Modularized Knowledge Bases Using Contexts, Conglomerates and a Query Language[*]

Krzysztof Goczyła, Aleksander Waloszek, Wojciech Waloszek, and Teresa Zawadzka

Gdansk University of Technology, Poland
{kris,alwal,wowal,tegra}@eti.pg.gda.pl

Abstract. The paper presents a novel approach to design and development of a modularized knowledge base. It is assumed that the knowledge base consists of a terminology (axioms) and a world description (assertions), both formulated in a Description Logics (DL) dialect. The approach is oriented towards decomposition of a knowledge base into logical components called contexts and further into semantic components called conglomerates. Both notions were elaborated separately elsewhere. The paper shows how contexts and conglomerates concepts can work in harmony to create a maintainable knowledge base. An architecture of a system that conforms to this approach, which additionally uses a query language called KQL (Knowledge Query Language), is presented. The approach is intended to be used to build a prototypical system that aims at integrating knowledge on cultural heritage coming from digital libraries, including user-defined libraries. The thorough discussion of related work is also given.

Keywords: ontology, knowledge base, modularization, contexts, conglomerates, knowledge integration.

1 Introduction

In this paper we present a novel approach to design and implementation of large and scalable knowledge bases that conform to Description Logics [1] dialects expressiveness (according to W3C standards: that conform to OWL2 [2] with SWRL [3] extensions). This approach is based on three major concepts: contexts, conglomerates and a query language. Contexts are to logically situate a knowledge based in different frameworks so that logical statements are interpreted in a proper way, according to a given "situation". Conglomerates are to divide a complex, large knowledge base into components, like a relational database is divided into relations. Conglomerates can be operated upon by algebraic operations, like relations in a relational database. The operations can be performed via Knowledge Query Language

[*] This work is partially supported by the Polish National Centre for Research and Development (NCBiR) 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".

(KQL) statements that structurally remind that of SQL statements, which make them more easily understood by systems engineers.

The paper is organized as follows. In the Section 2 we present the notion of contexts. In Section 3 we present the notion of conglomerates. In Section 4 the KQL language is presented. In Section 5 the architecture of the system that incorporates the three components is presented. The system is to be implemented as a part of the SyNat system intended to make the cultural and science Polish heritage available worldwide. Section 6 presents the related work. In Section 7 we conclude the paper.

2 Contexts

Contextualizing knowledge bases makes the process of their deployment and use more effective. Contextualized knowledge bases, if contextualization is properly conducted, are easier to design and maintain. Moreover, with use of contextualization, the efficiency of reasoning process can be greatly improved. In this section we describe our approach to modularization, *Structured-Interpretation Model* (SIM), an approach that has several distinguishing features, the most important of which is the relative straightforwardness of adapting the structure of knowledge the base to evolving needs of the user.

As a basis we have taken Description Logic formalism and we adapted to it the notions of generalization and context instance. Let us recall that a DL ontology consists of a terminology (TBox) and a world description (ABox). The Formal definition of contextualized DL ontology is as follows:

Definition 2.1. *A* contextualized TBox $T = (\{T_i\}_{i \in I}, \trianglelefteq)$ *consists of a set of TBoxes whose elements are called* contexts, *and a* generalization relation $\trianglelefteq \subseteq I \times I$ *which is a partial order established over the set of indexes* I. *The relation is acyclic and contains the only least element* m. *We also introduce the following notions*:

T_m *called the* root context *of the contextualized TBox* T,
T_i generalizes T_j *iff* $i \trianglelefteq j$,
T_i specializes T_j *iff* $j \trianglelefteq i$. □

The idea behind introducing such hierarchical arrangement of contexts was to allow for constrained interactions between parts of terminology. The general rule here is that more specialized terminologies may "see" more general ones, but more general terminologies may be unaware of the existence of more specialized ones.

Introduced contexts encompass only terminology. To deal with assertional part of the knowledge base we allow for creation of many ABoxes for one terminology. We call these ABoxes *context instances*.

Definition 2.2. *A* contextualized ABox $A = (\{A_j\}_{j \in J}, \text{inst}, \lessdot)$ *of contextualized TBox* $T = (\{T_i\}_{i \in I}, \trianglelefteq)$ *is a triple consisting of:*

1. *A set of ABoxes* $\{A_j\}_{j \in J}$, *each of which is called an* instance of context,
2. *The function* inst: $J \rightarrow I$ *relating each ABox from* $\{A_j\}_{j \in J}$ *with TBox from* $\{T_i\}_{i \in I}$,
3. *The* aggregation relation $\lessdot \subseteq J \times J$, *which is a partial order established over the set of indexes* J. *We require that:*

a. $\lessdot$ *is acyclic and contains the least element* n,
b. inst(n) = m, *where* m *is the least element of the relation* $\trianglelefteq$,
c. *for each* j $\lessdot$ k *such that* j $\neq$ k *holds* inst(j) $\trianglelefteq$ inst(k) *and* inst(j) $\neq$ inst(k).

We also say that:

A_n is called the root context instance of the contextualized ABox A,
A_j *is* an instance of the context T_i *iff* inst(j) = i,
A_j aggregates A_k *iff* j $\lessdot$ k,
A_j is aggregated by A_k *iff* k $\lessdot$ j,
A_j *is* an aggregating context instance *iff* $\exists$k: j $\lessdot$ k □

Definition 2.3. *A* contextualized knowledge base K = (T, A) *consists of a contextualized TBox* T *and a contextualized ABox* A *of* T. □

Contextualized knowledge base is given the interpretation in a specific way: each context instance is given its own interpretation. Such an approach gives some level of locality within context instances.

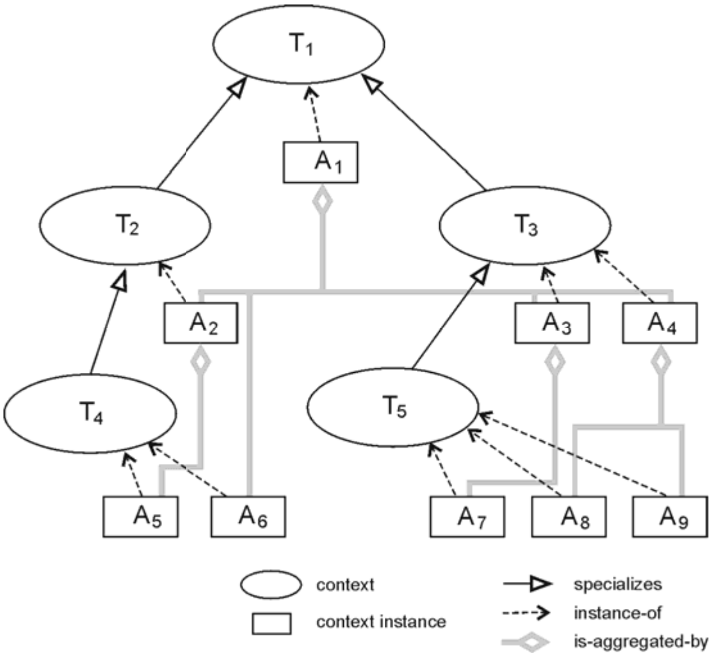

Fig. 1. An example of contextualized knowledge base

The general schema of a contextualized DL-ontology (a DL-knowledge base) is presented in Fig. 1. In this figure relationships between context instances and between context instances and contexts are depicted in the form of graph, e.g. the instance A_4 aggregates instances A_8 and A_9. For the sake of clarity only the transitive reductions of generalization and aggregation relations have been depicted.

The example of the aggregation conformance of denotation is presented in Fig. 2. Here we have three contexts: T_1 that describes general notions of WOMAN and MAN, T_2 that specializes T_1 towards description of voices in a choir, and T_3 that also specializes T_1 but towards description of social relations. Context instance A_1 aggregates context instances A_2 and A_3. Although ABox of A_1 is empty, interpretation $\mathcal{I}_1$ in order to be a model of the knowledge base has to assign Mary to the concept WOMAN (i.e. $\mathtt{Mary}^{\mathcal{I}_1} \in \mathtt{WOMAN}^{\mathcal{I}_1}$). As a consequence of this, the same rule enforces that in the interpretation $\mathcal{I}_3$ Mary is assigned to the concept WIFE (i.e. $\mathtt{Mary}^{\mathcal{I}_3} \in \mathtt{WIFE}^{\mathcal{I}_3}$), as the information about Mary being a woman "flows" down the aggregation relationships.

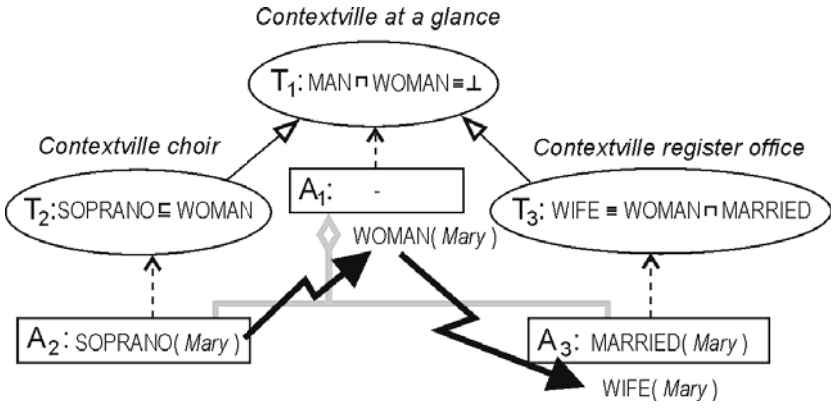

Fig. 2. An example of the aggregation conformance of denotation

Other aspects of contexts and contextualized knowledge bases can be found in [4].

3 Conglomerates

Conglomerates are used to modularize large knowledge bases into manageable pieces like relations in a relational database. The main motivation behind conglomerates is the hope to reach the maturity of the collaborative ontology development and reuse comparable to the one achieved by software engineering methods in the case of software modules.

In the following we introduce basic notions of conglomerates algebra and present examples of its use.

3.1 Basics

An assumption we take is that a user is interested in conclusions that can be drawn from an ontology rather than in particular axioms and assertions. So, we define an ontology module strictly semantically, focusing only on its interpretations.

Definition 3.1. (conglomerate). A conglomerate $M = (\mathrm{S}, \mathrm{W})$ is a pair of a signature S (called a *signature* of the conglomerate) and a class W of S-interpretations (called *models* of the conglomerate). The two parts of M are denoted as $\mathrm{S}(M)$ and $\mathrm{W}(M)$, respectively. Each S-interpretation from W we call a *model* of M. □

According to this definition, each conglomerate simply consists of all its models. We say that a conglomerate satisfies a particular sentence α, denoted $M \vDash \alpha$, iff $\forall \mathcal{I} \in \mathrm{W}(M)\colon \mathcal{I} \vDash \alpha$.

We think that creators of an ontology are not able to foresee all its future uses. By necessity, the creators have to focus on a small set of chosen contexts of use of particular modules, and the contexts of their choice may not be adequate for a particular application of a knowledge base with this ontology. So, the conglomerate algebra puts stress on various methods of manipulation for them; that in general allow for changing and combining signature and models.

3.2 Operators

We assume that description logic $\mathcal{L}$ and domain set Δ are chosen and fixed. We denote a set of all modules as $\mathcal{M}$, a set of all signatures as Σ, and a set of all S-interpretations as $\mathcal{I}(S)$. We also use the notion of a *projection* $\mathcal{I}' = \mathcal{I}|\mathrm{S}'$ of an S-interpretation $\mathcal{I}$ to a signature S' ($\mathrm{S}' \subseteq \mathrm{S}$). $\mathcal{I}'$ is an S'-interpretation for which the following holds: $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $X^{\mathcal{I}'} = X^{\mathcal{I}}$ for every $X \in \mathrm{S}'$.

Extend

$$\varepsilon_{\mathrm{S}}\colon \mathcal{M} \to \mathcal{M},\ \mathrm{S} \in \Sigma;$$
$$\varepsilon_{\mathrm{S}}(M) = (\mathrm{S}(M) \cup \mathrm{S},\ \{\mathcal{I} \in \mathcal{I}(\mathrm{S}(M) \cup \mathrm{S})\colon \mathcal{I}|\mathrm{S}(M) \in \mathrm{W}(M)\}).$$ □

Extension extends a signature of a given module M by names from a given signature S. The allowed set of interpretations of each original name is preserved, and so are the relationships between original concepts, roles, and individuals (e.g. if $M \vDash \alpha$, $\alpha \in \mathcal{L}(\mathrm{S}(M))$, then also $\varepsilon_{\mathrm{S}}(M) \vDash \alpha$).

Project

$$\pi_{\mathrm{S}}\colon \mathcal{M} \to \mathcal{M},\ \mathrm{S} \in \Sigma,\ \mathrm{S} \subseteq \mathrm{S}(M);$$
$$\pi_{\mathrm{S}}(M) = (\mathrm{S},\ \{\mathcal{I}|\mathrm{S}\colon \mathcal{I} \in \mathrm{W}(M)\}).$$ □

Projection reduces a signature of a given module. However, relationships between original concepts, roles, and individuals whose names remain in the signature are preserved (e.g. if $M \vDash \alpha$, $\alpha \in \mathcal{L}(\mathrm{S})$, then also $\pi_{\mathrm{S}}(M) \vDash \alpha$).

Rename

$$\rho_\gamma\colon \mathcal{M} \to \mathcal{M},\ \gamma \text{ is a signature mapping};$$
$$\rho_\gamma(M) = (\gamma(\mathrm{S}),\ \gamma(\mathrm{W})).$$ □

Renaming uses the notion of a *signature mapping*. Signature mapping γ is a triple: $(\gamma_C, \gamma_R, \gamma_I)$, each of them being a bijection from $\mathcal{N}$ to $\mathcal{N}$. By $\gamma(S)$ we mean $\gamma_C(C(S)) \uplus \gamma_R(R(S)) \uplus \gamma_I(I(S))$, and by $\gamma(\mathcal{I})$, where $\mathcal{I}$ is an S-interpretation, we mean an $\gamma(S)$-interpretation $\mathcal{I}'$ such that $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\gamma(X)^{\mathcal{I}'} = X^{\mathcal{I}}$ for every $X \in S$. Rename preserves relationships between concepts, roles, and individuals, however with respect to their name changes (e.g. if $M \vDash \alpha$, $\alpha \in \mathcal{L}(S(M))$, then $\rho_\gamma(M) \vDash \gamma(\alpha)$, where $\gamma(\alpha)$ is α transformed in such a way that all names in α have been systematically changed according to γ).

Select

$$\sigma_\alpha: M \to M, \alpha \in \mathcal{L}(S(M));$$
$$\sigma_\alpha(M) = (S, \{\mathcal{I} \in W(M): \mathcal{I} \vDash \alpha\}). \qquad \square$$

Selection leaves only these interpretations that are models of a sentence α. Obviously $\sigma_\alpha(M) \vDash \alpha$.

Union

$$\cup: M \times M \to M, S(M_1) = S(M_2);$$
$$M_1 \cup M_2 = (S(M_1), W(M_1) \cup W(M_2)). \qquad \square$$

Union performs a set-theoretic union of sets of models of conglomerates. The condition that $S(M_1) = S(M_2)$ is not very restrictive because we can easily upgrade this operation to a *generalized* union $\cup_g$: $M_1 \cup_g M_2 = \varepsilon_{S(M_2)}(M_1) \cup \varepsilon_{S(M_1)}(M_2)$.

Intersection

$$\cap: M \times M \to M, S(M_1) = S(M_2);$$
$$M_1 \cap M_2 = (S(M_1), W(M_1) \cap W(M_2)). \qquad \square$$

Difference

$$-: M \times M \to M, S(M_1) = S(M_2);$$
$$M_1 - M_2 = (S(M_1), W(M_1) - W(M_2)). \qquad \square$$

Intersection and difference are analogous to the union, and can be generalized in the similar way to the case when signatures of the operands differ.

Union and difference are non-linguistic (strictly semantic), i.e. their use may lead to generation of a conglomerate M for which there does not exist any corresponding set of sentences S in $\mathcal{L}$. This issue will be elaborated on later in the paper.

I-Join (intersecting join)

$$\times: M \times M \to M;$$
$$M_1 \times M_2 = (S', W');$$
$$S' = \gamma_1(S(M_1)) \cup \gamma_2(S(M_2));$$
$$W' = \{\mathcal{I} \in \mathbb{I}(S'): \quad \mathcal{I}|\gamma_1(S(M_1)) \in \gamma_1(W(M_1)) \wedge \mathcal{I}|\gamma_2(S(M_2)) \in \gamma_2(W(M_2))\}. \qquad \square$$

I-Join is an operation on two conglomerates. It uses two signature mappings γ_1, γ_2, each of them preceding every terminological name in a signature with a unique prefix. Thus, I-Join helps to solve potential naming conflict between conglomerates. I-Join preserves relationships between original concepts and roles in both modules (if $M_1 \models \alpha$, $\alpha \in \mathcal{L}(\mathrm{S}(M_1))$, then $M' \models \gamma_1(\alpha)$, analogically for M_2).

I-Join is a non-primitive operation, as $M_1 \times M_2$ can be expressed as $\rho_{\gamma_1}(M_1) \cap_g \rho_{\gamma_2}(M_2)$. This derivation justifies the name "intersecting join" as we may perceive this join as a "safe" way of intersecting modules.

U-Join (union join)

$$\begin{aligned}
&\bowtie: M \times M \to M;\\
&M_1 \bowtie M_2 = (\mathrm{S}', \mathrm{W}');\\
&\quad \mathrm{S}' = \gamma_1(\mathrm{S}(M_1)) \cup \gamma_2(\mathrm{S}(M_2));\\
&\quad \mathrm{W}' = \{\mathcal{I} \in \mathcal{I}(\mathrm{S}'): \quad \mathcal{I}|\gamma_1(\mathrm{S}(M_1)) \in \gamma_1(\mathrm{W}(M_1)) \vee\\
&\qquad \mathcal{I}|\gamma_2(\mathrm{S}(M_2)) \in \gamma_2(\mathrm{W}(M_2))\}. \qquad \square
\end{aligned}$$

U-Join is a counterpart of I-Join. γ_1, γ_2 have the same meaning as above. U-Join offers the "safe" way of performing union. Naturally, $M_1 \bowtie M_2 = \rho_{\gamma_1}(M_1) \cup_g \rho_{\gamma_2}(M_2)$.

Put-Under

$$\begin{aligned}
&\upsilon_C: M \times M \to M,\ C \in \mathcal{L}_C(\mathrm{S}(M_2));\\
&M_1\, \upsilon_C\, M_2 = (\mathrm{S}', \mathrm{W}');\\
&\quad \mathrm{S}' = \mathrm{S}(M_1) \cup \mathrm{S}(M_2);\\
&\quad \mathrm{W}' = \{\mathcal{I} \in \mathcal{I}(\mathrm{S}'): \quad \mathcal{I}|\mathrm{S}(M_2) \in \mathrm{W}(M_2) \wedge\\
&\qquad\qquad (\mathcal{I}|\mathrm{S}(M_1) \cap C^{\mathcal{I}}) \in \mathrm{W}(M_1)\}. \qquad \square
\end{aligned}$$

Put-Under correlates the domains of two conglomerates. We use here a *restriction* of an S-interpretation $\mathcal{I}$: by $\mathcal{I} \cap \Delta'$ we mean an interpretation $\mathcal{I}' = (\Delta', \cdot^{\mathcal{I}'})$ such that $X^{\mathcal{I}'} = X^{\mathcal{I}} \cap \Delta'$ for every $X \in \mathrm{S}$. As each conglomerate induces a set of laws that enforce certain relationships between concepts, roles, and individuals, then Put-Under can be perceived as a restriction of the scope of these laws to a fragment of a larger domain.

3.3 Examples

Example 3.1. (intersecting conglomerates). Consider two conglomerates: M_1 describes human resources and M_2 the structure of a hospital.

$$\begin{aligned}
M_1 &= M(\{\exists \mathit{isManagerIn}.\mathit{HTBusinessUnit} \sqsubseteq \mathit{Expert},\\
&\qquad \mathit{Expert} \sqsubseteq \mathit{Employee}\})\\
M_2 &= M(\{\mathit{leadsDepartment}(\mathit{johnSmith}, \mathit{neurosurgery}), \mathit{Department}(\mathit{neurosurgery})\}).
\end{aligned}$$

To merge the information from the two conglomerates in order to infer that *johnSmith* is an expert, we first create an intersection of the conglomerates: $M' = M_1 \cap_g M_2$, and then restrict the set of models by introducing additional "bridge" axioms: $M'' = M' \cap_g$

$M(\{leadsDepartment \sqsubseteq isManagerIn, Department \sqsubseteq HTBusinessUnit\})$. The last step can also be done by double selection. □

In the example we did not encounter any name conflict between conglomerates being merged. In general, such a conflict may occur and I-Join operator should be used. Below we show how to align two conglomerates in which the same set of terms is used to express different meanings.

Example 3.2. (joining conglomerates). Consider two conglomerates: M_1 and M_2. They contain assessment of several rooms for rent, and use the same categorization and signature S = $\{HSRoom, ASRoom, LSRoom\}$, where the concepts denote high, average and low standard rooms. But in M_1 and M_2 different criteria were used for categorization, as in the first case we were looking for a room to spend just one day and in the second case to stay for a longer period of time. We "import" the assessment from M_1 to M_2 performing necessary translation of classification between the conglomerates.

1. In the first step we simply I-Join the conglomerates (modules). As a result we obtain a conglomerate $M' = M_1 \times M_2$. The concepts have been renamed, so $S(M) = \{1{:}HSRoom, 2{:}HSRoom, 1{:}ASRoom, \ldots\}$.
2. Next, we make the criteria of assessment explicit. In this example we use only one criterion: a bathroom. So, we extend the signature of M' appropriately: $M'' = \varepsilon_{\{RoomWithBathroom\}}(M')$.
3. Afterwards, we bind the criteria with the assessment. In M_1 rooms with bathrooms were automatically considered high standard. According to the criteria used in M_2 no room with bathroom can be considered more than low standard.

   $M''' = M'' \cap_g M(\{RoomWithBathroom \sqsubseteq 1{:}HSRoom,$
   $\neg RoomWithBathroom \sqsubseteq 2{:}LSRoom\})$.

   Naturally, the second axiom is valid only if the domain consists of only rooms (which is assumed).
4. Finally we remove unwanted terms from the module signature: $M = \pi_{\{2:HSRoom,\ 2:ASRoom,\ 2:LSRoom\}}(M''')$.

In these steps all the translations possible to perform were done. All average or low standard rooms from M_1 were considered low standard in accordance with criteria from M_2. □

Some other aspects of using conglomerates can be found in [5].

4 Knowledge Query Language

The basis of our work on a query language for a DL knowledge base is our opinion that the development of a universal language for accessing knowledge bases must be based on assumptions similar to those adopted and practically proven in the case of SQL – beside the others, theoretical mathematical basis, language closure and availability of commands to create, manipulate and control the knowledge. Thus, we have based the KQL language on theoretical backgrounds presented in Sections 2 and 3, that is on contexts and conglomerates.

4.1 Basic Assumptions for KQL

The main statement for manipulating a conglomerate (called in KQL a conglomeration) is the SELECT statement of the following structure:

```
SELECT concept_list, role_list, attributes_list
         ADD axiom_list
         FROM conglomeration_expression
         WHERE concept_expression
         HAVING concept_expression
```

With respect to the conglomerates algebra, the SELECT clause corresponds to the projection—the choice of terms for a newly created conglomerate, the ADD clause corresponds to the selection—the contraction of the set of allowed interpretations, the WHERE and HAVING clauses correspond to the projection with respect to individual names. The difference between the WHERE and HAVING clauses lies in the fact that the WHERE clause selects those individuals that belong to a specified concept of the original conglomerate (as defined in the FROM clause), while the HAVING clause selects those individuals that belong to a specified concept of the target conglomerate. The query is conceptually executed in the following steps: (1) Determining the basic conglomerate on the basis of the FROM clause. (2) Reducing the individual names on the basis of the WHERE clause. (3) Extending the alphabet with new concepts / roles / attributes / individuals that occur in the statements contained in the ADD clause. (4) "Adding" (i.e. extending the ontology) to the conglomerate the statements from the ADD clause. (5) Projection of the alphabet only to the concepts / roles / attributes contained in the SELECT clause. (6) Reducing the individuals names basing on the HAVING clause.

A direct consequence of KQL closure is ability of query nesting. Every time a conglomerate is needed, one can use named conglomerates defined in knowledge base or conglomerates created on the fly by KQL expressions with the use of conglomerates algebra operators (such as INTERSECT, JOIN, UJOIN) or the SELECT clause. As a consequence of KQL closure and the ability of query nesting, the uniformity of modification and definition language has been achieved: one can use query nesting also in conglomerate creation statements:

```
CREATE CONGLOMERATION conglomeration_name
...
FROM conglomeration_expression
```

In KQL it is assumed that terminological queries are issued against a metaontology or a meta conglomerate. A metaontology is nothing but a single conglomerate knowledge base that stores information about conglomerates and rules. A meta conglomerate is also a single conglomerate knowledge base that stores information about exactly one conglomerate.

As a consequence of using the metaontology it is possible to issue terminological queries in a similar way how the assertional queries are issued; the only difference is that a query is directed to the metaontology rather than to the knowledge base itself. This follows from the fact that conglomerates, concepts, roles, attributes, and individuals from a knowledge base are reified to individuals in the metaontology and the relationships between them are reified to the corresponding binary relationships between individuals.

4.2 Examples of KQL Usage

The examples below have already been defined in terms of the conglomerate algebra.

Example 4.1. (simple import). We consider two conglomerates: M_1 describes human resources, and M_2 describes a structure of a hospital.

```
CREATE CONGLOMERATION M1
         ADD CONCEPT HTBusinessUnit
         ADD CONCEPT Expert
         ADD CONCEPT Employee
         ADD ROLE isManagerIn

CONSTRAIN M1
         EXIST isManagerIn HTBusinessUnit ISSUB Expert
         Expert ISSUB Employee

CREATE CONGLOMERATION M2
         ADD CONCEPT Department
         ADD ROLE leadsDepartment
         ADD INDIVIDUAL johnSmith
         ADD INDIVIDUAL neurosurgery

CONSTRAIN M2
         (johnSmith, neurosurgery) IS leadsDepartment
         neurosurgery IS Department
```

In KQL each conglomerate is created in two steps. In the first step the conglomerate signature is created (`CREATE CONGLOMERATION` statement). In the second step the constraints (sets of statements that must be satisfied) are added (`CONSTRAIN` statement). In the example above, the first conglomerate defines an employee and an expert. We assume that each expert is an employee, and anyone who manages at least one business unit is an expert. The second conglomerate describes John Smith who leads the department of neurosurgery. We want to ask whether `johnSmith` is the expert.

```
SELECT Expert
          ADD leadsDepartment ISSUB isManagerIn
          ADD Department ISSUB HTBusisnessUnit
          FROM M1 INTERSECT M2
          WHERE {johnSmith}
```

To merge the information from the two conglomerates in order to check whether johnSmith is an expert we first create an intersection of the conglomerates (FROM statement), and then restrict the set of model by introducing additional “bridge” axioms (ADD statement). The result of the query is a new conglomerate whose signature consists of two terms: > and `Expert`. In our example `johnSmith` is an instance of `Expert` concept. □

Example 4.2. (different versions and what-if problems) This example illustrates use of union and negation in KQL. Let us consider a conglomerate *M*:

```
CREATE CONGLOMERATION M
    ADD CONCEPT TrustedWitness    ADD CONCEPT CrimeScene
    ADD ROLE murdered
    ADD ROLE accuses
    ADD ROLE presentAt
    ADD INDIVIDUAL victim
CONSTRAIN CONGLOMERATION M
    Top ISSUB <= 1 INV (murdered) {victim}
    EXIST INV(accuses) TrustedWitness
       ISSUB EXIST INV(murdered) {victim}
    TrustedWitness ISSUB EXIST presentAt CrimeScene
```

The M conglomerate describes a world that assumes the only one murderer who is accused by a trusted witness (who has been present at the crime scene). We consider two (mutually exclusive) versions of facts (e.g. collected by two investigating agents: John Shady and Henry Brilliant). To achieve that we define two new conglomerates (one for each agent).

```
CREATE CONGLOMERATION M1
FROM(
          SELECT *
          ADD johnShady IS  TrustedWitness
          ADD (johnShady, tedInnocent) IS accuses
FROM M
)


CREATE CONGLOMERATION M2
FROM(
          SELECT *
          ADD henryBrillant IS TrustedWitness
```

```
            ADD (henryBrillant, markGuilty) IS accuses
            FROM M
)
```

Having such defined conglomerates M_1 i M_2 we would like to analyze different scenarios. Therefore we create a new conglomerate $M_0 = M_1$ UNION M_2.

We may assume that `henryBrillant` is not a trusted witness:

```
SELECT Murderer
            ADD NOT TrustedWitness(henryBrilliant)
            ADD Murderer EQ EXIST murdered {victim}
            FROM M1 UNION M2
```

We ask for a murderer. Therefore we define `Murderer` as a person who `murdered` a `victim`. The resulting conglomerate will return exactly one individual (`tedInnocent`) as an instance of `Murderer` concept.

We may assume that markGuilty is a murderer

```
SELECT TrustedWitness, NotTrustedWitness
            ADD (markGuilty, victim) IS murdered
            ADD NotTrustedWitness EQ NOT TrustedWitness
            FROM M1 UNION M2
```

In this case we can conclude that `henryBrillant` is a trusted witness (but this does not mean that `johnShady` is not a trusted witness). The conglomerate created within the query defines `NotTrustedWitness` concept (the complement of `TrustedWitness` concept defined in conglomerates M_1 and M_2). The conglomerate which is the result for this query will return a single instance of `TrustedWitness` concept and will return no instance of `NotTrustedWitnesss` concept.

We may assume that `johnShady` was not present at the crime scene

```
SELECT TrustedWitness, Murderer
            ADD johnShady IS EXIST presentAt CrimeScene
            ADD Murderer EQ EXIST murdered {victim}
            FROM M1 UNION M2
```

In this case we can conclude that `johnShady` is not a trusted witness and `markGuilty` is the murderer. To do this we define `Murderer` concept similarly to the second example. □

More examples of KQL usage can be found in [6].

5 The Architecture

The techniques described above can be composed to create a comprehensive and uniform approach to creating large federated knowledge bases. In such a base every component is treated as a sovereign entity (conglomerate), so that it is possible to exploit its contents in its full variety. While the entities are sovereign, they are not independent, but organized in hierarchical structure of contexts.

The SyNat project gives an opportunity to create such a federated base in a systematic way, and thus to validate the proposed approach. The notions of contexts, conglomerates, and the query language presented above are used in the following knowledge integration system (see Fig. 3). Data (pieces of knowledge) are stored in data (knowledge) sources[1] (at the left-hand side of the figure). They are "pushed" (pre-loaded statically) or "pulled" (loaded dynamically, on demand) into conglomerates (ontology modules) of a knowledge base. Contents of each source are perceived by the rest of the system as autonomous conglomerates.

The knowledge base is managed by a knowledge base management system called RKaSeA, the system developed at Gdansk University of Technology. The system, in its present stage of development, is equipped with the ability of handling the most important algebraic operations (intersection, projection, and selection), with the option of extending the range of operators by use of a specialized subsystem. The specialized subsystem takes advantage of $\mathcal{L}$-(2)-representation of conglomerates (described in details in [28]; in a nutshell it gives the possibility of representing conglomerates in the form of sets of sentences), which enables use of standard inference engines for selected reasoning tasks. Another subsystem of RKaSeA is Knowledge Layer [29], fully integrated (by use of MCA—Maximum Coverage Algorithm—for reasoning from both internally stored ABoxes and external knowledge) with the reasoning mechanism and allowing for attaching external knowledge sources as separate conglomerates. The external knowledge sources may also be NORs (non-ontological resources).

In the proposed architecture the pieces of knowledge are enriched with additional semantics. This enrichment is done by combining the knowledge from external modules with the selected fragments of integrated and contextualized Ontology of Science (Fig. 3). This name might be a bit misleading as we intend to include in this ontology also a choice of upper-level, commonly shared ontologies, primarily CIDOC [7] CRM. Such organization allows us to conduct an individualized method of integration of knowledge into the contextualized framework for each of the external conglomerates. It is also worth stressing that the system is flexible enough to easily accommodate new knowledge. Ontology of Science may contain "private" entities, added in personalized way by each user, such that use of them may improve the process of integration and tailor it for fulfillment of special needs of individual users and working groups of scientists.

[1] During the first phase of development knowledge sources from the Polish Federation of Digital Libraries (http://fbc.pionier.net.pl) will be exploited.

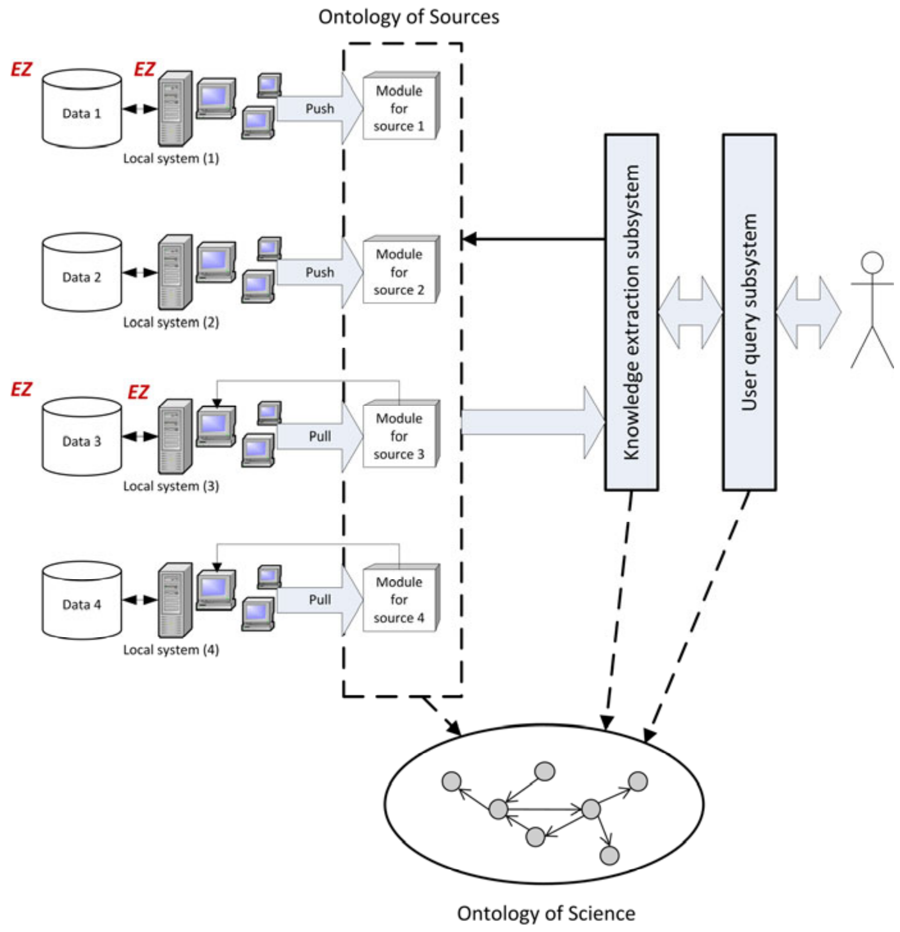

Fig. 3. The architecture of a knowledge integration system with contextualized knowledge bases divided into conglomerates and queried using KQL

Both external conglomerates and modules of Ontology of Science may be accessed by RKaSeA [8], that is able to interpret KQL queries issued by a user. The exact form of queries issued by the user is still subject to discussions. It is assumed though that the User Query Subsystem produces enough information to formulate KQL requests. Since the process of composing a query might be burdensome and difficult, we purport that it might be desirable to use some selected reasoning mechanisms to support it. This kind of reasoning is done over Ontology of Sources which holds meta-knowledge about the structure of the knowledge base.

The proposed general architecture forms a basis for two more detailed scenarios of use of the system. In the first scenario we assume that the contents of external modules is well-known and standardized to conform to selected upper-level ontologies and standards. In this scenario the primary role of the system focuses on selection of appropriate fragments of knowledge sources and on interpretation of queries issued by the users. In the second scenario we may not assume any initial level of conformance of the external source. In this scenario the system focuses on managing sources (with use of Ontology of Sources), selecting methods and procedures of integration, and maintaining uniform contextualized structure of the collection of managed external and internal conglomerates.

6 Related Work

In this section we present some related work concerning issues connected with integration of knowledge sources. In this section we describe two major approaches to modularization, and we also present some languages aimed at communication with knowledge bases.

We can distinguish between two main approaches in this field: an *inference approach* and an *algebraic approach*. The former addresses the question of how knowledge collected in one source affects logical consequences inferred from another source. The latter proposes algebraic methods to separate needed fragments from given sources and then join them in an appropriate way in order to find a proper answer for a given question.

The inference approach was firstly connected with the research field called *ontology merging* and then embraced issues addressed to the subject of *modularization* or *contextualization* of ontologies.

Modularization (contextualization) of ontologies is recently a domain of intensive research. It was the subject of two large European projects: Knowledge Web (http://knowledgeweb.semanticweb.org/semanticportal/sewView/frames.html) and NeOn (http://www.neon-project.org/). There exist a lot of motivations to apply a contextual approach to representing knowledge. The most significant are (according to the NeOn deliverable D 3.1.1 [10]):

1. Supporting different viewpoints.
2. Dealing with temporal information.
3. Dealing with inconsistent information.
4. Personalization.
5. Situation awareness in pervasive computing.
6. Scalability.
7. Ontology adaptation and views on ontologies.
8. Matching pairs (groups) of ontologies.

In the field of integration of knowledge from different and heterogeneous sources the results achieved during research on contextualization are very useful. Particularly the work on contextualization of logics is worth of mentioning. The examples of useful formalisms are, among others, *modal logic* ([11), *the logic of demonstratives* ([12]), and *context logics* ([13]). Obviously, all of them comply to the inference approach. In the environment of Semantic Web the most valuable is work connected with

MultiContext Systems/Local Model Semantics ([14][15]). The main idea of MCS/LMS relies on restricting the set of allowed interpretations of an ontology (called a *target ontology*) by a subset of conclusions drawn from another ontology (called a *source ontology*). Those conclusions are enabled via *bridge rules* which are the kind of inference rules defined particularly for the target ontology. MCS/LMS became a start point for the family of modularization methods that allow to create *distributed knowledge bases* consisting of a number of pairwise linked ontologies.

One of the most recognized methods is *Distributed Description Logic* (DDL) ([16]). In DDL, the idea of MCS/LMS is adopted to DL. In this idea the most important notion is a distributed interpretation—a set of local interpretations for the modules that are independent ontologies. The information is imported from a source module to a target module indirectly through bridge rules and individual correspondences, which are special bridge rules created for individual names. For a source ontology $\mathcal{K}_i$ and a target ontology $\mathcal{K}_j$ the rules have the form:

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

where $i{:}C$ and $i{:}a$ is respectively a concept or an individual defined in $\mathcal{K}_i$, and $j{:}D$ and $j{:}b$ is respectively a concept or an individual defined in $\mathcal{K}_j$.

Every module has its own local domain. In order to describe properly the semantics of the rules the relation between the two domains should be defined. The relation is called a domain relation and relates the two domains $r_{ij} \subseteq \Delta_i \times \Delta_j$, where Δ_i is the domain of $\mathcal{K}_i$, and Δ_j is the domain of $\mathcal{K}_j$. Usage of the domain relation r_{ij} allows to describe conditions under which a distributed interpretation satisfies a distributed knowledge base. If we denote the distributed interpretation $\mathcal{I} = \{\mathcal{I}_i\}_{i \in I}$ and the distributed knowledge base $\mathcal{K} = \{\mathcal{K}_i\}_{i \in I}$, we can say that $\mathcal{I}$ is a model of $\mathcal{K}$ iff:

$\mathcal{I}i$ satisfies $\mathcal{K}_i$

$r_{ij}(C^{\mathcal{I}i}) \subseteq D^{\mathcal{I}i}$ for all $i{:}C \xrightarrow{\sqsubseteq} j{:}D$

$r_{ij}(C^{\mathcal{I}i}) \supseteq D^{\mathcal{I}i}$ for all $i{:}C \xrightarrow{\sqsupseteq} j{:}D$

$b^{\mathcal{I}j} \in r_{ij}(a^{\mathcal{I}i})$ for all $i{:}a \longmapsto j{:}b$

where $C^{\mathcal{I}i}$ is a subset of Δ_i representing the concept C in the local interpretation $\mathcal{I}_i$, and $r_{ij}(C^{\mathcal{I}i})$ is a shortcut for $\{x | x \in \Delta_j \wedge \exists y \in C^{\mathcal{I}i} \wedge (x, y) \in r_{ij}\}$. Analogously, $r_{ij}(a^{\mathcal{I}i})$ means $\{x | x \in \Delta_j \wedge \exists y \in \Delta_i \wedge (x, y) \in r_{ij}\}$.

During reasoning over the target ontology the source ontology and the bridge rules should be taken into account. If, for example, the following bridge rules are defined:

$i{:}A \xrightarrow{\sqsubseteq} j{:}G$

$i{:}B \xrightarrow{\sqsupseteq} j{:}H$

and in $\mathcal{K}_i$ the axiom $B \sqsubseteq A$ exists, then, in $\mathcal{K}_j$, the axiom $H \sqsubseteq G$ holds.

Another technique is $\mathcal{E}$-Connections ([17]). The main semantic difference comparing to DDL is that domains of modules are disjoint, so a relation between them is not needed. The technique allows to define roles connecting individuals from different domains. These special roles are called *link relations*. All link relations are gathered in the set $\mathcal{E}$ consisting of sets $\mathcal{E}_{ij}$ of relations linking ontologies i and j. $\mathcal{E}$-Connections also defines the notion of distributed interpretation:

$$\mathcal{M} = \langle\ (\Delta_i)_{1 \leq i \leq n}\,, (\bullet^{\mathcal{M}i})_{1 \leq i \leq n}\,, (\bullet^{\mathcal{M}ij})_{1 \leq i,j \leq n}\ \rangle$$

where Δ_i is the domain of i-th ontology, for $i \neq j$ $\Delta_i \cap \Delta_j = \varnothing$, and interpretation functions maps each concept A of i-th ontology into $A^{\mathcal{M}i} \subseteq \Delta_i$, each role R into $R^{\mathcal{M}i} \subseteq \Delta_i \times \Delta_i$, each individual a into $a^{\mathcal{M}i} \in \Delta_i$, and each link relation $p \in \mathcal{E}_{ij}$ into $p^{\mathcal{M}ij} \subseteq \Delta_I \times \Delta_j$.

During creation of an ontology containing link relations it is possible to use special concept constructs $\exists p.Z$, $\forall p.Z$, $\leq np.Z$ or $\geq np.Z$, Z is a concept from j-th ontology, and $n \in N$. The semantics of such concepts is defined as follows:

$$(\exists p.Z)^{\mathcal{M}i} = \{a \in \Delta_i\colon \exists b \in \Delta_j\ ((a, b) \in p^{\mathcal{M}ij} \wedge b \in Z^{\mathcal{M}j})\},$$
$$(\forall p.Z)^{\mathcal{M}i} = \{a \in \Delta_i\colon \forall b \in \Delta_j\ ((a, b) \in p^{\mathcal{M}ij} \longrightarrow b \in Z^{\mathcal{M}j})\},$$
$$(\leq np.Z)^{\mathcal{M}i} = \{a \in \Delta_i\colon |\{b\colon ((a, b) \in p^{\mathcal{M}ij} \wedge b \in Z^{\mathcal{M}j})\}| \leq n\ \},$$
$$(\geq np.Z)^{\mathcal{M}i} = \{a \in \Delta_i\colon |\{b\colon ((a, b) \in \mathrm{p}^{\mathcal{M}ij} \wedge b \in Z^{\mathcal{M}j})\}| \geq n\ \}.$$

Usage of these concepts allows for reasoning in the terms of i-th ontology within j-th ontology. If, for example, in j-th ontology the axiom $D \sqsubseteq C$ is defined, and in i-th ontology there exist axioms $\exists p.C \sqsubseteq G$ and $H \sqsubseteq \exists p.D$, where $p \in \mathcal{E}_{ij}$, then in i-th ontology the axiom $H \sqsubseteq G$ holds.

The both presented methods try to solve the problems with contextual reasoning, i.e. how should interpretations of modules affect each other, how to preserve decidability, soundness and completeness. Although very important, results of this work do not help with the problem that we call the *problem of contextual designing*. They offer no tools that could allow to express why contexts were created and what is their purpose. In contrast to them, our proposal, the SIM method, is an example of a framework for designing *context-semantic knowledge bases*, i.e. knowledge bases where contexts are explicit elements of conceptualization and affect the semantics of another members of the model.

The second approach, the algebraic one, is based on the idea that all possible theories constitute a structure with the relationships between the elements describable by algebraic operation. In this approach we can distinguish two kinds of structures: *syntactic* and *semantic* ones.

A prominent example of an algebra for syntactic structures is the method described by Mitra in [18] (also by Mitra and Wiederhold in [19]). An ontology module is defined here as a set of sentences, namely RDF triples, so that every algebraic operation gives such a set as a result.

The simplest use case for the algebra is integration of knowledge from two ontologies. To show how this algebra works in practice, we follow here a slightly modified example from [19] and assume that the two ontologies being integrated, O_1 and O_2, describe respectively the domains of maritime vessels and cars (Fig. 4).

To enable conduction of any algebraic operation on the two ontologies, we firstly have to define special rules, called *articulation rules*, that explain the relationships between terms in the ontologies. For our example we define two such rules:

$\rightarrow$ *subClassOf*(O_1.*Motorboat*, *CombustionVehicle*)
$\rightarrow$ *subClassOf*(O_2.*Car*, *CombustionVehicle*)

These rules state that both a motorboat and a car are vehicles with a combustion engine. After defining the rules it is possible to perform binary operations on O_1 and O_2. There are three kinds of such operations: union, intersection and difference. Outcome of each operation is a set of sentences, defined as follows. We assume that $O_{1,2}$ denotes the RDF graph obtained by "executing" the *articulation rules* (treated as production rules). Then (see Fig. 4):

- the union of O_1 and O_2 is the union of graphs O_1, O_2 and $O_{1,2}$,
- the intersection of O_1 and O_2 is the graph $O_{1,2}$,
- the difference between O_1 and O_2 is the graph O_1 from which all nodes present in $O_{1,2}$ have been removed.

The described algebra is relatively straightforward and convenient for describing different points of view (or even contexts, as Mitra and Wiederhold argue in [19]). Nevertheless, its commitment to syntax can be a source of major difficulties with its use for practical, large ontologies. The outcome of algebraic operations relies heavily on the exact form of the sentences used in the modules being integrated. If the same semantic information is expressed with two different sentences, the result can easily convey the intention of the user. Moreover, to use the algebra one has to define a set of articulation rules. These sets constitute a new kind of extraontological objects that have their own syntax and particularities and need to be managed and maintained by a user, which can be cumbersome.

The aforementioned issues are not characteristic for the algebra by Mitra; on the contrary, they seem to be inherent for every algebra whose universe is based on syntactic form of sentences. This strongly suggests that semantic approach might be much more convenient for combining knowledge from different sources.

No approach known to the authors of this paper goes as deeply into semantic structures as conglomerate algebra. However, in the algebra proposed by participants of the NeOn project (deliverables D1.1.3 [20] and D1.1.4 [21]), a module is treated essentially (when we neglect its interfaces and version information) as a theory (i.e. a set of sentences augmented with all conclusions). This allows us to treat this method as a semantic one, because a theory can be identified with an appropriate set of models.

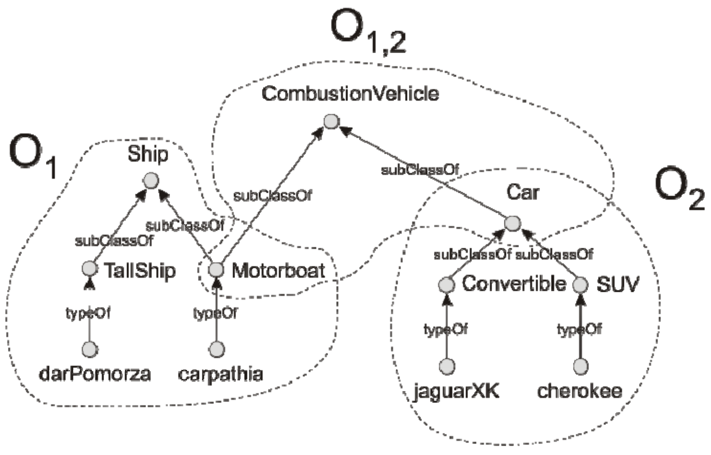

Fig. 4. An example of the aggregation conformance of denotation

In NeOn algebra there is no need for defining any special bridge axioms or rules. Instead, the algebra introduces several operations characterized in the terms of (semantic) entailment. Among them, there are the operations of union, intersection and difference (∪, ∩, –, resp.) defined as follows (α is any standard DL sentence):

$$M_1 \cup M_2 = M \text{ iff } \quad M \vDash \alpha \leftrightarrow M_1 \vDash \alpha \vee M_2 \vDash \alpha$$
$$M_1 \cap M_2 = M \text{ iff } \quad M \vDash \alpha \leftrightarrow M_1 \vDash \alpha \wedge M_2 \vDash \alpha$$
$$M_1 - M_2 = M \text{ iff } \quad M \vDash \alpha \leftrightarrow M_1 \vDash \alpha \wedge \neg (M_2 \vDash \alpha)$$

There is strong analogy between the above definitions and the conglomerate algebra. The union operation in NeOn and the intersection of conglomerates give the same outcome. Nevertheless, there are also important differences. The most notable one concerns difference between modules. NeOn definition seems to be faulty, because no module can possibly be a difference, which can be easily shown by putting for α any tautology.

While the definition of difference can probably be fixed, there is a discrepancy between NeOn and conglomerate algebra that is much deeper than a simple dissimilarity of definitions. Since NeOn module is a theory, it cannot exceed the expressivity of the used language. In conglomerate algebra it is not the case, as e.g. for union of $M(\{A \sqsubseteq B\})$ and $M(\{B \sqsubseteq A\})$ there is no corresponding set of sentences that expresses the result. While possibly this may be considered a drawback in some situations, this fact is also foundational for conglomerate algebra, as only through such assumption we may obtain an extension of Boolean algebra for semantic modules. Consequently, through this assumption, conglomerate algebra may constitute a basis for a query language that is equivalently strong as the relational algebra is for SQL.

In the last part of this section we refer to some languages created to communicate with knowledge bases. The most known and most widely applied is SPARQL [22], recommended by W3C. This is a query language designed for RDF repositories, thus it is not strictly addressed to ontologies. But, as OWL is built on top of RDF, it is often used for ontological purposes. As SPARQL is a query language, the syntax does not allow to enter changes into knowledge base.

There are many languages designed for ontologies. DIG ([23], [24]), proposed in University of Manchester, is a language defined for maintaining ontologies defined in DL. OWL-QL ([25]), designed for Stanford Knowledge Systems Laboratory is a continuation of DQL—DAML Query Language. nRQL [26] is the query language for RacerPro. SAIQL [27] is developed in University of Koblenz-Landau and University of Aberdeen during the works on the NeOn project. None of these languages is able to deal with modular knowledge bases. A query is always addressed to one ontology. None of them is closed: queries cannot be nested in themselves, although the last one—SAIQL—returns as an answer a set of sentences forming a fully working OWL DL ontology rather than just substitutions of variables.

7 Conclusions

This paper presents the architecture and methods for building large federated knowledge bases. As the primary criteria for selecting methods of knowledge representation in such knowledge bases an ontology organization, capabilities of inference for a selected representation and a set of characteristics of an internal query language were selected.

With respect to the ontology organization, we assumed that the adequate method of knowledge representation must provide:

1. An appropriate structure of ontological modules
   - with the ability to add new source modules and update *ontology of science*,
   - with the ability to organize knowledge at various levels of detail,
   - with the ability to dynamically create ontological modules when answering queries,
   - with the ability to dynamically locate a set of modules containing information that can affect the answer to the query.
2. Expressiveness of CIDOC CRM ($\mathcal{SHIN}$(D), version *Erlangen CRM*).
3. Capability of retrieving data from external sources using "push" and "pull" methods.
4. Capability of adding sources incompatible with CIDOC CRM (changing perspective) in a way that preserves possibility of metadata conversion and unification to CIDOC CRM-compatible form.

With respect to the inference capabilities, we assumed that the method of knowledge representation must provide:

5. Capability of inference from combined modules (including these combined *ad hoc*) with a well-defined semantics (*global semantics*) while still maintaining the semantics of each module (*local semantics*).
6. Capability of sound and complete inference from ontologies defined in description logic with minimal expressiveness of $\mathcal{ALCHI}$(D).
7. Capability of *Open World Assumption* aware inference.

With respect to the internal query language, we assumed that an adequate language must provide:

8. Capability of processing information expressed in description logic with minimal expressiveness of $\mathcal{ALCHI}$(D).
9. Capability of binding information contained in different modules of varying granularity.
10. Capability of transformation of assertional knowledge (ABox) to terminological knowledge (TBox) and vice versa.
11. Capability of nesting queries (i.e. using modules created ad hoc in further processing).

These requirements imply that the basic feature of the presented solution is its modularity, and a non-modular methods are only a theoretical basis for the modular methods. One of the key distinguishing factors is the division of modular methods to *semantic* and *non-semantic* ones. The non-semantic methods emphasizes on locality and compatibility problems associated strongly with the inference. Although these problems are not irrelevant with respect to the proposed solution (see requirement 5), the emphasis here is put on the dynamic and efficient management of modules from the point of view of contextual features, especially of detail and scope (requirements 1 and 5), but also perspective (requirement 4). This analysis led to exploiting in the presented solution the SIM method that allows automatic conclusions flow between modules (instances of contexts). In this way, knowledge bases built on the basis of the SIM method solve the detail and scope problem in a natural and effective way.

The conglomerate-based method used in our approach combines the features of semantic and non-semantic approach. With the latter it has in common the great emphasis put on the management of the global semantics of modules. However, the space of modules in the conglomerate-based method has an algebraic nature. The actual semantics of this space must be imposed by a designer of a knowledge base by different means of expressiveness, outside the scope of conglomerate-based method. Hence, the proposed combination of the SIM and conglomerate-based methods work smoothly together.

Although selecting a knowledge representation language is somehow independent on the choice of knowledge representation methods, it is easily seen that the choice of conglomerate knowledge base is tightly connected with KQL as the internal query language. KQL, as based on closed algebraic structure, is the only one that provides nesting queries in a natural way (requirement 11).

In conclusion we can state that none of the methods of knowledge representation is mature enough to meet all the requirements defined above. Using SIM and conglomerate-based methods together seems to simultaneously fulfill the requirements for the management of modules scope and details as well as perspective and offer a very flexible way to create a new ontological modules (by means of conglomerate algebra and KQL). So, next research and prototype implementation will be conducted in this direction.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: Description Logic Handbook. Cambridge University Press (2002)
2. OWL 2 Web Ontology Language Document Overview, W3C Recommendation, October 27 (2009), http://www.w3.org/TR/owl2-overview/
3. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, May 21 (2004), http://www.w3.org/Submission/SWRL/
4. Goczyła, K., Waloszek, A., Waloszek, W.: Contextualization of a DL knowledge base. In: Proceedings of the 20th International Workshop on Description DL 2007, Brixen/Bressanone, Italy, pp. 291–299 (2007)
5. Goczyła, K., Waloszek, A., Waloszek, W.: Algebra of ontology modules for semantic agents. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS (LNAI), vol. 5796, pp. 492–503. Springer, Heidelberg (2009)
6. Goczyła, K., Piotrowski, P., Waloszek, A., Waloszek, W., Zawadzka, T.: Terminological and Assertional Queries in KQL Knowledge Access Language. In: Pan, J.-S., Chen, S.-M., Nguyen, N.T. (eds.) ICCCI 2010. LNCS, vol. 6423, pp. 102–111. Springer, Heidelberg (2010)
7. Goczyła, K., Grabowska, T., Waloszek, W., Zawadzki, M.: The Knowledge Cartography – A New Approach to Reasoning over Description Logics Ontologies. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 293–302. Springer, Heidelberg (2006)
8. CIDOC Conceptual Reference Model (CRM), http://www.cidoc-crm.org/
9. Dublin Core Metadata Element Set, Version 1.1, http://dublincore.org/documents/dces/
10. Haase, P., Hitzler, P., Rudolph, S., Qi, G.: D3.1.1 Context Languages—State of the Art. The NeOn project deliverable (2006)
11. Garson, J.W.: Modal Logic for Philosophers. Cambridge University Press (2006)
12. Kaplan, D.: On the Logic of Demonstratives. Journal of Philosophical Logic (8), 81–98 (1978)
13. McCarthy, J.: Notes on Formalizing Context. In: Proceedings of IJCAI 1993, pp. 562–555. Morgan Kaufmann (1993)
14. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence 65(1), 29–70 (1994)
15. Ghidini, C., Giunchiglia, F.: Local model semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence 127(2), 221–259 (2001)
16. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. In: Spaccapietra, S., March, S., Aberer, K. (eds.) Journal on Data Semantics I. LNCS, vol. 2800, pp. 153–184. Springer, Heidelberg (2003)
17. Kutz, O., Wolter, F., Zakharyaschev, M.: Connecting abstract description systems. In: Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning 2002, pp. 215–226. Morgan Kaufmann (2002)
18. Mitra, P.: An Algebraic Framework for the Interoperation of Ontologies. PhD thesis, Stanford (2004)
19. Mitra, P., Wiederhold, G.: An Ontology-Composition Algebra. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 93–113. Springer, Heidelberg (2004)
20. d'Aquin, M.: D1.1.3 NeOn Formalisms for Modularization: Syntax, Semantics, Algebra. The NeOn project deliverable (2008)

21. d'Aquin, M.: D1.1.4 NeOn Formalisms for Modularization: Implementation and Evaluation. The NeOn project deliverable (2008)
22. Prud'hommeaux, E.: SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/
23. Bechhofer, S.: The DIG Descritption Logic Interface: DIG/1.1. University of Manchester (2003)
24. DIG 2.0: The DIG Description Logic Interface. DIG Working Group Note (September 2006), http://dig.cs.manchester.ac.uk/
25. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL—A Language for Deductive Query Answering on the Semantic Web. Knowledge Systems Laboratory, Stanford University, Stanford (2003)
26. RacerPro User's Guide, http://www.racer-systems.com/products/racerpro/manual.phtml
27. Kubias, A., Schenk, S., Staab, S.: SAIQL Query Engine - Querying OWL Theories for Ontology Extraction. In: OWLED 2007 OWL: Experiences and Directions Third International Workshop (2007)
28. Goczyła, K., Waloszek, A., Waloszek, W.: A Semantic Algebra for Modularized Description Logics Knowledge Bases. In: Grau, B.C. (ed.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford (2009)
29. Goczyła, K., Zawadzka, T., Zawadzki, M.: Managing Data from Heterogeneous data Sources using Knowledge Layer, Software Engineering Techiques: Design for Quality. In: Sacha, K. (red.) IFIP International Federation for Information Processing. 227, pp. 300–312. Springer, Boston (2006)