics for DDLs that correctly describes inter-ontology subsumption links.

Another important application for $\mathcal{E}$-connections for knowledge engineering and also for the Semantic Web is the ability to factor large ontologies. Instead of combining and integrating ontologies, $\mathcal{E}$-connections could be used the other way round, namely for decomposing large and heterogeneous[5] knowledge bases into smaller, more homogeneous $\mathcal{E}$-connected ontologies. When ontologies grow, they become more difficult to understand for modelers and also harder to reuse. Hence, the ability to "break" large ontologies into smaller, connected pieces may yield to the development of more effective and modular knowledge modeling techniques. Factoring ontologies in this way may also result in a computational efficiency gain when performing reasoning, since the number of non-absorbable general concept inclusion axioms in the original knowledge base is partitioned into different sets, corresponding to the different ontologies in the combination, which may result in a remarkable performance gain.

Future work includes the development of reasoning techniques for handling nominals in the combination (and hence OWL-DL ontologies), and also to explore the transition from PECs to full $\mathcal{E}$-connections. We also plan to extend the algorithm presented here to reason with Aboxes. When reasoning with individuals using this technique, we expect a major performance gain, since the number of individuals in the original ontology is divided among the different ontologies in the combination. Finally, we are also looking into integrating support for multiple ontologies in the SWOOPed ontology editor in order to make these formalisms as usable and intuitive as possible for modelers, which is crucial for successfully bringing them to the Semantic Web.

References

1. F. Baader, C. Lutz, H.Sturm, and F.Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research (JAIR), 16:1-58*, 2003.
2. A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics, 1:153-184*, 2003.
3. Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-owl: Contextualizing ontologies. In *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, 2003.
4. O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence 156(1):1-73*, 2004.
5. P.F. Patel-Schneider, P. Hayes, and I.Horrocks. Web ontology language (owl) abstract syntax and semantics. *W3C Recommendation*, 2004.

[5] The word heterogeneous is used here to state that a certain ontology covers different topics or domains