ANALYSIS OF MAPPING WITHIN S-MODULE FRAMEWORK

Krzysztof Goczyla, Aleksander Waloszek, Wojciech Waloszek and Teresa Zawadzka
Dept. of Software Engineering, Gdansk University of Technology, Narutowicza 11/12, Gdansk, Poland

Keywords: Ontology, Ontology modularization, Description logics, Ontology mapping, Ontology importing.

Abstract: In this paper we present the results of our work on s-module (semantic modules) framework. The framework, introduced recently, consists of a high-level semantic description of a modular knowledge base accompanied by an algebra for manipulating module contents. The main contribution of the article is the presentation of the process of expressing Distributed Description Logics knowledge base within the s-module framework. As the two methods exhibit two different approaches to modularization, analysis of this procedure is helpful in capturing the specifics of DDL, comparing it to other methods, and discussing the completeness of the s-module framework.

1 INTRODUCTION

Recently significant amount of effort has been put in the area of on ontology modularization. Ontologies gain importance in Computer Science, and use of modularization techniques broadens the possibilities of their efficient development and deployment.

In this paper we continue our work from (Goczyla et al., 2009a) on analyzing spaces of semantic modules (s-modules). In (Goczyla et al., 2009a) we described a procedure for constructing a space of possibly useful modules. Construction of the space is algebraic: we specify a set of base modules and a set of operators. Therefore, assimilation of knowledge by one module from another can be depicted as a "shift" in this space and described as a sequence of algebraic operations.

The s-module space was introduced as a common framework for describing properties and characteristics of a modular knowledge base or a specific modularization approach. In this paper we present a procedure of expressing Distributed Description Logics (DDL; Borgida and Serafini, 2003) knowledge base in this space. The conclusions are rather encouraging: such a translation is possible with a minimal number of additional assumptions and with choose of very natural base modules. The description is a source of interesting observations about DDL and s-module space, moreover, it provides an alternative way of proving soundness and completeness of the method.

2 PRELIMINARIES

Due to space limitations we cannot present the full introduction to ontologies formulated in Description Logic (DL) $\mathcal{ALC}$. Here we only review basic terms to establish the notation used henceforth.

In all DLs we assume that we have three sets of names: *constants* (individual names), *concepts* (unary predicates), and *roles* (binary predicates). The *full signature* $\hat{\mathbf{S}}$ contains all the valid names. Other signatures $\mathbf{S}$ are subsets of $\hat{\mathbf{S}}$.

The names are interpreted, and each *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ which assigns each constant an element of $\Delta^{\mathcal{I}}$, each concept a subset of $\Delta^{\mathcal{I}}$, and each role a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We assume that every base interpretation $\mathcal{I}$ of every in fact interprets *all* the valid names from $\hat{\mathbf{S}}$.

Projection $\mathcal{I}|\mathbf{S}$ of a base interpretation to some selected signature $\mathbf{S}$ produces a set of interpretations with the same domain as $\mathcal{I}$ and interpreting all the names from $\mathbf{S}$ in the same way:

$$\mathcal{I}|\mathbf{S} = \{\mathcal{J}: \Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \wedge \forall \mathrm{X} \in \mathbf{S}: \mathrm{X}^{\mathcal{J}} = \mathrm{X}^{\mathcal{I}}\} \quad (1)$$

An ontology $\mathcal{O}$ is simply a set of sentences. A signature of $\mathcal{O}$, denoted $Sig(\mathcal{O})$, is the set of all names used in any sentence of $\mathcal{O}$. An interpretation $\mathcal{I}$ satisfies $\mathcal{O}$ (is its model; denoted $\mathcal{I} \models \mathcal{O}$) iff it satisfies all the sentences in $\mathcal{O}$. Naturally if $\mathcal{I} \models \mathcal{O}$ then every $\mathcal{J} \in \mathcal{I}|Sig(\mathcal{O})$ also satisfies $\mathcal{O}$.

Goczyla K., Waloszek A., Waloszek W. and Zawadzka T..
ANALYSIS OF MAPPING WITHIN S-MODULE FRAMEWORK.
DOI: 10.5220/0003665402670272
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2011), pages 267-272
ISBN: 978-989-8425-80-5

3 S-MODULE SPACE

In this section we describe s-module approach introduced in (Goczyla et al., 2009a). The semantic modules are defined in a way which disregards the exact form of a language (like DL) and focuses only on interpretations. Each semantic module is in fact a set (more precisely, a class) of base interpretations. Each semantic module also has a finite signature **S** which expresses the range of names about which we want to reason using the module:

Definition 1. *A s-module* $M = (\mathbf{S}, \mathbf{W})$ is a pair of a signature **S** and a class **W** of base interpretations, such that $\mathbf{W}|\mathbf{S} = \mathbf{W}$. Each interpretation from **W** is called a *model* of M.

Henceforth we use $\mathbf{S}(M)$ and $\mathbf{W}(M)$ to describe the two parts of a s-module M.

For any ontology $\mathcal{O}$ we might construct a module $M(\mathcal{O})$ such that $\mathbf{W}(M(\mathcal{O})) = \{\mathcal{I}: \mathcal{I} \models \mathcal{O}\}$ and $\mathbf{S}(M(\mathcal{O})) = Sig(\mathcal{O})$. However, while $M(\mathcal{O})$ holds all the possible (base) models of the ontology $\mathcal{O}$, it "forgets" the exact form of sentences; e.g. $M(\{A \sqsubseteq B\}) = M(\{A \equiv A \sqcap B\})$.

(Goczyla et al., 2009a) define a number of operations for s-modules (M, L denote arbitrary s-modules, **S** any signature, γ a function $\hat{\mathbf{S}} \longrightarrow \hat{\mathbf{S}}$):

$$M \cap L = (\mathbf{S}(M) \cup \mathbf{S}(L), \mathbf{W}(M) \cap \mathbf{W}(L)) \quad (2)$$

$$M \cup L = (\mathbf{S}(M) \cup \mathbf{S}(L), \mathbf{W}(M) \cup \mathbf{W}(L)) \quad (3)$$

$$M - L = (\mathbf{S}(M) \cup \mathbf{S}(L), \mathbf{W}(M) - \mathbf{W}(L)) \quad (4)$$

$$\rho_\gamma(M) = (\gamma(\mathbf{S}(M)), \gamma(\mathbf{W}(M))) \quad (5)$$

$$\pi_{\mathbf{S}}(M) = (\mathbf{S}, \mathbf{W}(M)|\mathbf{S}) \quad (6)$$

The operations $\cap$, ρ, π form the backbone of the s-module algebra. The intersection ($\cap$) of s-modules representing ontologies corresponds to adding all sentences (importing) from one ontology to another: $M(\mathcal{O}_1) \cap M(\mathcal{O}_2) = M(\mathcal{O}_1 \cup \mathcal{O}_2)$.

Simple importing is possible only if there is no name conflict between two ontologies. In the presence of name conflicts we can use the rename operator (ρ). Rename operation (ρ) uses the notion of a *signature mapping*, which is a function γ: $\hat{\mathbf{S}} \longrightarrow \hat{\mathbf{S}}$. We also (like in (5)) apply γ to an interpretation in which case $\gamma(\mathcal{I}) = \mathcal{J}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $\forall X \in \hat{\mathbf{S}}$: $\gamma(X)^{\mathcal{J}} = X^{\mathcal{I}}$.

Sometimes we might not want to import all the names from an ontology. To restrict a set of names, but to preserve the relations between extensions of the remaining names, we use projection operator (π).

Example 1 Let $\mathcal{O}_1 = \{Teacher \sqsubseteq Employee, Publication \sqsubseteq Achievement\}$, and assume that $\mathcal{O}_2$ is much larger and contains axioms like: $Book \sqsubseteq Publication$, $Chapter \sqsubseteq Publication$, $Monograph \sqsubseteq Book$ etc. We like to reuse $\mathcal{O}_2$ in $\mathcal{O}_1$, however, in $\mathcal{O}_2$ the notion of *Publication* does not include position papers. Moreover, we find the term *Book* from $\mathcal{O}_2$ too general and do not want to include it the ontology. To meet our goals we take $\mathcal{O}_1' = \mathcal{O}_1 \cup \{NPPaper \equiv Publication \sqcap \neg PositionPaper\}$ and construct M: $M = M(\mathcal{O}_1') \cap \pi_{\mathbf{S}-Book}(\rho_{Publication \to NPPaper}(M(\mathcal{O}_2)))$.

In Ex. 1 we introduce some intuitive shortcuts to notation that we also exploit further in the paper. For example by $\rho_{A \to B}$ we mean that corresponding γ function changes only the name A to B and by $\pi_{\mathbf{S}-X}(M)$ we mean $\pi_{\mathbf{S}(M)-\{X\}}(M)$. Occasionally for denoting names we might use wildcards: e.g. $* \to 1{:}*$.

While $\cap$, $\cup$, $-$ are taken directly from Boolean algebra of sets,, π and ρ are equivalents of cylindrification and substitution from the cylindric algebra (Henkin, Monk and Tarski, 1971). Any *space of module* (class of modules **M** closed under π, ρ, $\cap$, $\cup$, $-$) can be used to construct a cylindric algebra.

A space of modules can be constructed by choosing a *base space* and closing it wrt. π, ρ, $\cap$, $\cup$, $-$. A very natural choice of a base space is $\{M(\alpha)\}$ where α is a sentence valid in a selected language $\mathcal{L}$.

Several auxiliary operators can be introduced for such a space. The *selection* operator σ is a shortcut for $\sigma_\alpha(M) = M \cap M(\alpha)$. The further two operators "put under" (υ) and *restriction* (ξ) are defined below ($\mathcal{I} \cap S$ denotes an interpretation $\mathcal{J}$: $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cap S$ and $\forall X \in \hat{\mathbf{S}}$: $X^{\mathcal{J}} = X^{\mathcal{I}} \cap S$):

$$L \,\upsilon_C\, M = (\mathbf{S}(L) \cup \mathbf{S}(M), \{\mathcal{I} \in \mathbf{W}(M): \mathcal{I} \cap C^{\mathcal{I}} \in \mathbf{W}(L)\}) \quad (7)$$

$$\xi_C(M) = (\mathbf{S}(M), \{\mathcal{I} \cap C^{\mathcal{I}}: \mathcal{I} \in \mathbf{W}(M) \wedge \mathcal{I} \cap C^{\mathcal{I}} \in \mathbf{W}(M)\}) \quad (8)$$

"Put under" (Goczyla *et al.*, 2009a) correlates the domains of two modules by introducing relationships between extensions of terms in L only to a fragment of M. The restriction operator ξ is an operator complementing υ. Namely it restricts the domain of the module to the extension given concept C. For $\mathcal{ALC}$ it may be simply treated as a shortcut for $\xi_C(M) = \sigma_{C \equiv \top}(M)$.

Theorem 1. For every module M from $\mathbf{M}(\mathcal{ALC})$ obtained from the basic space $\{M(\alpha)\}$ with use of operators (π, ρ, $\cap$, $\cup$, σ, υ, ξ) it is decidable whether the module is satisfiable (i.e. $\mathbf{W}(M) \neq \varnothing$).

4 DISTRIBUTED DL

Distributed Description Logics (DDL) is one of the most prominent modularization methods for DL ontologies. Originally proposed by Borgida and Serafini in (2003), it was extended and adapted in many works. The presentation in this Section is mainly based on (Homola and Serafini, 2010).

DDLs focus on mapping the terms from a source module to a target module. We assume there exists a collection of modules $\{O_i\}_{i \in I}$, indexed by a set I. Each module is simply an ontology (it has its local collection of sentences). Between each pair of modules O_i (as a source) and O_j (as a target; here and hence in this section $i, j \in I$, $i \neq j$) there is defined a (possibly empty) set of *bridge rules* $\mathfrak{B}_{ij}$.

There are three types of bridge rules (C, D are concepts and a, b constants, resp. from O_1 and O_2):

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

A *distributed knowledge base* (DKB) $\mathfrak{K} = (\{O_i\}, \{\mathfrak{B}_{ij}\})$, consists of modules and sets of bridge rules. Whenever $\mathfrak{B}_{ij}$ is non-empty, we say that O_j *uses* O_i.

A *distributed interpretation* $\mathfrak{I}$ is a pair $(\{\mathcal{I}_i\}, \{r_{ij}\})$, where $\{\mathcal{I}_i\}$ are interpretations (called *local interpretation*), and $\{r_{ij}\}$ are *domain relations* between the domains of $\mathcal{I}_i$ and $\mathcal{I}_j$. In contrast to standard DL, each local interpretation might also be *a hole*, a special interpretation $\mathcal{I}_\epsilon$ with empty domain. A distributed interpretation $\mathfrak{I}$ is a model of $\mathfrak{K}$ iff for each $i, j \in I$, we have $\mathcal{I}_i \vDash O_i$ and $\mathfrak{I} \vDash \mathfrak{B}_{ij}$. $\mathfrak{I} \vDash \mathfrak{B}_{ij}$ iff it satisfies all the rules $\mathfrak{B}_{ij}$ according to the following:

$\mathfrak{I} \vDash i: C \xrightarrow{\sqsubseteq} j: D$ iff $r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$
$\mathfrak{I} \vDash i: C \xrightarrow{\sqsupseteq} j: D$ iff $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$
$\mathfrak{I} \vDash i: a \longrightarrow j: b$ iff $r_{ij}(a^{\mathcal{I}_i}) \ni d^{\mathcal{I}_j}$

DDL exhibits a different behavior than s-modules. While the latter focuses on importing, DDL focuses on mapping between terms (this distinction is based on (Homola and Serafini, 2010)).

While in the basic DDL relation r_{ij} might be of any form, one might consider also more constrained versions of DDL, denoted by additional symbols: e.g. F for only functional r_{ij} or I for injective r_{ij}, e.g. DDL(F) or DDL(F, I). The following well-known "penguin" example illustrates the importance of the relation.

Example 2 (Grau et al., 2004) Let us consider the ontology $O_1 = \{Nonflying \equiv \neg Flying, Bird \sqsubseteq Flying\}$ and the ontology $O_2 = \{Penguin \sqsubseteq \top\}$. We define the mapping $\mathfrak{B}_{12}$ in the following way:

1: *Nonflying* $\xrightarrow{\sqsupseteq}$ 2:*Penguin*
1: *Bird* $\xrightarrow{\sqsupseteq}$ 2:*Penguin*

It might seems that *Penguin* is unsatisfiable "being subsumed" by both *Nonflying* and *Bird*. But we can still obtain a non-empty interpretation $\mathcal{I}_2$ in a model of $\mathfrak{K}$, if the relation r_{12} maps at least two individuals (one *Nonflying* and one *Bird*) to a single *Penguin*.

Originally intended for illustrating cumbersome behavior of DDL, in fact this example shows its distinctive capability: to combine knowledge about several individuals into one. In situations when such behavior is undesirable we can turn to DDL(F).

5 DDL IN S-MODULE SPACE

In this section we present the results of our work on expressing DDL in the s-module framework. Starting from a bit simplified conversion for DDL(F, I), we gradually move to less constrained versions of DDL.

5.1 DDL(F, I) with No Cycles

At first we consider a case of DDL(F, I) in which each individual from $\Delta^{\mathcal{I}_i}$ corresponds to at most one individual from $\Delta^{\mathcal{I}_j}$ and vice versa.

As it turns out, such assumption significantly reduces the difficulties of bridging DDL and s-modules. As a case-study let us consider a distributed KB $\mathfrak{K}$ with two simple DDL-modules $O_1 = \{C \sqsubseteq \top, D \sqsubseteq \top\}$ and $O_2 = \{E \sqsubseteq F\}$ and a set of rules $\mathfrak{B}_{21} = \{2{:}F \xrightarrow{\sqsubseteq} 1{:}C, 2{:}E \xrightarrow{\sqsupseteq} 1{:}D\}$. Such a mapping implies that for every model $\mathfrak{I}$, $\mathcal{I}_1 \vDash D \sqsubseteq C$.

Despite apparent simplicity of the example, while analyzing semantics we still have to consider several possibilities: a domain relation r_{21} might map the whole domain of $\mathcal{I}_2$ to $\Delta^{\mathcal{I}_1}$, or only a fragment of a domain of $\mathcal{I}_2$, or r_{21} might even be empty, resulting in empty interpretation for D. The second case is depicted in Fig. 1a with use of Venn diagrams: we can mentally visualize that with shrinking of $r_{21}(E^{\mathcal{I}_2})$ the area of $D^{\mathcal{I}_1}$ is also reduced.

To reflect this effect for s-modules, we have to simulate the behavior of r_{21}. The constructions of a s-module M_1 representing possible models of O_1 in $\mathfrak{K}$ proceeds as follows. First, we create a s-module with two special concepts: O_1 and O_2. Second, we put under these concepts modules $M(O_1)$ and $M(O_2)$ respectively (if they contain repeating names, we have to add prefixes). Subsequently, we enforce the bridge rules by using selection. Then, we project the signature only to the terms from $M(O_1)$. Finally, we

restrict the module to the concept O_1 and remove the concept from the signature.

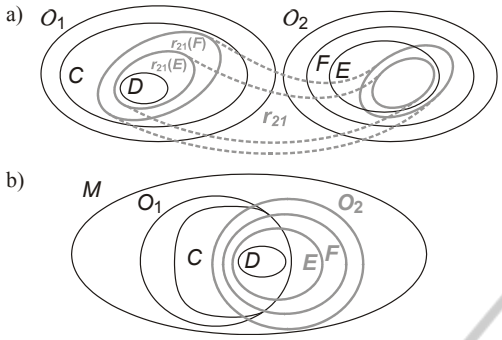

Figure 1: Different but equivalent effects of DDL(F) mapping (a) and combining s-modules (b).

The result of the first three steps is depicted in Fig. 1b. We can see that the outcome gives a similar effect as in the case of DDL. The area $r_{21}(\Delta^{\mathcal{I}_2})$ is represented by the intersection $O_1 \sqcap O_2$. Since the interpretation of the intersection may vary in size, and may even be empty, all the possible forms of $r_{21}(\Delta^{\mathcal{I}_2})$ are reflected by models of M. The fact that r_{21} is injective and functional is advantageous here: each instance of $O_1 \sqcap O_2$ represents one element e of $\Delta^{\mathcal{I}_1}$ and simultaneously one element $r_{21}^{-}(e)$ of $\Delta^{\mathcal{I}_2}$.

The procedure sketched above can be generalized and formalized as follows:

Definition 2. For a given DKB $\mathfrak{K} = (\{O_i\}, \{\mathfrak{B}_{ij}\})$, $i, j \in I$, $i \neq j$, a *converting function* c is a function that assigns each O_i a s-module.

Definition 3. A *bridge-rule operation* β_b for a bridge rule b and a module M is:

for $b = i{:}\ C \xrightarrow{\sqsubseteq} j{:}\ D$:
$\beta_b(M) = \sigma_\alpha(M), \alpha = \gamma_{* \to i:*}(C) \sqsubseteq \gamma_{* \to j:*}(D)$
for $b = i{:}\ C \xrightarrow{\sqsupseteq} j{:}\ D$:
$\beta_b(M) = \sigma_\alpha(M), \alpha = \gamma_{* \to j:*}(D) \sqsubseteq \gamma_{* \to i:*}(C)$
for $b = i{:}\ a \longrightarrow j{:}\ b$:
$\beta_b(M) = \gamma_{i:a \to j:b}(M)$

where by $\gamma(C)$ we understand a new concept with all the names substituted with use of γ.
A *bridge-rule operation* $\beta_{\mathfrak{B}}$ for a set of bridge rules $\mathfrak{B}$ and a module M is a composition of β_b, $b \in \mathfrak{B}$.

Definition 4. For given two modules O_i and O_j from a DKB $\mathfrak{K}$, such that O_j uses O_i, and a converting function c, *a s-module integrating* O_i and O_j *wrt. c* is:

$$M_{ij}^{c} = \beta_{\mathfrak{B}ij}(\rho_{* \to i:*}(c(O_i)) \upsilon_{O_i} M(\{O_i \sqsubseteq \top\}) \cap \rho_{* \to j:*}(M(O_j)) \upsilon_{O_j} M(\{O_j \sqsubseteq \top\}))$$

The construction of the integrating s-module corresponds to executing the three first steps of the described procedure (see also Fig. 1b).

Definition 5. For a given DKB $\mathfrak{K} = (\{O_i\}, \{\mathfrak{B}_{ij}\})$, a module O_j, and a converting function c, *an integrated s-module for* O_j *wrt. c* is $M_j^c = \cap_{i \in \{i:\, \mathfrak{B}_{ij} \neq \emptyset\}} M_{ij}^{c}$. *A fully integrated module for* O_j *wrt. c* is $FM_j^c = \rho_{j:* \to *}(\pi_{\mathbf{S} - O_j}(\xi_{O_j}(M_j^c)))$.

The notion of integrated module generalize the described procedure to the case when more modules are used. Full integration corresponds to the last two steps of the procedure. A fully integrated s-module is indeed useful for describing DDL semantics, as the following lemma shows.

Lemma 1. For a DKB $\mathfrak{K} = (\{O_i\}, \{\mathfrak{B}_{ij}\})$, $i, j \in I$, $i \neq j$, expressed in $\mathcal{ALC}$ and DDL(F, I), in which O_1 uses all the other modules, and all the other modules use none, and a converting function $c(O_i) = M(O_i)$ for all $i \neq 1$, $c(O_1) = FM_1^c$, a module O_i is satisfiable (i.e. for some model it has a local interpretation which is not a hole) iff $c(O_i) \neq M(\{\top \sqsubseteq \bot\})$.

The proof, omitted for brevity, consists of showing that a model $\mathfrak{I}$ with non-empty $\mathcal{I}_1$ exists iff there exists a model of M_1^c.

The result from Lemma 8 can be generalized to any acyclic DKB (i.e. DKB $\mathfrak{K}$ for which the relation $U_{\mathfrak{K}} = \{(O_i, O_j)\colon \mathfrak{B}_{ij} \neq \emptyset\}$ forms a forest).

Proposition 1. For any acyclic DKB $\mathfrak{K} = (\{O_i\}, \{\mathfrak{B}_{ij}\})$, $i, j \in I$, $i \neq j$, expressed in $\mathcal{ALC}$ and DDL(F, I), and a converting function $c(O_i)$ defined recursively as $c(O_i) = M(O_i)$ for leaves, and $c(O_i) = FM_i^c$ for other modules, a module O_i is satisfiable iff $c(O_i) \neq M(\{\top \sqsubseteq \bot\})$.

With use of Lemma 1 the proof is straightforward, by induction on each tree of using relation (Lemma 8 forms the induction base, and gives means for proving the induction hypothesis).

5.2 DDL(F, N^n) with No Cycles

Here we extend the results from the previous subsection towards slightly more expressive DDL, by adapting the introduced notion to the case when the domain relations are not necessarily injective.

Once again we start with a motivation example. We adapt the "penguin" example (see Ex. 3). DKB consists of two modules $O_1 = \{P \sqsubseteq \top\}$, and $O_2 = \{NF \sqsubseteq \neg F, B \sqsubseteq NF\}$, and one non-empty bridge rule set is: $\mathfrak{B}_{21} = \{2{:}NF \xrightarrow{\sqsupseteq} 1{:}P, 2{:}B \xrightarrow{\sqsupseteq} :P\}$.

As already mentioned above, the concept P in O_1 may be satisfiable, though for this to happen r_{21} has to map two individuals of $\Delta^{\mathcal{I}_2}$ to a single individual

of $\Delta^{\mathcal{I}_1}$. This situation is depicted in Fig. 2a.

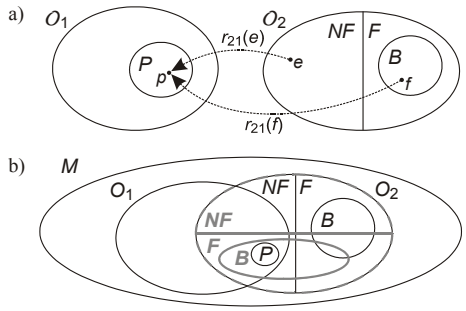

Figure 2: Different but (almost) equivalent effects of DDL(F, N^n) mapping (a) and combining s-modules (b).

The strategy from the previous section is not enough to model this situation in the realm of s-modules. Although we can overlap the domains of the two modules, simple overlapping (like in Fig. 1b) would render the concept P unsatisfiable. We have to somehow model the possibility of mapping two individuals into one.

A solution to this issue is illustrated in Fig. 2b. The main idea is to apply the conceptual decomposition twice to the same domain. It can be done with prefixes (omitted in Fig. 2 for readability), appropriate s-module operation might look like $M_{2.2} = \rho_{* \to 2.2.1:*}(M(\mathrm{O}_2)) \cap \rho_{* \to 2.2.2:*}(M(\mathrm{O}_2)$. After the transformation, every element of the domain of (any model of) $M_{2.2}$ represents in fact a pair of elements of the domain of (some model of) $M(\mathcal{O}_2)$.

There are, however, two issues connected with this approach. First of all, the constructed module represent *pairs* of the original domains. The same approach can be used to triples, quadruples etc., but there have to be some known and finite limit to the cardinality of the tuples. This is the motivation behind introducing a new constraint for DDL, namely N^n, $n \in \boldsymbol{N}$, which implies that every domain relation r_{ij} is at most n-to-one. The discussion in this section is thus constrained to DDL(F, N^n).

Second issue is that elements of the domain of $M_{2.2}$ represent in fact some pairs of elements of the original domain, like (e, e), that we do not want to include in our considerations. This problem can be technically overcome (by exploiting *disjoint union satisfiability property* of $\mathcal{ALC}$ introduced by Serafini *et al*. in (2005)), but the discrepancy between "double overlapping" and pair of domains still exists, and should be dealt with in future development of s-module framework (see Sec. 6).

In the following we adapt the notions from the previous section to the case of DDL(F, N^n).

Definition 6. A *n-bridge-rule operation* $\beta_b{}^n$ for a bridge rule b, a module M and given n is:

for $b = i{:}\ C \xrightarrow{\sqsubseteq} j{:}\ D$:
$\beta_b(M) = \sigma_\alpha(M)$, where α is defined below:
$\alpha = \bigsqcup_{k \in [1..n]} \bigsqcup_{l \in [1..k]} \gamma_{* \to i.k.l:*}(C) \sqsubseteq \gamma_{* \to j:*}(D)$
for $b = i{:}\ C \xrightarrow{\exists} j{:}\ D$:
$\beta_b(M) = \sigma_\alpha(M)$, where α is defined below:
$\alpha = \gamma_{* \to j:*}(D) \sqsubseteq \bigsqcup_{k \in [1..n]} \bigsqcup_{l \in [1..k]} \gamma_{* \to i.k.l:*}(C)$
for $b = i{:}\ a \longrightarrow j{:}\ b$:
$\beta_b(M) = \bigcup_{k \in 1..n} \gamma_{i.k.1:* \to j:b}(M)$

A *n-bridge-rule operation* $\beta_{\mathfrak{B}}{}^n$ for a set of bridge rules $\mathfrak{B}$, a module M and given n is a composition of $\beta_b{}^n$ for every $b \in \mathfrak{B}$.

Definition 7. For given two modules $\mathcal{O}_i$ and $\mathcal{O}_j$ from a DKB $\mathfrak{K}$, such that $\mathcal{O}_j$ uses $\mathcal{O}_i$, and a converting function c, let M be defined as follows:

$$M = M(\{O_{i.k} \sqcap O_{i.l} \sqsubseteq \bot{:}\ k, l \in [1..n], k \neq l\})$$

a s-module n-integrating $\mathcal{O}_i$ and $\mathcal{O}_j$ *wrt. c* is:

$${}^nM_{ij}{}^{\mathrm{c}} = \beta_{\mathfrak{B}ij}{}^n(\ \rho_{* \to j:*}(M(\mathcal{O}_j))\ \upsilon_{\mathcal{O}_j}\ M(\{O_j \sqsubseteq \top\}) \cap \bigcap_{k,\, l \in [1..n],\, k \neq l} (\rho_{* \to i.k.l:*}(c(\mathcal{O}_i))\ \upsilon_{O_{i.k.l}}\ M)\ )$$

The pairwisely disjoint concepts $O_{j.k}$ represent k-tuples of elements of the original domain.

Definition 8. For a given DKB $\mathfrak{K} = (\{\mathcal{O}_i\}, \{\mathfrak{B}_{ij}\})$, a module $\mathcal{O}_j$, a converting function c, and a number n *an n-integrated s-module for* $\mathcal{O}_j$ *wrt. c* is ${}^nM_j^c = \bigcap_{i \in \{i:\, \mathfrak{B}ij \neq \varnothing\}} {}^nM_{ij}{}^{\mathrm{c}}$. *A fully n-integrated module for* $\mathcal{O}_j$ *wrt. c* is $F^nM_j^c = \rho_{j:* \to *}(\pi_{\mathbf{S} - O_j}(\xi_{Oj}({}^nM_j^c)))$.

Once again we show that fully integrated modules are equisatisfiable with corresponding modules from DKB.

Lemma 2. For a DKB $\mathfrak{K} = (\{\mathcal{O}_i\}, \{\mathfrak{B}_{ij}\})$, $i, j \in I$, $i \neq j$, expressed in $\mathcal{ALC}$ and DDL(F, N^n), in which $\mathcal{O}_1$ uses all the other modules, and all the other modules use none, and a converting function $c(\mathcal{O}_i) = M(\mathcal{O}_i)$ for all $i \neq 1$, $c(\mathcal{O}_1) = F^nM_1{}^c$, a module $\mathcal{O}_i$ is satisfiable iff $c(\mathcal{O}_i) \neq M(\{\top \sqsubseteq \bot\})$.

The proof, which we omit for brevity, shows that a model $\mathfrak{I}$ with non-empty $\mathcal{I}_1$ exists iff there exists a model of ${}^nM_1{}^c$. Again, we can generalize the results of the lemma to a case of any acyclic DKB.

Proposition 2. For any acyclic DKB $\mathfrak{K} = (\{\mathcal{O}_i\}, \{\mathfrak{B}_{ij}\})$, $i, j \in I$, $i \neq j$, expressed in $\mathcal{ALC}$ and DDL(F, N^n), and a converting function $c(\mathcal{O}_i)$ defined recursively as $c(\mathcal{O}_i) = M(\mathcal{O}_i)$ for leaves, and $c(\mathcal{O}_i) = F^nM_i^c$ for other modules, a module $\mathcal{O}_i$ is satisfiable iff $c(\mathcal{O}_i) \neq M(\{\top \sqsubseteq \bot\})$.

Proof (sketch): Analogously like in proof for Prop. 1, but with use of Lemma 13.

5.3 Decidability

The discussion from the previous points gives us also means for creating a procedure for deciding satisfiability of modules in a DKB.

The decidability result from Th. 1 combined with Prop. 2 allows for immediate stating that DDL(F, N^n) is decidable for acyclic DKBs. However, we can extend this result a bit by including the DKBs which can contain cycles.

A basic idea behind such extention is simple: we proceed iteratively with determining $c(\mathcal{O}_i)$ for each module, assuming that in first iteration $c_1(\mathcal{O}_i) = M(\mathcal{O}_i)$ and then, in the next k-th iteration taking $c_k(\mathcal{O}_i) = F^n M_i^{c_{k-1}}(\mathcal{O}_i)$. As Serafini and Tamilin show in (2007), the fixpoint will finally be reached, which can be detected by adapted procedure for checking whether an ontology is a conservative extension of another (Lutz, Walther and Wolter, show in (2007) that this problem for $\mathcal{ALC}$ is decidable).

This leads us to the following conclusion:

Proposition 3. For a given DKB $\mathcal{K} = (\{\mathcal{O}_i\}, \{\mathcal{B}_{ij}\})$, $i, j \in I$, $i \neq j$, an recursive procedure for converting modules in the following way: $c_1(\mathcal{O}_i) = M(\mathcal{O}_i)$, $c_k(\mathcal{O}_i) = F^n M_i^{c_{k-1}}(\mathcal{O}_i)$, repeated until $c_k(\mathcal{O}_i)$ is a conservative extension of $c_{k-1}(\mathcal{O}_i)$ for all $i \in I$, is a terminating, sound and complete procedure for deciding satisfiability of modules for $\mathcal{ALC}$ and DDL(F, N^n).

6 CONCLUSIONS

In this section we summarize the main observations and contributions of the paper and relate them to other studies.

From the point of view of DDL, the results allows us to show some insight in the relation between mapping and importing (Homola and Serafini, 2010). Here we show how different kinds of mappings relate to specific kinds of importing (especially "putting under"). Further work will allow us to include also $\mathcal{E}$-Connection (Kutz, Lutz, Wolter, and Zakharyaschev, 2004) and P-DL (Bao, Voutsadakis, Slutzki, and Honavar, 2009), two other major methods of modularization.

The other result is an alternative way of proving decidability of DDL(F, N^n) for $\mathcal{ALC}$. Though at the current stage of research it does not extend the results already available in literature, it shows the practical application of the results from Th. 1. The further development might result in a set of techniques for proving decidability for a wide range of modularization methods.

From the perspective of s-module framework the presented discussion provides interesting hints about its further development. The s-module framework cannot easily handle situations in which we want to refer to a tuple of elements of a domain. Sec. 5.3 suggests it may be useful to extend the framework by some kind of treatment for limits (i.e. the ability to determine bounds for an arbitrary set of modules).

Finally, the paper presents some extensions to the framework of s-modules: definition of s-module space, restriction operator, and a slightly extended result for decidability (cf. Sec. 3).

ACKNOWLEDGEMENTS

This work is partially supported by the Polish National Centre for Research and Development under Grant No. SP/I/1/77065/10 by the strategic scientific research and experimental development program: „Interdisciplinary System for Interactive Scientific and Scientific-Technical Information".

REFERENCES

Bao, J., Voutsadakis, G., Slutzki, G. & Honavar, V. (2009). Package-Based Description Logics. In: *Modular Ontologies*. Springer: Berlin Heidelberg.

Borgida, A. & Serafini, L. (2003). Distributed Description Logics: Assimilating Information from Peer Sources. *J. Data Semantics*, 1, 153-184.

Grau, B. C., Parsia, B., & Sirin, E. (2004). Working with Multiple Ontologies on the Semantic Web. In: *The Semantic Web – ISWC 2004* (pp. 620-634).

Goczyla, K., Waloszek, A. & Waloszek, W. (2009a). S-modules - Approach to Capture Semantics of Modularized DL Knowledge Bases. *Proc. of KEOD 2009* (pp. 117-122).

Goczyla, K., Waloszek, A. & Waloszek, W. (2009b) A Semantic Algebra for Modularized Description Logics Knowledge Bases. *Proc. of DL 2009*.

Homola, M. & Serafini, L. (2010). Towards Formal Comparison of Ontology Linking, Mapping and Importing. *Proc. of DL2010*.

Kutz, O., Lutz, C., Wolter, F. & Zakharyaschev, M. (2004). E-connections of abstract description systems. *Artificial Intelligence*, 156(1), 1-73.

Lutz, C., Walther, D. & Wolter, F. (2007). Conservative extensions in expressive description logics. *Proc. of IJCAI-2007*, 453-459. doi:10.1.1.117.2884.

Serafini, L., Borgida, A. & Tamilin, A. (2005). Aspects of Distributed and Modular Ontology Reasoning. In *Proc. of IJCAI 2005*, pp. 570-575.

Serafini, L. & Tamilin, A. (2007). Aspects of Distributed and Modular Ontology Reasoning. *Technical report*.