OWL as Yet Another Data Model to be Integrated

Leonid Kalinichenko[1], Sergey Stupnikov[1]

[1] Institute of Informatics Problems, Russian Academy of Sciences
Vavilov st., 44 bldg. 2, 119333, Moscow, Russia
{leonidk, ssa}@ipi.ac.ru

Abstract. The paper[1] argues against cultivation in the ontological community of the opinion that ontologies are at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level. The paper claims that rather it would be right to consider OWL as yet another data model to be integrated with other heterogeneous information models. Applying the SYNTHESIS – an extensible language for heterogeneous information resource integration and mediator definition – we show how a sound mapping of a conceptual schemas expressed in OWL 2 QL into SYNTHESIS schemas can be defined. The soundness of the mapping is justified. The paper shows also how the integration of the OWL-defined databases into a SYNTHESIS-based mediator can be provided.

Keywords: OWL 2 QL, SYNTHESIS, conceptual schema, mediator, database integration, data model mapping.

1 Introduction

During last years the expansion of ontologies into the area of databases and information systems becomes more and more noticeable. In the beginning of 90ies in connection with the development of approaches for interoperability of the information system components the ontologies were identified as essential constituents of the semantically interoperable systems and were defined as a "formal, explicit specification of a shared conceptualization" [7]. By 2010 [8] the ontologies evolved to be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, intended for modeling knowledge about individuals. Ontologies are said to be at the "semantic" level, whereas database schemas are models of data at the "logical" or "physical" level. Due to such positioning, there is an intention to use ontologies for integrating heterogeneous databases, enabling interoperability among disparate systems.

Such expansion is based on a simple idea – to apply ontology as a (database) conceptual schema. In a number of Web publications "ontology" is identified literally

[1] This research has been done under the support of the RFBR (projects 10-07- 00342-a, 11-07-00402-a) and the Program for fundamental research of the Presidium of RAS № 15П (project 4.2).

with “conceptual schema”: “In computer science, an ontology is the attempt to formulate an exhaustive and rigorous conceptual schema within a given domain, a typically hierarchical data structure containing all the relevant entities and their relationships and rules within that domain”. This activity is accompanied with the introduction of new terms, such as Ontology Based Information Systems (OBIS), Ontology Based Data Access (OBDA), Ontology Based Data Integration (OBDI) [4]. The respective “ontology based technologies” assume usage of databases on the basis of application domains conceptualization and relational database access mediated with the conceptual view over the data. QuOnto system [1, 16] is a well-known example of implementation of such approach. QuOnto supports reasoning over ontologies expressed in a description logic of DL-Lite family [5] and answering union of conjunctive queries over the external relational DBMS (such as Oracle, DB2, SQL Server, MySQL, etc.). It is worth noting that before its execution, the query is deductively expanded applying axioms of the conceptual schema and the inference engine in the respective description logic. It is clearly seen that the researches in the area of the ”ontology based” information systems are focused on the development of the ontological languages for conceptual modeling, that is on the development of conceptual data models based on description logics. The most visible results of such works are reflected in the OWL 2 QL profile [6] based on the DL-Lite$_R$ logic.

In this paper we claim that in the context of databases and information systems OWL, chosen in the paper as a W3C recommendation and widely used ontological language [15], might be considered as yet another data model to be integrated with another heterogeneous information resources in frame of more expressive languages. In the paper we show that “ontology based” conceptual models can be mapped with preserving of their semantics into the existing mediation oriented data models used for heterogeneous database integration. This approach is demonstrated applying the combined frame-based and object-oriented language SYNTHESIS [14] intended for canonical information modeling and mediator definition, supported also by methods and tools for problem solving in application-driven distributed heterogeneous information environment [2]. It will be shown in this paper how the database conceptual schemas defined as OWL 2 ontologies can be mapped with preserving of their semantics into the mediation schemas defined in the SYNTHESIS language to be integrated with other heterogeneous information resources. The aim of this paper is to demonstrate a soundness of the mapping for limited subsets of the languages. For this purpose it is sufficient to use for interpretation of SYNTHESIS schemas the same structures as used for interpretation of OWL schemas [9]. Using such structures allows to simplify significantly the proof of soundness and to save space. Also we do not provide here an exhaustive mapping and proof of soundness but illustrate them by examples to save space.

Another approach, including into the functionality of the mediation data model the capabilities of OWL directly, consists in extending the mediation data model with the dependencies introduced recently in the Datalog$_0^{\pm}$ [3]. Datalog$_0^{\pm}$ generalizes the DL-Lite logic family preserving tractability and provides a translation from DL-Lite into Datalog$_0^{\pm}$. To compare approaches in Section 3 we provide a respective Datalog$_0^{\pm}$ construct for every pair of OWL and SYNTHESIS constructs mapped to each other.

The paper is structured as follows. The next section provides a brief introduction into the SYNTHESIS data model developed as an extensible language for information

resource integration. In section 3 a mapping of OWL 2 QL into SYNTHESIS is defined and compared with the mapping of DL-Lite into $Datalog_0^{\pm}$. Section 4 justifies the soundness of the mapping between schemas expressed in OWL 2 and SYNTHESIS. Section 5 shows how the integration of the OWL-defined databases into a SYNTHESIS-based mediator can be provided.

2 SYNTHESIS Data Model as an Extensible Language for Information Resource Integration

The SYNTHESIS language [14] was motivated by the integration of heterogeneous information resources and particularly by the *subject mediation* approach [11] application for various subject domains. The approach is significantly based on so called "canonical" model – the model used for uniform representation of various information models in one paradigm. The model intends to provide for uniform (canonical) representation of mediator and heterogeneous resources. The SYNTHESIS language facilities determine the canonical information model kernel that should be sufficiently rich for refining mapping into it of various models used for heterogeneous information resources support [10, 13]. The main principle of canonical model synthesis consists in its extensibility providing for preserving various data model semantics in the refining mappings.

Subject mediators support the process of systematic registration and classification of resources providing the uniform ontological knowledge and metainformation for discovery and composition of information resources relevant to mediator [11].

A process of registration of heterogeneous information resources in a subject mediator is mostly based on Local As View (LAV) approach [12]. According to LAV the schemas of resources being registered are considered as materialized views over virtual classes of a mediator. Such registration technique provides for stability of application specification during any modifications of specific information resources and of their actual presence (their removing, adding new ones, etc.) as well as for scalability of mediators w.r.t. the number of resources registered in them.

The main features of the SYNTHESIS language are briefly outlined in the remainder of the section. The SYNTHESIS language is based on a combined semi-structured and object data model.

Semi-structured (self-defined) data are represented by *frames* [14]. Frames are used for the definition of any entity of the language including metadefinitions of other facilities of the language such as type and class specifications, function and assertion specifications and so on. A frame can be considered as a set of attributes called *slots*. Every slot can have several values. An additional metainformation can be associated with frames, slots and values. It also takes a form of a frame called a metaframe, a metaslot or a metavalue respectively.

A basic unit of specification in the SYNTHESIS language is a *module.* A module may specify a generalized representation of an information source or a subject domain or a conceptual project of an information system. Structured data correspond to *abstract data types* (ADT) specifying a state and a behavior of their instances in terms of type attributes and type methods. Type specifications can contain type invariants –

constraints expressed by closed logic formulae. The language also contains a rich collection of built-in datatypes. A set of abstract data types with a subtype relation forms a lattice. Subtype relation assumes that subtype invariants imply invariants of a supertype and pre- (post-) condition of a method of a supertype is relaxed (strengthened) by pre- (post-) condition of the respective method of a subtype.

Collections of uniformly structured objects are represented by *classes*. Type and class specifications are explicitly separated to underline a role of a class as a collection of objects and a role of a type as a specification of state and behavior of class instances (objects). Class specifications can also contain invariants. Collections can be associated by generalization/specialization (class/subclass) relationship. A superclass contains all instances of a subclass, instance type of a subclass is a subtype of an instance type of a superclass. Object attributes are considered in their turn as objects of *association metaclasses* establishing relationships among objects from association domain and objects from association range.

Formulae are specified applying a typed variant of the first order predicate logic.

3 A Mapping of the OWL 2 QL into the SYNTHESIS

This section is not aimed to formally define an exhaustive mapping of the OWL 2 QL into the SYNTHESIS. To save space mappings for typical constructs of OWL 2 QL are defined by examples. Note that OWL specifications are represented in XML-free *functional-style syntax*.

Schema and class mapping. An OWL schema (for instance, *Hermitage* – a schema describing Hermitage museum digital collection) is mapped into a *module* of the SYNTHESIS language. A class (for instance, *artist*) is mapped into a class and its instance type of the SYNTHESIS language.

OWL	SYNTHESIS
Ontology(Hermitage …)	{ Hermitage; in: module; … }
Declaration(Class(artist))	{ artist; in: class; instance_section: Artist; … } { Artist; in: type; … }

Object property mapping. Object properties are represented in the OWL with the help of *ObjectProperty* construct (if a range of a property is a class, for instance, the *artworks* property of the *artist* class) or *DataProperty* construct (if a range of a property is a built-in datatype).

OWL	SYNTHESIS	$Datalog_0^{\pm}$
Declaration(ObjectProperty(artworks)) ObjectPropertyDomain(artworks artist) ObjectPropertyRange(artworks entry) InverseObjectProperties(artworks author)	{ Artworks; in: association, metaclass; instance_section: { domain: artist; range: entry; }; } { Artist; in: type; artworks: {set; type_of_element: Entry;}; metaslot in: Artworks; inverse: Entry.author; end }	artworks (X, Y) → artist(X), entry(Y). artworks(X, Y) → author(Y, X). author(X, Y) → artworks(Y, X).

In the SYNTHESIS language object properties are represented by association metaclasses (for instance, the *Artworks* metaclass) establishing relationships between

objects from association domain (*artist* class) and objects from association range (*entry* class). An attribute of an ADT (for instance, the *artworks* attribute of the *Artist* type) is specified belonging to respective association metaclass. Inverse properties (for instance, *author*) are represented by *inverse* slot in the metaslot of a respective property (*artworks*).

Class relationship mapping. Simple cases of subclass relationship (for instance, *drawing* is a subclass of *entry*) are represented in SYNTHESIS in the *superclass* slot of the respective subclass. More complicated cases (for instance, using *ObjectSomeValuesFrom*, *ObjectIntersectionOf*, *ObjectComplementOf*) described in OWL by class axioms, are represented in SYNTHESIS by means of class invariants (*osvf, sco*).

OWL	SYNTHESIS	$Datalog_0^{\pm}$
SubClassOf(drawing entry) SubClassOf(sculptor ObjectSomeValuesFrom(artworks sculpture)) SubClassOf(purePainter ObjectIntersectionOf(painter ObjectComplementOf(sculptor)))	{ drawing; in: class; superclass: entry; } { sculptor; in: class; instance_section: { osvf: { in: invariant; all s/Sculptor (is_in(sculptor, s) -> ex sc/Sculpture(is_in(sculpture, sc) & is_in(s.artworks, sc)) } } } { purePainter; in: class; instance_section: { sco: { in: invariant; purePainter <= intersect(painter, differ(thing, sculptor))} } }	drawing (X) → entry(X). sculptor(X) → ∃Y(artworks(X, Y), sculpture(Y)). purePainter(X) → painter(X), ¬ sculptor(X).

Here *all* means universal quantifier, *ex* means existential quantifier, *s/Sculptor* is a typed variable, *is_in* predicate means a set membership, –> sign means logical implication, <= means a subset relation, *intersect* and *differ* mean set intersection and difference respectively.

Reflexive, irreflexive, symmetric and asymmetric object properties mapping. To specify these special kinds of properties respective association metaclasses are defined in the SYNTHESIS language. For instance, to specify reflexive properties the *reflexive* association metaclass is defined:

```
{ reflexive; in: association, metaclass;
  instance_section: { reflexivity: { in: invariant;
    all x ((is_in(this.domain, x)->is_in(this, [x, x])) &
          (is_in(this.range, x)->is_in(this, [x, x])))
} } }
```

Keyword *this* here means any association class belonging to *reflexive* metaclass, *domain* means association class domain, *range* means association class range, *[a, b]* means a pair of objects related by association. A metaclass of every reflexive property becomes a subclass of *reflexive* (for instance, *SameRoom*).

OWL	SYNTHESIS	$Datalog_0^{\pm}$
ReflexiveObjectProperty(sameRoom)	{ SameRoom; in: association, metaclass; superclass: reflexive; }	room(X) → sameRoom(X, X).

Irreflexive, symmetric and asymmetric properties are mapped similarly.

Fact mapping. Facts (assertions defining objects' states) are represented in OWL by means of *NamedIndividual, ClassAssertion, DataPropertyAssertion,*

ObjectPropertyAssertion constructs. The respective assertions are represented in SYNTHESIS by means of object type constants.

OWL	SYNTHESIS	$Datalog_0^{\pm}$
Declaration(NamedIndividual(TheLittaMadonna)) ClassAssertion(painting TheLittaMadonna) DataPropertyAssertion(author TheLittaMadonna “Leonardo da Vinci”) ObjectPropertyAssertion(inRoom TheLittaMadonna TheLeonardoDaVinciRoom)	{ TheLittaMadonna; in: painting; author: “Leonardo da Vinci”; inRoom: TheLeonardoDaVinciRoom; }	painting(TheLittaMadonna). author(TheLittaMadonna, “Leonardo da Vinci”). inRoom(TheLittaMadonna, TheLeonardoDaVinciRoom).

Subproperties mapping. An example of a subproperty (*paintings*) of a property (*artworks*) is shown below. In SYNTHESIS the subproperty is represented by a specific invariant of the *Painting* association metaclass.

OWL	SYNTHESIS	$Datalog_0^{\pm}$
SubObjectPropertyOf(paintings artworks)	all c, p (is_in(this, [c, p]) -> is_in(c.artworks, p))	paintings(X, Y) → artworks(X, Y)

Disjoint properties and disjoint classes mapping. Examples of disjoint properties (*birthDate* and *deathDate*) of the class *artist* and disjoint classes are shown below.

OWL	SYNTHESIS	$Datalog_0^{\pm}$
DisjointDataProperties(birthDate deathDate)	all a/Artist(is_in(artist, a) -> a.birthDate <> a.deathDate)	birthDate(X, Y), deathDate(X, Y) → ⊥
DisjointClasses(freestanding relief)	intersect(freestanding, relief) = {}	freestanding(X), relief(X) → ⊥

In the SYNTHESIS language disjoint properties and disjoint classes are represented by the respective class invariant. Here {} denotes empty set.

4 Soundness of the Mapping of the OWL into the SYNTHESIS

Denote as Σ: $S_{OWL} \times S_{SYN}$ arbitrary mapping from a set S_{OWL} of schemas represented in OWL to a set S_{SYN} of schemas represented in the SYNTHESIS language. Formally this mapping is a relation – a set of pairs of OWL and SYNTHESIS well formed schemas. Schema $o \in S_{OWL}$ is mapped into schema $s \in S_{SYN}$ iff $\langle o, s \rangle \in \Sigma$.

Let Σ mapping be *sound* if for all pairs $\langle o, s \rangle$ of schemas $o \in S_{OWL}$ and $s \in S_{SYN}$ such that $\langle o, s \rangle \in \Sigma$ the following condition holds: for any interpretation I_o of o such that $I_o \models o$ an equivalent interpretation I_s of s exists such that $I_s \models s$ and vice versa. Here $I_o \models o$ means I_o *satisfies* o [9].

Denote as σ the mapping illustrated by the examples in the section 3. The mapping σ is a generalization of the examples for arbitrary schemas of OWL and SYNTHESIS. The aim of this section is to show that the mapping σ is sound. As the first step to achieve this goal the semantic structures interpreting schemas of the OWL and SYNTHESIS language are to be provided. The interpretations of OWL are defined in [9]. The SYNTHESIS language is more expressive than OWL and semantic structures required to interpret it are more complicated. SYNTHESIS

possesses different vocabulary and datatypes. To prove soundness of an exhaustive mapping of OWL into SYNTHESIS it is required to relate OWL and SYNTHESIS semantic structures. The aim of this paper is to demonstrate a soundness of the mapping for limited subsets of the languages. For this purpose it is sufficient to use for interpretation of SYNTHESIS schemas the same structures as used for interpretation of OWL schemas [9] and neglect differences between vocabularies and datatypes. Using such structures allows to simplify significantly the proof of soundness and to save space.

Consider an interpretation $I = \langle \Delta_I, \Delta_D, \bullet^C, \bullet^{OP}, \bullet^{DP}, \bullet^I, \bullet^{DT}, \bullet^{LT} \rangle$ as a tuple of the following components:

- Δ_I – *object domain;*
- Δ_D – *data domain*;
- $\bullet^C$ – *class interpretation function* that assigns to a class *Cl* a subset $(Cl)^C \subseteq \Delta_I$;
- $\bullet^{OP}$ – *object property interpretation function* that assigns to each object property *op* a subset $(op)^{OP} \subseteq \Delta^I \times \Delta^I$;
- $\bullet^{DP}$ – *data property interpretation function* that assigns to each data property *dp* a subset $(dp)^{DP} \subseteq \Delta_I \times \Delta_D$;
- $\bullet^I$ – *individual interpretation function* that assigns to each individual *o* an element $(o)^I \in \Delta_I$;
- $\bullet^{DT}$ – *datatype interpretation function* that assigns to each datatype *dt* a subset $(dt)^{DT} \subseteq \Delta_D$;
- $\bullet^{LT}$ – *literal interpretation function* that that assigns to each literal *l* of a datatype *dt* an element $(l)^{LT} \in (dt)^{DT}$.

An interpretation *I satisfies* an OWL schema *O* if a set of specific conditions related to axioms constituting the schema is satisfied [9]. To define that an interpretation *I satisfies* a SYNTHESIS schema *S* the conditions related to syntactic constructs of the SYNTHESIS language are provided in tables below. To simplify reasoning the conditions are provided only for constructs of the language shown in the previous section. For a syntactic construct a respective semantic condition in terms of SYNTHESIS schema vocabulary and interpretation *I* is provided.

Notice semantic differences between OWL, Datalog$_0^{\pm}$ and SYNTHESIS. In OWL an interpretation *I* is a *model* of a specification *o* if an interpretation *J* exists such that *J* coincides with *I* on all named individuals and *J satisfies o* [9]. This gap between *I* and *J* is filled in by the OWL and Datalog$_0^{\pm}$ systems with the help of inference rules (tuple generating dependencies). In SYNTHESIS *I* is a model of a schema *s* if *I* satisfies *s*. But for the purpose of resource integration in mediators with SYNTHESIS as a canonical model the OWL ontologies are just resources to be queried and inference is carried out inside of such resources [2]. For the mediator the interpretation *I* together with TGDs is equivalent to the interpretation *J* containing all inferred anonymous individuals. So to verify the mapping σ it is sufficient to consider only interpretations satisfying OWL schemas and not general OWL models. Notice also that SYNTHESIS and Datalog$_0^{\pm}$ constructs shown in the tables of the section 3 are not semantically equivalent in cases containing TGDs, they are both equivalent to an original OWL construct but with different semantics. The reason is explained by different purposes of the OWL mappings: the mapping to Datalog$_0^{\pm}$ is intended to support efficient ontological reasoning equivalent to OWL QL and its inference and

the mapping to SYNTHESIS is intended to implement integration of OWL resources in mediators.

Conditions for object properties. Binding a class (*artist*) with its instance type (*Artist*) imposes respective constraints on domain and range of attributes constituting the type. If an attribute association metaclass is a subclass of some special metaclass (for instance, metaclass *SameRoom* of the attribute *sameRoom* is a subclass of the *reflexive* metaclass) then the attribute has to satisfy all invariants of the superclass (the *sameSalary* property has to be reflexive).

SYNTHESIS construct	Condition
{ artist; in: class; instance_section: Artist; … } { Artist; in: type; artworks: {set; type_of_element: Entry;}; metaslot in: Artworks; inverse: Entry.author; end } { Artworks; in: association, metaclass; instance_section: {domain: artist; range: entry; }; }	$\forall a, e\ (\ (a, e) \in (artworks)^{OP} \rightarrow a \in (artist)^{C} \wedge e \in (entry)^{C}\ )$ $\forall a, e\ (\ (a, e) \in (artworks)^{OP} \leftrightarrow (e, a) \in (author)^{OP}\ )$
{ SameRoom; in: association, metaclass; superclass: reflexive; instance_section: { domain: entry; range: entry; }; } { Entry; in: type; sameRoom: {set; type_of_element: Entry;}; metaslot in: SameRoom end }	$\forall e1, e2\ (\ (e1, e2) \in (sameRoom)^{OP} \rightarrow e1 \in (entry)^{C} \wedge e2 \in (entry)^{C}\ )$ $\forall e\ (\ e \in (entry)^{C} \rightarrow (e, e) \in (sameRoom)^{OP})$

Conditions for facts.

SYNTHESIS construct	Condition
{ TheLittaMadonna; in: painting; author: “Leonardo da Vinci”; inRoom: TheLeonardoDaVinciRoom; }	$(TheLittaMadonna)^{I} \in (painting)^{C} \wedge ((TheLittaMadonna)^{I}, (\text{“}Leonardo\ da\ Vinci\text{”})^{LT}) \in (author)^{DP} \wedge ((TheLittaMadonna)^{I}, (TheLeonardoDaVinciRoom)^{I}) \in (inRoom)^{OP}$

Conditions for subclasses.

SYNTHESIS construct	Condition
{ drawing; in: class; superclass: entry; }	$(drawing)^{C} \subseteq (entry)^{C}$

Conditions for invariants.

SYNTHESIS construct	Condition
all s/Sculptor (is_in(sculptor, s) -> ex sc/Sculpture(is_in(sculpture, sc) & is_in(s.artworks, sc))	$\forall s\ (\ (s)^{I} \in (sculptor)^{C} \rightarrow \exists sc\ (\ (sc)^{I} \in (sculpture)^{C}\ ) \wedge ((s)^{I}, (sc)^{I}) \in (artworks)^{OP}\ )$
purePainter <= intersect(painter, differ(thing, sculptor))	$(purePainter)^{C} \subseteq (painter)^{C} \cap (\Delta_{I} \setminus (sculptor)^{C})$
a.birthDate <> a.deathDate	$\forall b, d\ (\ (a, b) \in (birthDate)^{DP} \wedge (a, d) \in (deathDate)^{DP} \rightarrow b \neq d\ )$

As far as schemas of OWL and SYNTHESIS are interpreted by the same semantic structures it is possible to consider a common interpretation I for both schemas $s \in S_{SYN}$ and $o \in S_{OWL}$ mapped to each other by σ. To prove that σ is sound it is sufficient to show that I satisfies s iff I satisfies o. So an equivalence of conditions of satisfying a schema in OWL [9] and conditions of satisfying a schema in SYNTHESIS (demonstrated earlier in this section) is to be shown.

Generally this equivalence is to be proved by induction over elements constituting a schema (classes, properties, facts, axioms). In this section the equivalence is demonstrated only by examples from the previous section. Thus main points of the proof are illustrated. Left column of the tables contains conditions for OWL constructs (considered in the previous section) according to the semantics of OWL [9]. Right column of the tables contains conditions (already considered in this section) for respective SYNTHESIS constructs.

Conditions for object properties.

Conditions for OWL constructs	Conditions for SYNTHESIS constructs
$\forall x, y\ (\ (x, y) \in (artworks)^{OP} \rightarrow x \in (artist)^{C})$ $\forall x, y\ (\ (x, y) \in (artworks)^{OP} \rightarrow y \in (entry)^{C})$ $(artworks)^{OP} = \{\ (x, y) \mid (y, x) \in (author)^{OP}\}$	$\forall a, e\ (\ (a, e) \in (artworks)^{OP} \rightarrow$ $a \in (artist)^{C} \wedge e \in (entry)^{C})$ $\forall a, e\ (\ (a, e) \in (artworks)^{OP} \leftrightarrow$ $(e, a) \in (author)^{OP})$

Conditions for relationships among classes.

Conditions for OWL constructs	Conditions for SYNTHESIS constructs
$(drawing)^{C} \subseteq (entry)^{C}$	$(drawing)^{C} \subseteq (entry)^{C}$
$(purePainter)^{C} \subseteq$ $(painter)^{C} \cap (\Delta_I \setminus (sculptor)^{C})$	$(purePainter)^{C} \subseteq$ $(painter)^{C} \cap (\Delta_I \setminus (sculptor)^{C})$
$(sculptor)^{C} \subseteq \{\ x \mid \exists y\ ((x, y) \in (artworks)^{OP} \wedge$ $y \in (sculpture)^{C})\ \}$	$\forall s\ (\ (s)^{I} \in (sculptor)^{C} \rightarrow \exists sc\ (\ (sc)^{I} \in$ $(sculpture)^{C}\ ) \wedge ((s)^{I}, (sc)^{I}) \in (artworks)^{OP}\ )$

Conditions for facts.

Conditions for OWL constructs	Conditions for SYNTHESIS constructs
$(TheLittaMadonna)^{I} \in (painting)^{C}$ $((TheLittaMadonna)^{I},$ $(\text{“Leonardo da Vinci”})^{LT}) \in (author)^{DP}$ $((TheLittaMadonna)^{I},$ $(TheLeonardoDaVinciRoom)^{I}) \in$ $(inRoom)^{OP}$	$(TheLittaMadonna)^{I} \in (painting)^{C} \wedge$ $((TheLittaMadonna)^{I},$ $(\text{“Leonardo da Vinci”})^{LT}) \in (author)^{DP} \wedge$ $((TheLittaMadonna)^{I},$ $(TheLeonardoDaVinciRoom)^{I}) \in$ $(inRoom)^{OP}$

Conditions for reflexive properties.

Conditions for OWL constructs	Conditions for SYNTHESIS constructs
$(paintings)^{OP} \subseteq (artworks)^{OP}$	$\forall c, p\ ((c, p) \in (paintings)^{OP} \rightarrow$ $(c, p) \in (artworks)^{OP})$

Conditions for subproperties.

Conditions for OWL constructs	Conditions for SYNTHESIS constructs
$\forall x\ (x \in \Delta_I \rightarrow (x, x) \in (sameRoom)^{OP})$	$\forall e\ (e \in (entry)^{C} \rightarrow (e, e) \in (sameRoom)^{OP})$

Conditions for disjoint properties and disjoint classes.

Conditions for OWL constructs	Conditions for SYNTHESIS constructs
$(birthDate)^{DP} \cap (deathDate)^{DP} = \varnothing$	$\forall a\ (a \in (artist)^{C} \rightarrow$ $\forall b, d\ (\ (a, b) \in (birthDate)^{DP} \wedge$ $(a, d) \in (deathDate)^{DP} \rightarrow b \neq d\ ))$
$(freestanding)^{C} \cap (relief)^{C} = \varnothing$	$(freestanding)^{C} \cap (relief)^{C} = \varnothing$

It is easy to see that conjunction of conditions for OWL and conjunction of conditions for SYNTHESIS are equivalent. In simplest cases they are the same (for instance, some conditions for relationships among classes) or distinguished only by variable names. This allows to conclude that the mapping σ illustrated in the previous section is sound.

5 Integration of the OWL resources into a Subject Mediator

In this section a cultural heritage domain for a subject mediator example is considered. A limited subset of a respective mediated schema is provided in the table 1. The more detailed schema of a subject mediator for the cultural heritage domain is provided in [12].

Table 1. Mediator schema example

CulturalHeritage; in: module;	
type: { Person; in: type; name: string; }, { Creator; in: type; supertype: Person; culture: string; works: {set; type_of_element: Heritage_Entity;}; }, { Heritage_Entity; in: type; supertype: Entity; title: string; created_by: Creator; place_of_origin: Address; date_of_origin: time; in_collection: Collection; }, { Painting; in: type; supertype: Heritage_Entity; dimensions: {sequence; type_of_element: integer;}; }, { Collection; in: type; in_repository: Repository; }, { Repository; in: type; name: string; collections: {set; type_of_element: Collection;}; };	class_specification: { creator; in: class; instance_section: Creator; }, { heritage_entity; in: class; instance_section: Heritage_Entity; }, { painting; in:class; superclass: heritage_entity; instance_section: Painting; }, { museum; in: class; instance_section: Repository; };

Consider an OWL information resource – *Hermitage* digital collection – relevant to the cultural heritage mediator. A subset of the Hermitage schema expressed in OWL is shown in the left column of the table 2. A respective representation of the schema in the SYNTHESIS language required for uniformity of the resource and the mediator specifications is shown in the right column of this table. The *Hermitage* OWL schema is mapped into the *Hermitage* SYNTHESIS module in accordance with the mapping σ described in section 3. To save space declarations of classes *entry, drawing, artist, place* and properties *name, author, style, place_of_origin, date, height, width, artworks* like *Declaration(Class(entry))* and *Declaration(DataProperty(name))* are omitted.

During the registration of the *Hermitage* resource in the subject mediator classes of the resource are described as LAV views over virtual classes of the mediator. A view has a form of a conjunctive query. A pair of views aimed to illustrate the result of the registration is shown below.

drawing(d/Drawing[name, author, style, place_of_origin, date, height, width]) ⊆
painting(d/Painting[name: title, author: created_by, place_of_origin,
date: date_of_origin, r_name: in_collection.in_repository.name,
height: dimensions.elem(1), width: dimensions.elem(2)]),
creator(c/Creator[author: name, style: culture]),
r_name = 'Hermitage', date.year >= 1100, date.year < 1990
artist(a/Artist[name, artworks]) ⊆ creator(a/Creator[name, works])

Expression *T[a, b]* here denotes a reduct of the type *T* to attributes *a, b*. Expression *T[c: a]* denotes renaming of the attribute *a* to *c*.

Table 2. Resource schema example

OWL	SYNTHESIS
Ontology(Hermitage SubClassOf(drawing entry) DataPropertyDomain(name entry) DataPropertyRange(name xs:string) ObjectPropertyDomain(author entry) ObjectPropertyRange(author artist) DataPropertyDomain(style entry) DataPropertyRange(style xs:string) ObjectPropertyDomain(place_of_origin entry) ObjectPropertyRange(place_of_origin place) DataPropertyDomain(date entry) DataPropertyRange(date xs:date) DataPropertyDomain(height drawing) DataPropertyRange(height xs:integer) DataPropertyDomain(width drawing) DataPropertyRange(width xs:integer) DataPropertyDomain(name artist) ObjectPropertyDomain(artworks artist) ObjectPropertyRange(artworks entry))	{ Hermitage; in: module; type: { Entry; in: type; name: string; style: string; date: time; author: {set; type_of_element: Artist;}; place_of_origin: Place; }, { Drawing; in: type; supertype: Entry; width: integer; height: integer; }, { Artist; in: type; name: string; artworks: {set; type_of_element: Entry;}; } { Place; in: type; }; class_specification: { entry; in: class; instance_type: Entry; }, { drawing; in: class; superclass: entry; instance_type: Drawing; }, { artist; in: class; instance_type: Artist; }, { place; in: class; instance_type: Place; } }

Views used for integration of other resources (*Uffizi* and *Louvre* digital collections) in the *CulturalHeritage* mediator are provided in [12]. The resources to be integrated together with *Hermitage* collection may be represented in various data models such as relational one, XML, ODMG ODL and so on. Views are crucial for rewriting queries over mediator into queries over resources. In this paper we show rather simple views and avoid complicated issues of view construction taking into account resource and mediator invariants as it is a future work. One more thing required for rewriting is a mapping between a mediator query language and a resource (OWL) query language. We neglect the mapping of query languages mostly due to the fact that there exist a lot of different ontology oriented query languages but no standard of OWL query language.

6 Conclusion

The paper enters into controversy in the context of expansion of ontologies into the area of databases and information systems and cultivation of the opinion that ontologies are at the "semantic" level, whereas database schema are models of data at the "logical" or "physical" level. We show that in the area of databases and information systems OWL would rather be considered as yet another data model to be integrated with other heterogeneous information models in frame of more expressive languages. In the paper we demonstrate that “ontology based” conceptual models can be mapped with preserving of their semantics into the existing data models, specifically those that are used for mediation of heterogeneous databases. This makes possible to integrate in such mediators existing databases implemented in OBDA

under OWL schemas with another, conventional databases. Applying the SYNTHESIS – an extensible language for heterogeneous information resource integration and mediator definition – we show how a sound mapping of a conceptual schemas expressed in OWL 2 QL into SYNTHESIS schemas can be defined. The soundness of the mapping is justified. The paper shows also how the integration of the OWL-defined databases into a SYNTHESIS-based mediator can be provided.

References

1. Acciarri, A., Calvanese, D. et al.: QUONTO: QUerying ONTOlogies. In: AAAI 2005, pp. 1670--1671. (2005)
2. Briukhov D., Kalinichenko L., Martynov D., Skvortsov N., Stupnikov S., Vovchenko A., Zakharov V., Zhelenkova O. Application driven mediation middleware of the Russian virtual observatory for scientific problem solving over multiple heterogeneous distributed information resources. In: Scientific Information for Society – from Today to the Future: 21st CODATA Conference, pp. 80--85. (2009)
3. Cali, A., Gottlob, G., Lukasiewiczz, T. A General Datalog-Based Framework for Tractable Query Answering over Ontologies. In: PODS 2009, pp. 77--86. ACM (2009)
4. Calvanese, D. et al.: Ontology-based database access. In: 15th Italian Conf. on Database Systems, pp. 324--331. (2007)
5. Calvanese, D. et al.: Reasoning Ontologies and Databases: The DL-Lite Approach. In: Web 2009. LNCS, vol. 5689, pp. 255--356. Springer, Berlin Heidelberg (2009)
6. Calvanese, D. et al.: OWL 2 Web Ontology Language: Profiles. W3C, http://www.w3.org/TR/owl2-profiles/ (2009)
7. Gruber, T. R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. J. Human-Computer Studies 43, 907--928. (1995)
8. Gruber, T. R.: "Ontology". In: Liu, L., Özsu M. T. (eds.) Encyclopedia of Database Systems. Springer (2008)
9. Horrocks, I., Parsia, B., Sattler, U.: OWL 2 Web Ontology Language Direct Semantics. W3C, http://www.w3.org/TR/owl2-direct-semantics/ (2009)
10. Kalinichenko, L.A.: Canonical model development techniques aimed at semantic interoperability in the heterogeneous world of information modeling. In: CAiSE INTEROP Workshop, pp. 101--116. Riga Technical University, Riga (2004)
11. Kalinichenko, L.A., Briukhov, D.O., Martynov, D.O., Skvortsov N.A., Stupnikov, S.A.: Mediation Framework for Enterprise Information System Infrastructures. In: The 9th International Conference on Enterprise Information Systems, vol. Databases and Information Systems Integration, pp. 246--251. Funchal (2007)
12. Kalinichenko, L.A., Martynov, D.O., Stupnikov, S.A. Query rewriting using views in a typed mediator environment. In: ADBIS 2004. LNCS, vol. 3255, pp. 37--53. Springer, Berlin-Heidelberg (2004)
13. Kalinichenko L.A., Stupnikov S.A.: Heterogeneous information model unification as a pre-requisite to resource schema mapping. In: V Conference of the Italian Chapter of Association for Information Systems, pp. 373--380. Springer Physica Verlag (2009)
14. Kalinichenko, L.A., Stupnikov, S.A., Martynov, D.O.: SYNTHESIS: a Language for Canonical Information Modeling and Mediator Definition for Problem Solving in Heterogeneous Information Resource Environments. IPI RAS, Moscow (2007)
15. Patel-Schneider, P. F.: OWL 2 Web Ontology Language New Features and Rationale. W3C, http://www.w3.org/TR/owl2-new-features/ (2009)
16. QuOnto Querying ONTOlogies, http://www.dis.uniroma1.it/quonto/?q=node/30