Source Registration and Query Rewriting Applying LAV/GLAV Techniques in a Typed Subject Mediator

Dmitry Briukhov, Leonid Kalinichenko, Dmitry Martynov

Institute of informatics problems RAS
{brd, leonidk}@ipi.ac.ru, domartynov@gmail.ru

Abstract

New methods and tools for application development in collaborative scientific enterprises (like Virtual Observatories (VO)) over multiple distributed sources of data and programs are required. In this paper we focus on results of research and experimental work oriented on problem-driven subject mediation emphasizing aspects of LAV/GLAV information sources integration in the mediator. The approach considered has the following distinguishing features: typed, object canonical model is used instead of usually applied relational one; a technique of refining mapping of source information models into extensible canonical one is provided; registration in a mediator of a relevant source is done so that a mediator type should be provably refined by a relevant source type or by a composition of such types (the conflict resolving functions are to be specified, if required); rewriting of non-recursive logical programs containing strongly typed rules is applied. These features provide methodological context for the current paper that is focused on description of the role the LAV/GLAV approach plays in the mediator. Using astronomical example taken from the Russian VO context, we show the technique of information source registration at the mediator and query rewriting technique in a typed specification environment applying LAV/GLAV approach.

Keywords: subject mediator, LAV, GLAV, canonical typed model, refinement, query rewriting, source registration.

1 Introduction

In this paper[i] subject mediation as an instrument for problem solving over multiple heterogeneous information sources (data and services) will be emphasized. To be specific, we consider subject mediation in e-science that will increasingly be carried out through distributed global collaborations enabled by the Internet. Typically, a feature of such collaborative scientific enterprises is that they will require access to very large sets of data sources, large scale computing resources and high performance visualization providing a feedback to the scientists. For instance, in astronomy such enterprises take a form of Virtual Observatories (VO). To get a consensus on the standards of the enterprise information infrastructures they create alliances (e.g., IVOA in astronomy [11]). Further we shall refer to VO as an example of a scientific enterprise having in mind that the approaches discussed are applicable to various branches of science.

Various scientific problems are to be supported by VO, each of them considering in its own, specific context (probably, overlapping with contexts of other problems). Heterogeneous information sources of various kinds relevant to VO include data sources, service sources, process sources, ontological sources. Many of such sources are autonomous and evolve with time. A set of sources relevant to a specific problem is quite rapidly changing. The technologies applied for creating relevant sources while observing the Universe and to support them are also rapidly evolving. Justifiable identification of sources relevant to a problem, reaching semantic integration of various kinds of them in context of the problem, making problem solving information system stable in the rapidly evolving world of relevant sources constitute serious challenges. New methods and tools for VO application development over multiple distributed collections of data and programs are required.

We emphasize two principally different approaches to the integrated representation of multiple information sources for a scientific problem: 1) moving from sources to problems (an integrated schema of multiple sources is created independently of a definition of specific application) and 2) moving from a problem to the sources (a description of an application subject domain (in terms of concepts, data structures, functions, processes) is created, into which sources relevant to the application are mapped). The first approach *driven by information sources* is not scalable with respect to the number of sources, does not make semantic integration of sources in a context of specific application possible, does not lead to justifiable identification of sources

Proceedings of the 9th Russian Conference on Digital Libraries RCDL'2007, Pereslavl, Russia, 2007

relevant to specific problem, does not provide the required information system stability w.r.t. evolution of the observation sources (e.g., appearance of a new information source relevant to the problem lead to reconsideration of the integrated schema).

The second approach (*problem-* or *application-driven*) assumes creation of subject mediator that supports an interaction between an application and sources on the basis of the application subject domain definition (description of the mediator). For problem solving the subject mediator approach removes the disadvantages mentioned for the approach driven by information sources.

Two approaches for the heterogeneous data sources integration are known as Global-as-View (GAV) and Local-as-View (LAV). According to GAV [10,21] a global schema is defined in terms of the pre-selected sources. LAV [10,21] is opposite: sources are defined as views over the mediator schema. It is easy to see that technically GAV corresponds to the information sources driven approach, and LAV – to the problem driven approach. Later a variation of LAV appeared (called GLAV) [7] allowing the head of the LAV view definition rules to contain any source schemas query and hence is able to express the case where a source schemas are used to define the global schema constructs (GAV).

In this paper we focus on some results of research and experimental work oriented on *problem-driven subject mediation* emphasizing aspects of LAV/GLAV information sources integration in the mediator. The problem-driven approach we are focused on has the following distinguishing features comparing to the usually considered in the information integration context, including:

- the problem solving over multiple heterogeneous distributed sources (data and services) is emphasized (not just integration of sources) [4,5,19,20];
- this feature imposes specific requirements on the canonical mediator model exploited (typed, object data model is used [13] instead of usually considered relational one);
- the notion of extensible canonical information model is introduced [16,15] as a unifying model into which various data and service models (further source models) are mapped;
- a technique of refining mapping of source models into extensible canonical one is provided (kernel of the canonical model is fixed as a hybrid object-oriented/frame-based model; for each source model an extension of canonical model is provided so that the kernel together with extension should be refined by a source data model, the refining condition is formally justified [18,25,26]);
- a mediator is defined in a canonical model, independently of the information sources. Mediator defines a domain of the problems to be solved. The definition contains non-functional requirements (expressed through the mediator metadata – e.g., data quality requirement), ontological specifications, specification of types, classes, functions, constraints, processes for the entities involved;
- for registration in a mediator the relevant source identification is provided through the selection from various registries of the candidates satisfying non-functional requirements (applying metadata), as well as ontologically, structurally and behaviorally matching to respective specifications of a mediator [6];
- a type calculus is applied [14] to provide source schema mapping (expressed in canonical model) into the mediator schema during the registration of relevant to the mediator sources in the mediator.

The features listed above have been described in the cited publications elsewhere. These features provide methodological context for the current paper that is self-contained and focused on description of the role the LAV/GLAV approach plays in the mediator. Using astronomical example (distant galaxies identification problem) taken from the Russian VO context [4], we focus on the techniques of information source registration at the mediator and query rewriting. Both techniques work in a typed mediation environment applying LAV/GLAV.

2 Mediator example definition

In the rest of the paper we will use the example of a subject mediator for distant galaxies discovery problem [4] to demonstrate our techniques. Study of properties of distant galaxies (being on the early stage of evolution) allows to justify cosmological theories. An approach used here is based on distant galaxies detection from radio source samples by matching with optical objects. The candidate object selection procedure may include a few steps to select objects: by radio properties, - angular size (for considerable redshifts usual object angular sizes are in range 1 arcsec to 1 arcmin), morphology (powerful radio galaxies are considered), flux limit (objects considered have fluxes about 100 mJy (on average)), spectral index (steep spectrum sources are preferred), and optical properties - color dropouts and magnitude limits, galaxy position on K-z Hubble diagram and selection by X-ray emission [4]. Definitions of the subject mediator types and classes containing respective information are given in 2.2.

2.1 Canonical information model

For the canonical model of a mediator a hybrid frame-based, object-oriented model (Synthesis language [13]) is used. The canonical model provides support of wide range of data - from untyped data on one end of the range to strictly typed data on another. Typed data should conform to the abstract data type (ADT) specifications prescribing behavior of their instances by means of the type operations. ADT describes interface of a type whose signature defines names and types of its operations. Classes are collections of instances of respective types. Subtyping relationship is supported by the model.

Formulae in the language are used to specify queries, rules and constraints. To specify formulae a variant of a typed (multisorted) first order predicate logic language is used. Predicates in formulae correspond to collections (such as sets and bags of non-object instances), classes treated as set subtypes with object-valued instances, functions. ADTs of instance types of collections and classes should be defined. Rules look as:

$$Q(v/T_v) :- C_1(v_1/T_{v1}), \ldots, C_n(v_n/T_{vn}), F_1(t_1,y_1), \ldots, F_m(t_m,y_m), B$$

where in the body (having SPJ semantics) C_i is a class predicate, F_i is a functional predicate and B is a constraint, that is a conjunction of predicates over the variables $v, v_1, \ldots, v_n$, typed by $T_v, T_{v1}, \ldots, T_{vn}$, or output variables $y_1 \cup y_2 \cup \ldots \cup y_m$ of functional atoms. Each atom $C_i(v_i/T_{vi})$ or $F_j(t_j,Y_j)$ ($i = 1, \ldots, n; j = 1, \ldots, m$) is called a subgoal. t_i is a sequence of terms (might be empty) corresponding to input parameters of a function. Rules can be combined in program blocks.

2.2 Mediator schema specification

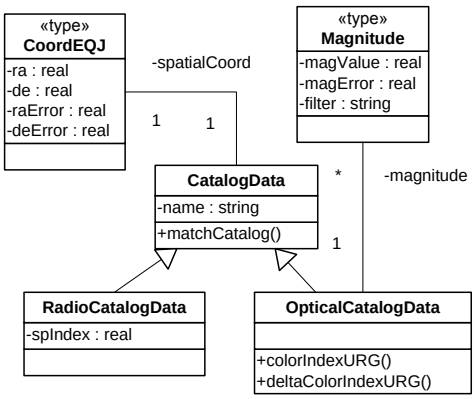

Fig.1 Specification of the mediator schema types

Fig.1 shows a fragment of the mediator schema module for distant galaxy discovery. It includes observed scientific data holdings in radio or optical catalogs (classes *RadioCatalogData* and OpticalCatalogData). The superclass *CatalogData* contains objects from both catalogs. It has the attribute *spatialCoord* (which defines coordinates of an astronomical object), the attribute name (which defines the name of the object in the catalog) and the method *matchCatalog()*, which returns true if two points in space are close to each other.

The functions have the following signatures:

```
matchCatalog: {in: function; params: {
+ra1/real, +de1/real, ra2/real, +de2/real,
+rad1/real, +rad2/real, -returns/boolean};
};
colorIndexURG: {in: function;
  params: {-returns/real};
};
deltaColorIndexURG: {in: function;
  params: {-returns/real};
};
```

Autonomous mediator functions are defined in the module *I_Mediator* with the following functions:

```
{xmatch; in: function; params:{ +ra1/real,
+de1/real, ra2/real, de2/real, +rad1/real,
+rad2/real, -returns/boolean};},

{colorIndexURG; in: function; params: {
+magnitude/{set; type_of_element:
Magnitude;}, -returns/real};},

{deltaColorIndexURG; in: function;
params: { +magnitude/{set; type_of_element:
Magnitude;}, -returns/real};}
}
```

3 Information source registration approach

Each mediator supports the process of systematic discovery and registration of sources uniformly expressing their definitions in terms of the mediator. Such registrations can be done concurrently, at any time and are semi-automatic.

Discovery of relevant data types (classes) in sources is based on three models: metadata model describing the information sources (according to the IVOA standard [4,11]), ontological model defining concepts of a subject domain [15] and canonical model providing for definition of structure and behavior of objects of the subject domain and of the sources.

Source registration in a mediator applying canonical model is a process of specification transformation that includes decomposition of the mediator specifications into consistent fragments, search among relevant sources of the appropriate type fragments considered as candidates for refinement by them of the mediator types, constructing expressions defining source classes as compositions of the mediator classes. It is said that specification *A refines* specification *D*, if it is possible to use *A* instead of *D* so that the user of *D* does not notice this substitution [14]. The fact of specification refinement can be proved in the first order logic due to defining a mapping of the mediator canonical model into formal specification [25]. To manipulate specifications during source registration process a specification calculus has been defined [14]. In the calculus a type specification can be decomposed into set of type *reducts* that can be used as fragments for refinement construction and composition. An operation of constructing *common* (and most common) *reduct* for a pair of type specifications T_1 and T_2 has been defined (a common reduct for types T_1, T_2 is such reduct $R(T_1)$ that there exists a reduct $R(T_2)$ such that $R(T_2)$ is a refinement of $R(T_1)$). In the type algebra the operations for type specification compositions (*meet* and *join*) have also been defined [14]. During registration, concretizing types reconciling the conflicts (of values, structures, behaviors) are defined so that an instance type of the mediator class (or its reduct) would be refined by an instance type of the relevant source class (or its reduct). The main registration result is a *GLAV expression* defining how a source class is determined as a composition of the mediator classes. The registration approach is intended to cope with a dynamic, possibly incomplete set of sources. Sources may change their exported schemas, become unavailable from time to

time. In process of sources evolution a specification of mediator remains stable, only such GLAV expressions need to be modified.

Process of information source registration contains the following activities:

1. select the source candidates satisfying the non-functional mediator requirements (applying metadata and conventional metadata registries);
2. from the candidates obtained at the first step select the sources ontologically relevant to respective specifications of a mediator;
3. for a pair of ontologically relevant types in source and in mediator construct their most common reduct. In case of a conflict between type specifications the conflict resolving functions are to be specified;
4. for each source class construct a GLAV view for this source class and all its ontologically relevant mediator classes.

These activities are semi-automated and supported by the registration tool.

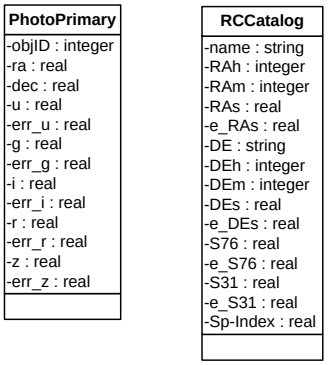

Fig.2 Specification of SDSS (a) and RC Catalog (b) schemas fragments

3.1 Searching for the candidate sources

The process of searching for the candidate sources to be registered at the mediator starts with the metadata search. In this paper we omit the details of this process and assume that two relevant sources have been identified: RC catalog – catalog of radio objects [4] and SDSS (www.sdss.org) – survey of one-quarter of the entire sky, in 5 spectral ranges from ultraviolet to infrared.

Fig.2a shows the fragment of the specification of SDSS optical catalog, the table *PhotoPrimary* after mapping it into the canonical model as a class definition. Fig.2b shows the fragment of the specification of RC radio catalog. Basic attributes for these catalogs are the following: *objID* and *name* specify the name (identifier) of an object in the catalog, *ra*, *de*, *RAh*, *RAm*, *RAs*, *DE*, *DEh*, *DEm*, *DEs* specify the coordinates of an object in space.

3.2 Identification of relevant schema elements

The process of identifying of relevant elements (classes, types, attributes, functions) in mediator and candidate source schemas is based on an ontology integration process [6]. Here we omit the details of this process.

The result of this phase is a set of ontologically relevant elements. A sample of relevant elements in the mediator and SDSS catalog specifications follows:

```
opticalCatalogData ~ photoPrimary
OpticalCatalogData ~ PhotoPrimary
CatalogData.name ~ PhotoPrimary.objID
CoordEQJ.ra ~ ra
CoordEQJ.de ~ dec
Magnitude.magValue ~ u
Magnitude.magError ~ err_u
```

3.3 Most common reduct construction

In our example the source type *PhotoPrimary* is relevant to 3 mediator types - *OpticalCatalogData*, *CoordEQJ* and *Magnitude*. For each of these pairs we construct a most common reduct and respective concretizing reduct. The specification of such reducts for the pair *PhotoPrimary* and *OpticalCatalogData* types follows:

```
{R_OData_SDSS; in: reduct;
    metaslot of: OpticalCatalogData;
      taking: {name, coord, magnitude};
      c_reduct: CR_OData_SDSS;
    end;
};
{CR_OData_SDSS; in: c_reduct;
    metaslot of: PhotoPrimary;
      taking: {objID};
      reduct: R_OData_SDSS;
    end;
simulating: {
R_OData_SDSS.name ~ CR_OData_SDSS.objID,
R_OData_SDSS.coord ~ R_OData_SDSS.get_coord,
R_OData_SDSS.magnitude ~
R_OData_SDSS.get_magnitude
;
get_coord: {in: function; params:{ -returns/
{set; type_of_elements: CoordEQJ;}};
    {{ returns' = this/CR_Coord_SDSS }}
};
get_magnitude: {in: function; params:{ -
returns/ {set; type_of_elements:
Magnitude;}};
    {{ returns' = this/CR_Magnitude_SDSS }}
  };
};
```

The most common reduct *R_OData_SDSS* defines the fragment of mediator type *OpticalCatalogData* which can be refined by the fragment of source type defined by concretizing reduct *CR_OData_SDSS*. Slot *taking* includes a list of attributes taken from the type being reduced.

The line *R_OData_SDSS.name ~ CR_OData_SDSS.objID* in the slot *simulating* indicates that the attribute *name* of the mediator type *OpticalCatalogData* corresponds to the attribute *objID* of the source type *PhotoPrimary*.

The line *R_OData_SDSS.coord ~ CR_OData_SDSS.get_coord* in the slot *simulating* indicates that the attribute *coord* of the mediator type *OpticalCatalogData* is calculated from attributes of the source type *PhotoPrimary* by the conflict resolving function *get_coord*.

Conflict resolving functions (such as *get_coord* and *get_magnitude*) may resolve various kinds of conflicts between the specifications of mediator and source

types.
Similarly we construct most common reducts *R_Coord_SDSS* and *CR_Coord_SDSS* (for *PhotoPrimary* and *CoordEQJ* types), and *R_Magnitude_SDSS* and *CR_Magnitude_SDSS* (for *PhotoPrimary* and *Magnitude* types).

3.4 GLAV views construction

The process of heterogeneous data sources registration in the subject mediator is based on LAV/GLAV approach. This approach allows the head of the LAV view definition to contain any source schemas query and hence is able to express the case where a source schemas are used to define the global schema constructs (GAV). The GAV approach allows to specify functions resolving various kinds of conflicts between mediator and source specifications. It also allows to define rules for transformation of query results from source to mediator.
Thus the registration process consists of constructing a set of GLAV views, having the following form:

$$V(v/T_v) :- Cm_1(v_1/T_{v1}),\ldots,Cm_n(v_n/T_{vn}),\ Fm_1(X_1,Y_1),\ldots,Fm_m(X_m,Y_m),Bm \quad \text{(LAV rule)}$$

$$V(v/T_v) :- Cs_1(v_1/T_{v1}),\ldots,Cs_n(v_n/T_{vn}),\ Fs_1(X_1,Y_1),\ldots,Fs_m(X_m,Y_m),\ Bs \quad \text{(GAV rule)}$$

where Cm_i is a mediator class predicate, Fm_i is a mediator functional predicate, Cs_i is a source class predicate, Fs_i is a source functional predicate or conflict resolving function, *Bm* and *Bs* are conditions.
We start with constructing partial views for each pair of ontologically relevant mediator and source classes.
Then for each source class we compose all its partial views into complete view.
In our example the source class *photoPrimary* corresponds to the mediator class *opticalCatalogData*. As it was shown in 3.3, its instance type *PhotoPrimary* has 3 relevant mediator types (and 3 common reducts *R_OData_SDSS*, *R_Coord_SDSS* and *R_Magnitude_SDSS* corresponding to these types). Therefore we first construct new composed types *CT_OData* and *CT_SDSS* as a join of the common reducts and their respective concretizing reducts. Operation *join* $T_1 \mid T_2$ of the type specification calculus [14] yields the "join" of specifications of the operand types T_1 and T_2 .

```
CT_OData[name, coord, coord_ra, coord_de,
magnitude, magnitude_magValue,
magnitude_magError, magnitude_filter] =
R_OData_SDSS[name, coord, magnitude] |
R_Magnitude_SDSS[magnitude_magValue:
magValue, magnitude_magError:magError,
magnitude_filter:filter] |
R_Coord_SDSS[coord_ra:ra, coord_de:de]
```

```
CT_SDSS[objID, get_coord, get_magnitude, u,
err_u, g, err_g, i, err_i, r, err_r, z,
err_z, get_magValue, get_magError,
get_filter, ra, dec] = CR_OData_SDSS[objID,
get_coord, get_magnitude] |
CR_Magnitude_SDSS[u, err_u, g, err_g, i,
err_i, r, err_r, z, err_z, get_magValue,
get_magError, get_filter] |
  CR_Coord_SDSS[ra, dec]
```

Then using this definition, we construct a GLAV view *v_SDSS_RData* for source class *photoPrimary* and mediator class *opticalCatalogData*.
First we specify a LAV rule:

```
v_SDSS_OData(x/CT_OData[name, coord,
coord_ra, coord_de, magnitude,
magnitude_magValue, magnitude_magError,
magnitude_filter])
:-
opticalCatalogData(x/OpticalCatalogData
[name, coord, magnitude,
coord_ra:coord.ra, coord_de:coord.de,
magnitude_magValue: magnitude.magValue,
magnitude_magError: magnitude.magError,
magnitude_filter: magnitude.filter])
```
(LAV_1)

Instance type of the class predicate *v_SDSS_OData* is *CT_OData*. Attributes from the *OpticalCatalogData*, *Coord* and *Magnitude* types are inserted into the class predicate.
Then we specify a GAV rule:

```
v_SDSS_OData(x/CT_OData[name, coord,
coord_ra, coord_raError, coord_de,
coord_deError, magnitude,
magnitude_magValue,
magnitude_magError, magnitude_filter])
:- photoPrimary(x/CT_SDSS[name:objID,
coord_ra: ra, coord_de: dec])
& get_coord(x, coord)
& get_magnitude(x, magnitude)
& get_magValue(x, magnitude_magValue)
& get_magError(x, magnitude_magError)
& get_filter (x, magnitude_filter)
```
(GAV_1)

Reduct in the class predicate *v_SDSS_OData* contains all attributes from the composed type *CT_OData*. Reduct in the class predicate *photoPrimary* includes only attributes from the composed type *CT_SDSS*. Also we rename source attributes as view attribute names if needed. Such renaming is taken from slot *simulating* at a concretizing reduct specification. E.g., the reduct element *name: objID* defines that the *objID* attribute is renamed to *name*, according to the correspondence *R_OData_SDSS.name ~ CR_OData_SDSS.objID* obtained at the concretizing reduct *CR_OData_SDSS* specified above. Then for each function from the composed type *CT_SDSS* a respective function predicate is appended to the GAV view body. E.g., the function *get_coord* in the appended predicate is taken from the *CR_OData_SDSS* concretizing reduct and the result attribute name is determined as *coord*, the name of the attribute taken from the correspondence *R_OData_SDSS.coord ~ CR_OData_SDSS.get_coord* obtained from the concretizing reduct *CR_OData_SDSS* specified above.
In our example a source class *photoPrimary* corresponds to only one mediator class *opticalCatalogData*, so we do not need to construct a new view as a composition of partial views.
In the same way we construct GLAV view for RC catalog:

```
CT_RData = R_RData_RC | R_Coord_RC
CT_RC = CR_RData_RC | CR_Coord_RC
```

```
v_RC_RData(x/CT_RData[name, spIndex,
coord, coord_ra, coord_raError,
coord_de,coord_deError])
:- radioCatalogData(
x/RadioCatalogData[name,
spIndex,coord, ra: coord.ra, raError:
coord.raError, de: coord.de, deError:
coord.deError])
```
(LAV$_2$)

```
v_RC_RData(x/CT_RData[name, spIndex,
coord, coord_ra, coord_raError,
coord_de, coord_deError])
:- rcCatalog(x/CT_RC[name, spIndex:
Sp-Index, coord_ra, coord_raError,
coord_de, coord_deError])
& get_coord(x, coord)
```
(GAV$_2$)

4 Query program rewriting

To support GLAV approach we have developed a query rewriting algorithm composed of three phases: formal rewriting and semantic analysis implementing LAV approach followed by the third phase - GAV-views unfolding. For non-recursive query programs consisting of several rules the rewriting is applied to each rule separately. Implementation of the LAV approach is based on a variant of the inverse-rule algorithm [27] extended for typed query language [17]. To the results of the formal rewriting phase, we apply semantic analysis that checks a satisfiability of the rule and eliminates Skolem functions.

4.1 Formal rewriting

The formal query rewriting is based on a notion of the inverse rule (further, IR). We consider two kinds of IRs: class-to-class IRs and function-to-funtion IRs. Class-to-class IRs are obtained from a LAV view definition as pairs of the head of the view and a class predicate from the body of the view. The class-to-class IR has the form $C(x/T) \leftarrow V(\_/R)$ where the predicate $C(x/T)$ is called a head of the IR and $V(\_/R)$ is a body of the IR (underscore denotes anonymous variable). For a view $V(\_/R)$:- $C_1(x_1/T_1)$ & ... & $C_n(x_n/T_n)$ & $F_1(t_1, y_1)$ & ...& $F_m(t_m, y_m)$ & B we construct class-to-class IRs $C_1(x_1/T_1)) \leftarrow V(\_/R)$, … , $C_n(x_n/T_n)) \leftarrow V(\_/R)$. IRs define how a particular body predicate expressed in terms of the mediator schema can be rewritten into a predicate expressed in terms of GAV views and the source schema. E. g., occurrences of $C(x/T)$ in a rule can be replaced with $V(\_/R)$ to obtain a contained rewriting if there is an IR $C(x/T)) \leftarrow V(\_/R)$ and reducts T and R designate ADTs related so that R refines T. Generally, R does not include all attributes defined in the body of the view and it should be extended by the skolemization. Since in our data model attributes are defined as respective get-functions, we consider the get-functions of extended attributes as Skolem functions (and mark them with # symbol). E. g., in the view $v(\_/R[a, b]):- p(x/T_p[a, b])$ & $q(y/T_q[b, c])$ & $f(x, z)$, the head reduct does not include attributes x, y, c and z, hence it is extended to the reduct $R[a, b, x:\#x, y:\#y, c:\#c, z:\#z]$. Then two IRs are constructed: $p(x/T_p[a, b]) \leftarrow v(\_/R[a, b, x:\#x])$ and $q(y/T_q[b, c]) \leftarrow v(\_/R[b, y:\#y, c:\#c])$.

In our example, from the view LAV_1 we construct the following class-to-class IR:

```
radioCatalogData(x/RadioCatalogData[
name, spIndex, coord, ra: coord.ra,
raError: coord.raError, de: coord.de,
deError: coord.deError, flux])<-
v_RC_RData(_/CT_RData[x:#x, name,
spIndex, coord, coord_ra,
coord_raError, coord_de,
coord_deError, flux])
```
(IR$_1$)

and from the view LAV_2 we construct the following class-to-class IR:

```
opticalCatalogData(x/OpticalCatalogData
[name, coord, coord_ra: coord.ra,
coord_de: coord.de, magnitude]) <-
v_SDSS_OData(_/CT_OData[x:#x, name,
coord, coord_ra, coord_de, magnitude])
```
(IR$_2$)

Function-to-function IRs are defined directly during the source registration. The function-to-function IR is a formula, which has the form *all* $x_1/T_1, \dots, x_n/T_n$ $F(t, y) \leftarrow G(s, y)$ where $x_1, \dots, x_n$ are variables typed by types $T_1, \dots, T_n$, t and s are terms over these variables, and y is the resulting attribute. Function-to-function IRs are used to rewrite function predicates similarly to class-to-class IRs used to rewrite class predicates.

In our example, there are three function-to-function IRs defined during the registration process:

```
all x/OpticalCatalogData
colorIndexURG(x, ci)
<-ocdColorIndexURG(x.magnitude,ci)
```
(IR$_3$)

```
all x/OticalCatalogData
deltaColorIndexURG(x, dci)
<-ocdDeltaColorIndexURG(x.magnitude,dci)
```
(IR$_4$)

```
all ra1/real, de1/real, ra2/real,
de2/realb, rad1/real, rad2/real
matchCatalog(ra1,de1,ra2,de2,rad1,rad2,b)
<- xmatch(ra1,de1,ra2,de2,rad1,rad2,b)
```
(IR$_5$)

The formal rewriting is applied to the body of a rule. For each body predicate which is not a built-in predicate we choose an IR such, that its head can be unified with this body predicate. The unification of the head of the IR with the body predicate implies the following transformation of the body of the IR.

A class-to-class IR $C(x/T[x_1/T_1:t_1, \dots, x_n/T_n:t_n]) \leftarrow V(\_/R[x/T:y, x_1/T_1:y_1, \dots, x_n/T_n:y_n])$, where $t_1, \dots, t_n$ are attribute paths, can be unified with a body predicate $C'(z/T'[z_1/S_1:t_1.s_1, \dots, z_k/S_k:t_k.s_k])$, where $k \le n$, C is a subclass of the class C', S_i is a subtype of T_i, and $s_1, \dots, s_2$ are attribute paths. The transformed body of the IR is constructed as the following class predicate: $V(\_/R[z/T:y, z_1/S_1:y_1.s_1, \dots, z_k/S_k:y_k.s_k])$.

A function-to-function IR *all* $x_1/T_1, \dots, x_n/T_n$ $F(t, y) \leftarrow G(s, y)$ can be unified with a body predicate $F(t', z)$ if a unification of terms t and t' can be established as a conjunction of equalities predicates $x_1=t'_1$ & ... & $x_n=t'_n$ & E_1 & ... & E_k, where each variable x_i is unified with a term t'_i and E_j are equalities that equate subterms t_j (subterm of t) and t'_j (subterm of t') providing their unification. The transformed body of the IR is constructed as a conjunction of a function predicate

Body predicate	IR	Transformed body of the IR
radioCatalogData(y/RadioCatalogData[name, ra: coord.ra, de: coord.de])	IR_1	*v_RC_RData(_/CT_RData[y:#x, name, ra: coord_ra, coord_raError, de: coord_de])*
opticalCatalogData(x/ OpticalCatalogData[name1, ra1: coord.ra, de1: coord.de])	IR_2	*v_SDSS_OData(_/CT_OData[x:#x, name1: name, ra1: coord_ra, de1: coord_de])*
colorIndexURG(x, ci)	IR_3	*ocdColorIndexURG(x.magnitude, ci)*
deltaColorIndexURG(x, dci)	IR_4	*ocdDeltaColorIndexURG(x.magnitude, dci)*
matchCatalog(ra, de, ra1, de1, 45, 45)	IR_5	*xmatch(ra, de, ra1, de1, 45, 45)*

Table 1 Rewriting of body predicates

$G(s', z)$ & E_1 & ... & E_k where s' is constructed from s replacing all occurrences of $x_1, \ldots, x_n$ with terms $t_1, \ldots, t_n$. E. g., the function-to-function IR *all x/real f(45, x, y) ← g(x,y)* can be unified with *f(a, b, c)* and the transformed body of the IR is constructed as *g(b, c) & a = 45*.

For a given rule $Q(\_/R) :- C_1(x_1/T_1)$ & ... & $C_n(x_n/T_n)$ & $F_1(t_1, y_1)$ & ...& $F_m(t_m, y_m)$ & B and a collection of IRs, where unification of a class-to-class IR with a body predicate $C_i(x_i/T_i)$ implies transformation of the body of the IR: $V_i(\_/R_i)$, and unification a function-to-function IR with the body predicate $F_j(t_j, y_j)$ implies transformation of the body of the IR: $G_j(s_j, y_j)$ & E_j, the following rewriting is constructed: $Q(\_/R) :- V_1(\_/R_1)$ & ... & $V_n(\_/R_n)$ & $G_1(s_1, y_1)$ & ...& $G_m(s_m, y_m)$ & E_1 & ... & E_m & B. The constructed rewriting is a contained rewriting of the initial rule (the limit of the paper size does not permit to include the proof of this statement).

Consider the following rule for our example:

```
r(_/R[ra, de, name, name1, ra1, de1])
:-radioCatalogData(y/RadioCatalogData[name,
ra: coord.ra, de: coord.de]) &
opticalCatalogData(x/OpticalCatalogData[name
1, ra1: coord.ra, de1: coord.de])
& colorIndexURG(x, ci)
& deltaColorIndexURG(x, dci)
& matchCatalog(ra, de, ra1, de1, 45, 45)
& ra >= 120.0 & ra <= 255.0 & de >= 4.39 &
de <= 5.61 & ci > dci
```

This rule is a part of the problem solving. We get astronomical objects contained both in radio catalog and optical catalog applying cross-match function, having coordinates in a given range, and satisfying the specified condition with the methods *colorIndexURG* and *deltaColorIndexURG*.

Table 1 describes the formal rewriting applied to the body of this rule (body predicates of the rule are listed in the first column, the collection of chosen IRs, which are unified with the body predicates, are listed in the second column and implied transformations of the bodies of the IRs are listed in the third column).

After the unifications and transformations have been performed, the following rewriting of the rule is constructed from the transformed bodies of the chosen IRs:

```
r (_/R[ra, de, name, name1, ra1, de1])
:- v_RC_RData(_/CT_RData[y:#x, name, ra:
coord_ra, de: coord_de])
& v_SDSS_OData(_/CT_OData[x:#x, name1: name,
ra1: coord_ra, de1: coord_de])
& ocdColorIndexURG(x.magnitude, ci)
& ocdDeltaColorIndexURG(x.magnitude, dci)
& xmatch(ra, de, ra1, de1, 45, 45)
& ra >= 120.0 & ra <= 255.0 & de >= 4.39
& de <= 5.61 & ci > dci
```

4.2 Semantic Analysis

Since some obtained rewritings may contain Skolem functions and unsatisfiable constraints we should apply additional analysis to derive rules without Skolem functions and filter out unsatisfiable rules.

During this phase we track atomic values of all terms found in the rule and bind them to variables. For a term t, which has a non-scalar type, an ADT with attributes $a_1, \ldots, a_n$, we bind to variable terms $t.a_1, \ldots, t.a_n$ recursively. Projections and joins are translated to a set of equalities over variables. Function predicates are translated to equalities of the form $f(v_1, \ldots, v_n)=y$. E. g., for the rule *r(_/T[a, b]) :- p(_/T[a, b, c]) & f(c, y)* we bind attributes in the head predicate to variables v_1 and v_2, attributes in the class predicate – to variables v_3, v_4, v_5, the attribute y – to v_6. The rule is translated to the set of equalities: $v_1=v_3$, $v_2=v_4$, $f(v_5)=v_6$. A variant of congruence closure algorithm is applied to the set of equalities to construct congruence classes of terms [22] providing a convenient form to represent the set of equalities. E.g., from $v_1=v_3$, $v_2=v_4$, $f(v_5)=v_6$ we construct two congruence classes $\{v_1, v_3\}$ and $\{v_2, v_4\}$. In term of this representation we perform analysis of the rule and LAV views. Congruence classes collected from LAV views referred in the rule (by class predicates) represent equalities that are implied by views and, therefore, it is safe to remove them from the set of equalities of the rule. Then the rule reconstructed from the set of equalities will not contain Skolem functions. In this way we implement elimination of Skolem functions during analysis.

For the rule obtained in the previous section, during semantic analysis we establish that the extended view *v_SDSS_OData* implies equalities between variables bound to *#x.magnitude* and variable bound to *magnitude*, and, therefore, these equalities can be removed and the attribute *magnitude* is used instead of *#x.magnitude* in the rule:

```
r (_/R[ra, de, name, name1, ra1, de1])
:- v_RC_RData(_/CT_RData[name, ra: coord_ra,
de: coord_de]) & v_SDSS_OData(
_/CT_OData[magnitude, name1: name, ra1:
coord_ra, de1: coord_de])
& ocdColorIndexURG(magnitude, ci)
& ocdDeltaColorIndexURG(magnitude, dci)
& xmatch(ra, de, ra1, de1, 45, 45)
& ra >= 120.0 & ra <= 255.0
& de >= 4.39 & de <= 5.61 & ci > dci
```

Along with Skolem functions elimination, we test

satisfiability of the rule. Additionally, we collect all arithmetic predicates found in the constraints of a rule and constraints of a view used in the rule. We operate in terms of the obtained bindings, and, therefore, there are only variables and constants in arithmetic predicates. We consider arithmetic predicates $<$, $\leq$, $\geq$, $>$ and construct an inequality graph to test satisifiability [1]. Also we distinguish inequality graph constructed from arithmetic predicates implied by views and use it to remove arithmetic predicates with Skolem functions.

4.3 GAV unfolding

The first two phases produce a rewriting which can refer to GAV views. To obtain this rewriting expressed in terms of the source schema we perform GAV unfolding. In our approach, it is safe to consider that constructed (during the registration process) GAV views do not share common functions and local classes. In this case, we can perform GAV unfolding of every GAV view found in the rule independently. To unfold a GAV view for a body predicate we unify the head of the GAV view with the body predicate. This unification implies a respective transformation of the body of the GAV view. Then we substitute the body predicate with the transformed body of the GAV view. Function predicates with resulting attributes, which are not referred in the unified head of the GAV view, are removed from the transformed body of the GAV view. To distinguish attributes not included in the head of GAV view and variables, which occur with the same names in few GAV views, we add a unique suffix to all variables and such attributes (in our example, we will use the suffix "1" for *v_RC_RData* and the suffix "2" for *v_SDSS_OData*).
To perform GAV unfolding in the rule obtained in the previous section, we unify the head of GAV_1

```
v_RC_RData(x/CT_RData[name, spIndex, coord,
coord_ra, coord_raError, coord_de,
coord_deError])
:- rcCatalog(x/RCCatalog[name, spIndex: Sp-
Index])
& calcRA2Deg(x, coord_ra)
& calcDE2Deg(x, coord_de)
& calcArcsec2Deg(x, coord_raError)
& calcArcsec2Deg(x, coord_deError)
& get_coord(x, coord)
```

with the body predicate of the rule

```
v_RC_RData(_/CT_RData[name, ra: coord_ra,
de: coord_de])
```

which is replaced with the following transformed body of GAV_1

```
rcCatalog(x1/RCCatalog[name])
& calcRA2Deg(x1, ra) & calcDE2Deg(x1, de)
```

In the same way we unfold *v_SDSS_OData*, replacing it with the following transformed body of GAV_2:

```
photoPrimary(x2/PhotoPrimary[name1: objID,
ra1: ra, de1: dec]) & get_magnitude(x2,
magnitude)
```

The result of GAV unfolding gives the following final rewriting:

```
r (x/R[ra, de, name, name1, ra1, de1])
:-rcCatalog(x1/RCCatalog[name])
& calcRA2Deg(x1, ra) & calcDE2Deg(x1, de)
& photoPrimary(x2/PhotoPrimary[name1: objID,
ra1: ra, de1: dec])
& get_magnitude(x2, magnitude)
& ocdColorIndexURG(magnitude, ci)
& ocdDeltaColorIndexURG(magnitude, dci)
& xmatch(ra, de, ra1, de1, 45, 45)
& ra >= 120.0 & ra <= 255.0
& de >= 4.39 & de <= 5.61 & ci > dci
```

5 Related work

Mediation objectives. Distinguishing feature of the mediation intention in our work comparing to usually considered in the literature [10] is its usage for problem solving over heterogeneous distributed information sources [4,5,19,20] (integration of data sources is only part of the issue). One of the consequencies of such objective is that the mediator schema alongside with data structure specifications includes specification of functions (methods). Refining implementation of the mediator functions are identified during registrtaion among pre-existing sources (services).
Schema mapping and view definitions. Schema mapping is one of the crucial issues for data integration. Clio project [9] for data integration and data exchange supports generation of correspondences between schemas and mappings (queries) between schemas. Given a pair of schemas, the correspondence engine generates and manages a set of candidate correspondences between the two schemas (purely syntactic, based on matching of terms, field names, etc.). The mapping engine supports creation, evolution and maintenance of mappings between pairs of schemas. A mapping is a set of queries from a source schema to a target schema that will translate source data into the form of the target schema. These tasks are interactive, semi-automatic. Currently [9] relational and hierarchical XML schemas are considered in Clio.
Another method, AutoMed [12], is a framework supporting schema transformation for data integration applying LAV, GAV, GLAV approaches. The framework consists of a low-level hypergraph-based data model and a set of primitive schema transformations defined for this model. Higher level data models and primitive schema transformations for them are defined in terms of this lower-level common data model. Pathway is a transformation composed of primitive transformations. This syntactic technique is semi-automatic, it can be specialized for various structured data models.
The work presented in our paper takes completely different, *semantic* information model transformation approach. First, for information integration an extensible canonical information model is introduced (in particular, its kernel is a hybrid object-oriented/frame-based information model [13]). For any source model (defined by syntax and semantics of two languages – information description language and information manipulation language) a mapping into an extension of the canonical model is defined. This mapping is reversible [16,15] and is supported by the

respective compiler (that is used also in process of wrapper generation). Term rewriting approach provided by Meta Environment [3] is applied for compilers generation. A relevance of any source schema to the mediator is established ontologically [15,16]. A relevant source schema expressed in some source model is compiled into the canonical schema. The source schema refines [16] the canonical model schema. The refinement of the schema mapping is formally checked [25]. Now, having schemas expressed in the same notation (canonical model) a LAV/GLAV registration of the source schema in the mediator schema is done relying on the ontological relationships between types, attributes, functions, classes and applying type calculus [14] to form the LAV/GLAV mapping rules as defined in section 3.4.

Query rewriting. The state of the art in the area of answering queries using views ranging from theoretical foundations to algorithm design and implementation has been surveyed in [10]. Additional evaluations of the query rewriting algorithms can be found in other papers [8,27]. Inverse rules algorithms are recognized due to their conceptual simplicity, modularity and ability to produce the maximally-contained rewriting in time that is polynomial in the size of the query and the views. Rewriting unions of general conjunctive queries using views [27] compares favorably with existing algorithms, it generalizes the MiniCon [23] and Ujoin [24] algorithms and is more efficient than the Bucket algorithm. Finding contained rewritings of union queries using general conjunctive queries (when the query and the view constraints both may have built-in predicates) are important properties of the algorithm [27]. No concern of object query semantics in typed environment has been reported. Main contribution of our work is providing an extension of the query rewriting approach using views for the typed subject mediation environment. In contrast with [27], in [17] we showed how to extend conjunctive queries with object SPJ semantics based on type refinement relationship and type calculus. Our LAV-based approach [17] shows that refinement of the mediator class instance types by the source class instance types is the basic relationship required for establishing query containment in the object environment. In current paper, preserving the typed environment, we show specifically how LAV/GLAV technique is used to implement the source/mediator schema conflict resolving functions. The heads of the LAV view definition rules may contain any source schema query resolving conflict through a concretizing view expressed as a GAV rule over respective source. To finalize a rewriting, this source view is to be unfolded according to the GAV approach (section 4).

6 Conclusion

To operate huge, diverse and distributed data volume, to organize efficient problem solving over it, the collaborative scientific enterprises (like VO in astronomy) now are created. New approaches to problem solving organization over multiple distributed collections of data sources and services are required. We distinguish between two principally different such approaches: 1) moving from sources to problems and 2) moving from a problem to the sources. We emphasize here the second approach (*problem-driven*) that supports an interaction between an application and sources on the basis of the application domain definition (description of the mediator). Description of the mediator does not depend on the relevant sources the set of which might be changed arbitrarily during life time of the mediator.

In this paper we focus on results of research and experimental work oriented on problem-driven subject mediation emphasizing aspects of LAV/GLAV information sources integration in the mediator. The approach considered has the following distinguishing features: typed, object canonical model kernel is used; a technique of refining mapping of source information models into extensible canonical one is provided; registration in a mediator of the relevant source is done so that an ontologically relevant mediator type should be provably refined by a source type or by a composition of such types (the conflict resolving functions are to be specified, if required); rewriting of non-recursive logical programs containing strongly typed rules is applied. Using astronomical example taken from the Russian VO context, we have shown the techniques of information source registration at the mediator and query rewriting in a typed mediation environment applying LAV/GLAV approach.

We clearly show that usage of mediation technique for problem solving over heterogeneous distributed information sources requires application of typed (object) canonical model and an ability to semantically identify refining implementation of the mediator during registration of relevant sources (services). We have developed an approach for mapping source information models into extensible canonical model based on refinement check of the schema mapping [16]. This approach has been checked over various kinds of information models [16,18,26]. Obtaining schemas expressed in the same notation, we show in this paper how LAV/GLAV registration of the source schema in the mediator schema is done relying on the ontological relationships between types, attributes, functions, classes and applying type calculus [14] to form the LAV/GLAV mapping rules as defined in 3.4.

Regarding query rewriting, main contribution of our work is applying rewriting to the typed subject mediation environment. Our LAV-based approach [17] shows that refinement of the mediator class instance types by the source class instance types is the basic relationship required for establishing query containment in the object environment. In the current paper, preserving the typed environment, we show how LAV/GLAV technique is used to implement the source/mediator schema conflict resolving functions. For practical purposes the implemented mediation approach is planned to be used in a synergy with the AstroGrid VO infrastructure [2].

Acknowledgements. Authors are grateful to Olga Zhelenkova (Special Astrophysical Observatory of RAS) for formulation of the distant galaxies discovery problem and for participation in the mediator schema consolidation.

References

1. Afrati, F., Li, C., and Mitra, P. Rewriting queries using views in the presence of arithmetic comparisons. Theor. Comput. Sci. 368, 1-2 (Dec. 2006), p. 88-123
2. AstroGrid Release 2007.1. http://software.astrogrid.org/
3. M.G. Van Dn Brand, J.Heering, P.Klint, P.A.Oliver. Compiling Language Definitions: the ASF + SDF Compiler. ACM TOPLAS, Vol. 24, N 4, July 2002
4. Briukhov D.O., Kalinichenko L.A., Zakharov V.N., et al Information Infrastructure of the Russian Virtual Observatory (RVO). Second Edition. IPI RAN, 2005, 173 p.
5. D.Briukhov, L.Kalinichenko, V.Zakharov. Diversity of domain descriptions in natural science: virtual observatory as a case study. Proceedings of the 7th Russian Conference on Digital Libraries RCDL2005, Yaroslavl, October 2005, p. 23 – 30
6. D.O. Briukhov, L.A. Kalinichenko, N.A. Skvortsov. Information sources registration at a subject mediator as compositional development. In Proc. of the East European Conference on "Advances in Databases and Information Systems", Lithuania, Vilnius, Springer, LNCS No. 2151, 2001
7. Marc Friedman, Alon Levy, and Todd Millstein. Navigational Plans for Data Integration. In Proceedings of the National Conference on Artificial Intelligence (AAAI), 1999
8. J. Grant, J. Minker. A logic-based approach to data integration. Theory and Practice of Logic Programming, Vol 2(3), May 2002, 293-321
9. Laura M. Haas, et al Clio Grows Up: From Research Prototype to Industrial Tool. Proc. of the ACM SIGMOD Conference, June 14-16, 2005, Baltimore, Maryland, USA.
10. Alon Y. Halevy. Answering Queries Using Views: A Survey. VLDB Journal, 10(4), 2001.
11. Robert Hanisch, Peter Quinn. The International Virtual Observatory. http://www.ivoa.net/pub/info/
12. Edgar Jasper, Nerissa Tong, Peter Mc.Brien, Alexandra Poulovassilis. Generating and Optimising Views from Both as View Data Integration Rules. Proc. 6th Baltic Conference on Database and Information Systems (DBIS'04), Riga, June 2004
13. L. A. Kalinichenko. SYNTHESIS: the language for description, design and programming of the heterogeneous interoperable information resource environment. Institute of Informatics Problems, Russian Academy of Sciences, Moscow, 1995
14. Kalinichenko L.A. Compositional Specification Calculus for Information Systems Development. Proc. of the Conference on Advances in Databases and Information Systems (ADBIS'99), Maribor, Slovenia, September 1999, Springer Verlag, LNCS
15. L.A. Kalinichenko, N.A. Skvortsov. Extensible Ontological Modeling Framework for Subject Mediation . Proceedings of the Fourth All-Russian Conference on Digital Libraries, RCDL'2002, Dubna, October 15 – 17, 2002
16. L.A.Kalinichenko. Canonical model development techniques aimed at semantic interoperability in the heterogeneous world of information modeling. Proceedings of the Open INTEROP Workshop "Enterprise modeling and ontologies for interoperability" at the 16th Conference on Advanced Information Systems Engineering (CAiSE). Riga, Latvia, 7-11 June, 2004, p. 101 – 116
17. Kalinichenko L.A., Martynov D.O., Stupnikov S.A. Query rewriting using views in a typed mediator environment. Proceedings of the East-European Conference on "Advances in Databases and Information Systems" (ADBIS'04), Hungary, Budapest, Springer, Lecture notes in Computer Science, Vol. 3255, September 2004
18. L.Kalinichenko, S.Stupnikov, N.Zemtsov. Extensible Canonical Process Model Synthesis Applying Formal Interpretation. Proceedings of the 9th East European Conference Advances in Databases and Information Systems, ADBIS-2005, Tallinn, September 2005, p. 183 - 198
19. L. Kalinichenko, S. Stupnikov, et al. Russian Virtual Observatory Community Centre for Scientific Problems Solving over Multiple Distributed Information Sources. Proc. of the 8th Russian Conference on Digital Libraries RCDL2006, Suzdal, Russia, 2006, p. 120 – 129
20. Leonid Kalinichenko. Subject mediation approach for scientific problem solving in Virtual Observatories. XXVIth General Assembly of the International Astronomical Union, Prague, August 14 – 25 2006, p. 454-455
21. Maurizio Lenzerini. Data Integration: A Theoretical Perspective. In Proceedings of the ACM Symposium on Principles of Database Systems (PODS), 2002.
22. R. Nieuwenhuis and A. Oliveras. Congruence closure with integer offsets. Proceedings of the 10th Int. Conf. Logic for Programming, Artif. Intell. and Reasoning (LPAR). LNAI, vol. 2850, 2003, p. 78--90
23. R.Pottinger, A.Levy. A scalable algorithm for answering queries using views. In Proc. Of the Int. Conf. on Very Large Data Bases (VLDB), Cairo, Egypt, 2000
24. X. Qian. Query folding. In Proc. of Int. Conf. on Data Engineering (ICDE), p.p. 48 – 55, New Orleans, LA, 1996
25. Stupnikov S.A. Mapping of canonical model core specifications in Abstract Machine Notation. In paper collection "Formal methods and models in compositional infrastructures of distributed information systems", IPI RAS, Moscow, 2005, p. 64 – 88 (in Russain)
26. Stupnikov S.A., Kalinichenko L.A., Bressan S. Interactive discovery and composition of complex Web services. In Proceedings of the East-European Conference on "Advances in Databases and Information Systems" (ADBIS'06), Springer, 2006, p. 216 – 231
27. J. Wang, M.Maher, R. Topor. Rewriting Unions of General Conjunctive Queries Using Views. In Proc. of the 8th International Conference on Extending Database Technology, EDBT'02, Prague, Czech Republic, March 2002

[i] This research has been partially supported by the RFBR grants 06-07-08072 and 06-07-89188 and RAS project 1-10 of the program on "Fundamentals of IT and systems"