Constructing of Mappings of Heterogeneous Information Models into the Canonical Models of Integrated Information Systems

Leonid Kalinichenko and Sergey Stupnikov

Institute of Informatics Problems, Russian Academy of Science
{leonidk, ssa}@synth.ipi.ac.ru

Abstract. The paper proposes an approach for semi-automatic construction of mappings of information models of heterogeneous information resources (such as databases, services, processes, ontologies) into the unifying, canonical models of the integrated interoperable information systems. The approach proposed is based on verifiable methods and tools of information model mapping preserving information and operations and synthesis of extensible canonical information models. An architecture of the Information Model Unifier that has been developed for supporting the methods is briefly described and illustrated by an example of mapping of a specific information model into the canonical one. A short overview of existing database schema mapping approaches and tools as well as their comparison with the approach developed in our project is provided.

1 Introduction

The paper[1] is devoted to analysis, transformation and unification of information models for design and development of integrated and interoperable information systems (*I-systems* for short).

The current period of IT development is characterized by an explosive process of information models creation. This development takes place in frame of specific distributed infrastructures (such as OMG architectures, model driven architectures (MDA), semantic Web architectures, services-oriented architectures, digital library architectures, architectures of the information grid), as well as in the standards of concrete information models — data models (such as ODMG 2000, SQL 2003, UML, XML and RDF stacks), workflow models (e.g. Staffware, COSA, InConcert, Eastman, FLOWer, Domino, Meteor, Mobile, MQSeries, Forte, Verve, Vis. WF, Changeng, IFlow, SAP/R3), process service composition languages (XPDL, BPEL4WS, BPML, XLANG, WSFL, WSCI), semantic models (including ontological models and models of metadata), etc. This process is accompanied by another trend — the accumulation of based on such models information

[1] This research has been done under the support of the RFBR (projects 06-07-08072-ofi-a, 06-07-89188-a, 08-07-00157-a) and the Program of the Department of Nanotechnologies and Information Technologies of RAS entitled as *Basic principles of information technologies and systems* (project 1-10).

resources, the number of which grows exponentially. This growth causes the accelerating need for interoperable use (integration) of components and services represented in heterogeneous models in various applications, as well as their reuse and composition implementing new interoperable information systems [1]. The indicated trends are contradictory: the more variety of used models we meet in various resources, the more complex become problems of their integration and composition. Generally resources are heterogeneous (represented in various models) so the problems of I-systems development require to unify these models in the frame of some unifying information model called *canonical*. This semantic unification of various models specifications in the canonical model demands special methods and tools that do not exist in practice of I-systems development. So the experts have to rely on intuitive techniques of identification of required resources with given properties and transform information models manually neglecting semantics of used models and justification of transformations. Thus satisfactory results are achieved only for the simplest models.

Unification of heterogeneous specifications requires primarily a technique of matching the specifications of various resources. The matching have to answer whether it is possible to use a specification of existing resource instead of an I-system specification fragment while implementing of the I-system. For that it is sufficient to prove that specifications considered are in *refinement* relation. 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. Methods and tools of proving that a specification of some resource is refined by a specification of other resource (based on the model-theoretic notations and supporting tools) constitutes foundations of methods of unifying (canonical) information models development for I-systems. The canonical information model is used as an Esperanto for the uniform representation of semantics of various information models used in resources of I-systems.

The main principle of the canonical model synthesis for an I-system is the *extensibility* of the canonical model kernel in heterogeneous environment, including various models used for the representation of resources of the I-system. A kernel of the canonical model is fixed. For each specific information model M (called *source*) of the environment an extension of the kernel is defined so that this extension together with the kernel *is refined* by M. Such refining transformation of models should be *provably correct*. The canonical model for the environment is synthesized as *the union of extensions*, constructed for all models M of the environment. Also preserving of operations and information of a source model while mapping it into the canonical one is required. This could be reached under the condition that the mappings of the models are commutative [2].

Over a long period of time the authors have been developing the methods of the canonical models synthesis for the wide range of real information models: structured, object, service, process including their arbitrary combinations [3][2][4][5]. During that full specifications of models (languages) including both information structures (data types) and behavior (operations, functions and processes) were considered.

With regard to explosive extension of information model diversity it seems impossible to operate manually with such "Tower of Babel" of information models using created methods of mapping of resource specifications into I-systems specifications and methods of I-systems canonical models synthesis applying the refinement theory. That is why the creation of Unifying Information Models Constructor (*Model Unifier* in short) aimed at partial automation of methods of the canonical models synthesis for the I-systems development is important. The Unifier allows to provably reduce a set of heterogeneous source information models to the canonical unifying representation and implements strategically important stage of I-systems development.

Though the results of the work considered can be applied generally to various I-systems, the specific pragmatic purpose of this particular work is to develop information model mapping approach for the heterogeneous information resource integration in a specific subject mediator middleware [1]. Due to that, the subject mediator specification information model (the SYNTHESIS language [6]) has been chosen as the canonical model kernel. The SYNTHESIS language distinguishing features include: hybrid semistructured and object oriented data model and includes facilities for definitions of frames, abstract data types, classes and metaclasses, facilities for definition of functions, assertions and processes, logical formulae facilities applied for description of constraints, queries, pre- and postconditions of functions, assertions and conditions related to processes. For extension of the canonical model kernel we apply metaclasses, metaframes, parameterized constructions including assertions and generic data types. Comprehensive facilities of the kernel provide for an ability to construct refining mapping of various kinds of information models into the canonical model kernel chosen. The Unifier is considered as a constituent part of the subject mediator middleware [1].

The paper is structured as follows. In section 2 the architecture of tools for unifying information models development is briefly described. In section 3 an example of mapping of OWL (Web Ontology Language) into the canonical model is presented. The main stages of the mapping construction are illustrated: formalization of OWL syntax and semantics, integration of concepts of OWL and the canonical model, creation of the canonical model extension, construction of translator of OWL into extended canonical model, verification of refinement of extended canonical model by OWL. In section 4 the related works are considered and their comparison with the proposed approach is presented. In conclusion the contribution of the work is summarized.

2 Architecture of Tools for Unifying Information Models Development

The aim of Model Unifier is to unify a set of information models (called *source* models) used interoperably in some I-system. A source model R is said to be *unified* if it is mapped into the canonical model C. This means a creation of such extension E of the canonical model kernel (note that such extension can be empty) and such mapping M of a source model into extended canonical one that the source model *refines* the extended canonical one. Model refinement of C by R means that for any admissible specification r represented in R its image $M(r)$ in C under the mapping M is refined by the specification r. Process of model mapping includes a possibility of proving that arbitrary specification r represented in R refines its image $M(r)$. Verification of model refinement is realized over a set of source model specification samples.

Hence the following languages and formal methods are required to support the information model mapping:

- a kernel of the canonical information model;
- formal methods allowing to describe information model syntax as well as semantic mappings (translators) of one model to another;
- formal methods supporting verification of refinement reached by the mapping.

As a kernel of the canonical information model the SYNTHESIS language [6] is considered in this paper. The language is intended for canonical information modeling and mediator definition for problem solving in application-driven distributed heterogeneous information environment.

For the formal description of model syntax and translators the metacompilation languages SDF (Syntax Definition Formalism) and ASF (Algebraic Specification Formalism) are used. For the languages a tool support — Meta-Environment [7] — is provided.

The AMN (Abstract Machine Notation) language [8] based on the first order predicate logic and set theory is used for model's semantics formalization and refinement verification. AMN is supported by technology and tools for proving of refinement (B-technology) [9].

Mapping of the source model R realized by expert with a support of Model Unifier is divided into the following stages (syntax and semantics of the canonical model kernel are supposed to be already defined):

1. formalization of the model R syntax and semantics;
2. definition of *reference schemas* of the model R and the canonical information model(if the latter has not yet been defined);
3. integration of reference schemas of the model R and the canonical information model;
4. creation of required extension E of the canonical model C;
5. construction of translator of the model R into extended canonical model;
6. verification of refinement of extended canonical model by model R.

Reference schema of information model is an abstract description containing concepts related to constructions of the model and significant associations among these concepts. Both concepts and associations may be annotated by verbal definitions (looking like entries in explanatory dictionary).

Note that the stage of model refinement verification is labor-intensive and technically complicated so it may be optionally applied when required.

Unifier consists of the following main components (groups of components):

– Meta-Environment (used for the formal description and correctness checking of model syntax and translators);
– B-Toolkit (supporting AMN and tools for proving of specification refinement) [9];
– metainformation repository;
– model manager.

Meta-Environment and B-Toolkit are independent products. Metainformation repository is an object-relational database and is used for the implementation of *model registry* and as specification storage. Model registry contains *registration cards* of models, canonical model extensions, specification samples. All the information produced during mapping of models (including information produced during interaction of expert with Meta-Environment and B-Toolkit) is stored in the registration cards.

Model manager provides a graphical interface allowing an expert to connect to concrete metainformation repository; to search for, view and register information models and extensions of the canonical model; to call specific components for generating templates, editing, loading into repository and integration of reference schemas, generating templates for translators of source models into the canonical one, translation of source models specifications into AMN or into canonical specifications, translation of canonical specifications into AMN.

3 An Example of Mapping of a Source Information Model into the Canonical One

In this section an example of mapping of OWL (Web Ontology Language, [10]) — a semantic markup language for publishing and sharing ontologies on the World Wide Web — into the canonical model is presented. Only a fragment of the language is considered.

3.1 OWL Syntax Formalization

Syntax and semantics of OWL is described in [11]. Note that the document describes not XML-syntax but *abstract syntax* independent of any XML representation. Syntax is represented in a version of extended Backus–Naur form. The rules look as follows:

```
axiom ::= ... |
  'ObjectProperty(' propertyID
  [ 'Functional' | 'Transitive' ]
  { 'domain(' classID ')' } { 'range(' classID ')' } ')'
```

Here the symbol `::=` separates a head and a body of a rule, vertical line (|) means alternative, optional parts of a rule are enclosed by brackets [ ], repeating parts are enclosed by braces {}.

The first stage of model mapping is a transformation of the source syntax into formal SDF-syntax:

```
module unifier/owl/OWL-Syntax
sorts  Axiom ObjectPropertyAxiom ...
context-free syntax
  ObjectPropertyAxiom -> Axiom
  "ObjectProperty" "(" OwlID  PropertyKind?
  ObjectPropertyDomain* ObjectPropertyRange* ")"
  -> ObjectPropertyAxiom
```

As illustrated by the example the formal syntax of OWL forms the SDF-module *OWL-Syntax* including the section of sorts and the section of context-free syntax (consisting of rules). A rule in SDF is separated by the symbol `->`, a head of the rule is located on the right of the separator, a body is located on the left of the separator. Optional parts of the rule are marked by question mark `?`, repeating parts – by the symbol `*`.

3.2 Formalization of OWL Semantics in AMN

Formalization of semantics means a construction of ASF-translator mapping OWL specifications into AMN-specifications. In this section the main principles of mapping of OWL into AMN are illustrated by a small subset of wine ontology[2]:

```
Class(Wine
  restriction(hasSugar cardinality(1)))
Class(DessertWine  Wine
  restriction(hasSugar allValuesFrom(oneOf(OffDry Sweet))))
ObjectProperty(hasSugar domain(Wine) range(WineSugar))
EnumeratedClass(WineSugar Sweet OffDry Dry)
```

The main class of ontology is *Wine*, class *DessertWine* is a subclass of *Wine*. Wine is characterized by a sugar content property *hasSugar*. Sugar content of dessert wine have to be sweet or off-dry only.

An image under mapping of this ontology into AMN is the specification *Wines*:

REFINEMENT *Wines*
SETS *Ind*; $WineSugar = \{Sweet, OffDry, Dry\}$
VARIABLES $Wine, DessertWine, hasSugar$
INVARIANT
$Wine \in POW(Ind) DessertWine \in POW(Ind) \wedge DessertWine \subseteq Wine \wedge$
$hasSugar \in Wine \leftrightarrow WineSugar \wedge$
$\forall\, wine.(wine \in Wine \Rightarrow \text{card}\,(hasSugar[\{wine\}]) = 1) \wedge$
$\text{ran}\,(DessertWine \lhd hasSugar) = \{OffDry, Sweet\}$
OPERATIONS
$set\_hasSugar(ind, val) =$

[2] Full example of wine ontology is presented in *OWL Web Ontology Language Guide, W3C Proposed Recommendation*, http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine.rdf.

```
PRE
   $ind \in Wine \wedge val \in WineSugar \wedge$
   $ind \in DessertWine \Rightarrow val \in \{OffDry, Sweet\}$
THEN
   $hasSugar(ind) := val$
END
$include\_Wine(ind) = \dots$
$include\_DessertWine(ind) = \dots$
END
```

A set of individuals of the ontology is represented in AMN by the set *Ind*. Enumerated classes (*WineSugar* for instance) are represented by separate sets. Classes (*Wine* for instance) are represented by variables typed in invariant as subsets of *Ind*. Object properties (*hasSugar* for instance) are represented by variables typed as relations between set representing domains and ranges of properties. Various restrictions over classes and properties (value restrictions, cardinality restrictions, class equivalence axioms, etc.) are represented by the conjunctive parts of the invariant. Every class is represented also by an operation adding a new instance to the class. Every object property is represented also by an operation modifying the property value of some individual. In the example the body of only one operation (related to the property *hasSugar*) is considered. The operation modifies the value of the property *hasSugar* of the individual *ind* to the value *val* if and only if the precondition of the operation holds. The precondition guarantees preservation of the invariant after the execution of the operation.

To save space, the details of ASF-translator of OWL into AMN are omitted here. Some details concerning general structure of ASF-translators are illustrated in subsection 3.5.

3.3 Definition and Integration of the Reference Schemas of OWL and the Canonical model

In this section the integration of reference schemas of OWL and the canonical model is illustrated by a small example (fig. 1).

To save space and make reference schemas (especially associations between the elements of schemas) more readable they are represented here as UML class diagram.

On the top of the figure the UML-diagram of types, attributes and associations of OWL reference schema is shown. Two kinds of associations are presented: generalization – specialization relation (for instance, type *Restriction* is a subtype of type *Description*) and associations of various cardinalities (for instance, class axiom *ClassAxiom* can be associated with several descriptions *Description*).

On the bottom of the figure the types of the canonical model reference schema are shown.

The definition of the reference schemas starts with the automatic generation of reference schema templates on the base of the SDF-syntax of the models

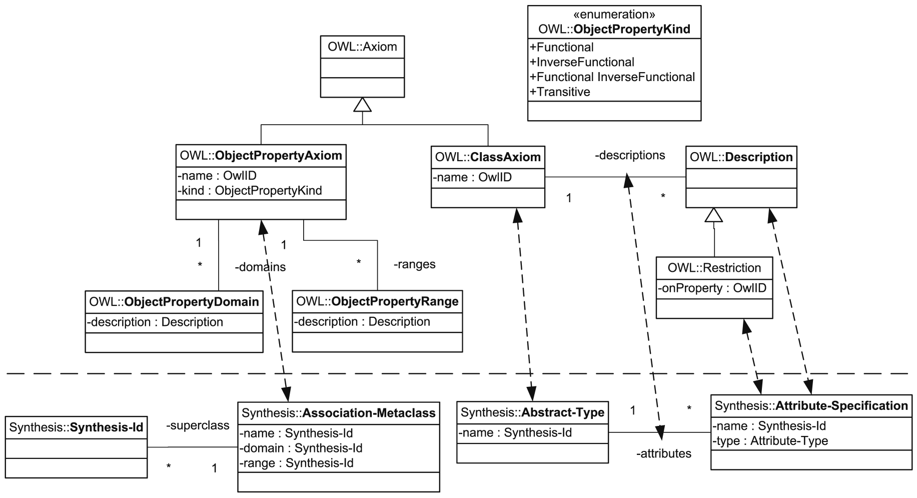

Fig. 1. Integration of the reference schemas of OWL and the canonical model

(for instance, types *Axiom, ObjectPropertyAxiom, ObjectPropertyDomain, ObjectPropertyRange*, their attributes and associations are created in accordance with SDF-syntax rules shown in section 3.1). After that an expert completes the template using reference schema editor (types, attributes and associations may be added, deleted or modified). Types, attributes and associations constituting reference schemas may be also annotated by verbal definitions created by an expert on the base of model descriptions (such as [11] or [6]). To save space, the annotations are omitted in the example.

The aim of the integration of the reference schemas is to identify the relevant constructions of source and canonical models. For this identification verbal annotations are used. The establishing is done automatically with an expert interaction.

The list of correspondences between OWL and the canonical model constructions for the subsets of the models shown on the figure 1 is presented in the table 1. The correspondences are shown as dotted arrows on the figure.

The correspondences presented in the table are one-to-one, but methods of construction of translators illustrated in subsection 3.5 allows them to be many-to-many. This adds not methodological but technical difficulties.

3.4 Creation of the Canonical Model Extension

During the integration of OWL and canonical model reference schemas it appeared that attribute *kind* of type *ObjectPropertyAxiom* (defining a kind of an object property — transitive, functional etc.) and type *ObjectPropertyKind* can not be put into correspondence to any construct of the canonical model kernel.

Table 1. Correspondences between OWL and the canonical model constructions

OWL construction	Canonical model construction
OwlID	Synthesis-Id
ClassAxiom	Abstract-Type
ClassAxiom.name	Abstract-Type.name
ClassAxiom.descriptions	Abstract-Type.attributes
Description	Attribute-Specification
Restriction	Attribute-Specification
Restriction.onProperty	Attribute-Specification.name
ObjectPropertyAxiom	Association-Metaclass
ObjectPropertyAxiom.name	Association-Metaclass.name
ObjectPropertyAxiom.domains	Association-Metaclass.domain
ObjectPropertyAxiom.ranges	Association-Metaclass.range

At the same time the concept of object property (*ObjectPropertyAxiom*) of OWL corresponds to the concept of association metaclass (*Association-Metaclass*) of the canonical model. It was decided to extend the canonical model kernel by new association metaclasses — *Transitive*, *Functional*, etc. Consider the specification of *Transitive* metaclass in SYNTHESIS as an example:

```
{ Transitive;
  in: association, metaclass;
  instance_section: {
    domain: type; range: type;
    transitivity: { in: predicate, invariant;
      {{ all a/this.domain.inst, b/this.domain.inst, c/this.domain.inst(
           b/this.range.inst & in([a,b], this) & in([b,c], this) ->
             in([a,c], this)) }}
} } }
```

Association transitivity is expressed by the invariant *transitivity* stating the following. Let a, b, c be instances of some type T. If a is associated with b according to association *assoc* and b is associated with c according to *assoc* then a is associated with c according to *assoc*.

Mapped into AMN this semantics is expressed by the following invariant:

$$\forall\, a, b, c\ (a \in ext\_T \wedge b \in ext\_T \wedge c \in ext\_T \wedge \\ a \in assoc(b) \wedge b \in assoc(c) \Rightarrow a \in assoc(c))$$

Generally, creation of an extension means definition of its SDF-syntax, AMN-semantics and reference schema. After that the integration of reference schemas of source and extended canonical model is refined: correspondences between constructions of source model and extension are established and fixed by an expert.

In our example the list of correspondences between OWL and canonical model constructions is extended by the correspondence between attribute *kind* of type

ObjectPropertyAxiom of OWL reference schema and association *superclass* of type *Association-Metaclass* of the canonical model reference schema. The correspondence means that the kind of OWL object property is represented in the canonical model by superclass relation between association metaclass representing object property and an association metaclass of the canonical model extension (for instance, *Transitive*).

3.5 Construction of Translator of OWL into Extended Canonical Model

Input data of this stage are a list of correspondences between constructions of source and canonical models, SDF-syntax and verbal semantics of source and canonical models.

The following elements of ASF-translator are generated automatically (actually a template of the translator is generated to be completed by an expert):

– translator name – `unifier/owl2synthesis/owl-translator`;
– list of imported modules:

```
imports unifier/owl/OWL-Syntax
imports unifier/synthesis/Synthesis-Syntax
```

– list of variables, e.g.:

```
"ObjectPropertyAxiom"[0-9\']* -> ObjectPropertyAxiom
"Attribute-Specification*"[0-9\']* -> Attribute-Specification*
```

  The definitions mean that variables of the respective sorts may be used in rules of translation (for instance, variable *ObjectPropertyAxiom2* of sort *ObjectPropertyAxiom*).
– list of translation function signatures, e.g.:

```
t-Module-Def(Ontology) -> Module-Def
```

  The function *t-Module-Def* here transforms an OWL ontology into module of the canonical model. The signature is generated according to correspondence of elements *Ontology* and *Module-Def*.
– list of translation rule templates, e.g.:

```
[Module-Def]
Synthesis-Id := t-Module-Name(OwlID),
Type-Specification* := t-Type-Specification-List(Directive*),
Class-Declarator* := t-Class-Declarator-List( Directive* )
====>
t-Module-Def( Ontology( OwlID Directive*) ) =
{ Synthesis-Id; in: module;
  type: Type-Specification*;
  class_specification: Class-Declarator*; }
```

The rule describes the function *t-Module-Def*. The rule is generated according to the analysis of the element *t-Module-Def*, its attributes and associations (module name corresponds to OWL ontology name, type specifications and class declarators correspond to directives). The rule uses recursive function calls for transformations of directive list into type and class lists.

Lists of imported modules, variables, translation function signatures, translation rules are completed (extended, modified) by an expert.

According to the ASF-definition the translator program code (C language) is generated automatically by means of Meta-Environment tools. The translator obtained is used for mapping of source model specifications into the canonical model specifications. One of the primary aims of the translator creation is the use of the translator for I-system development. Translator is used to map a schema of resource specified in a source model into a schema specified in the canonical model.

3.6 Verification of Refinement of Extended Canonical Model by OWL

In this section a verification of mapping of OWL into the canonical model is illustrated by an example of OWL schema (wine ontology described in section 3.2). OWL specification and its AMN semantics have been already considered. Mapping of the ontology into the canonical model, mapping of the canonical specification into AMN and verification of AMN-specifications refinement are considered further.

Canonical specification generated by the translator of the OWL wine ontology into the canonical model (briefly described in previous section) looks as follows:

```
{ Wines; in: module, ontology;
type:
{ Wine; in: type, owl;
  hasColor: WineColor;
   metaslot in: HasColor; min_card: 1; max_card: 1 end
},
{ DessertWine; in: type, owl;
  supertype: Wine;
  restriction_hasSugar: {
    in: predicate, invariant;
    {{ all w/DessertWine (in(w.hasSugar, {OffDry, Sweet})) }}
  };
},
{ HasSugar; in: association, metaclass, owl;
  instance_section: { domain: Wine; range: WineSugar; };
},
{ WineSugar; { enum; enum_list: {Dry, OffDry, Sweet} } };

class_specification:
{ wine; in: class, owl;
```

```
  instance_section: Wine;
},
{ dessertWine; in: class, owl;
  superclass: wine;
  instance_section: DessertWine;
}; }
```

Enumerated classes of OWL (for instance, *WineSugar*) are represented in the canonical model as enumerated types having the same names. Classes (for instance, *Wine*) are represented by types (*Wine*) and classes (*wine*) of the canonical model. Object properties (for instance, *hasSugar*) are represented by type attributes and association metaclasses (*HasSugar*). Property value restrictions are represented by type invariants (for instance *DessertWine.restriction_hasSugar*).

The image under mapping of this canonical specification into AMN looks as follows:

REFINEMENT *Wines*
SETS $AVAL$, $WineSugar = \{Sweet, OffDry, Dry\}$
CONSTANTS $Obj, ext\_Wine, ext\_DessertWine$
PROPERTIES
$\quad Obj \in POW(AVAL) \wedge ext\_Wine \in POW(Obj) \wedge$
$\quad ext\_DessertWine \in POW(Obj) \wedge ext\_DessertWine \subseteq ext\_Wine$
VARIABLES
$\quad wine, iceWine, dessertWine, hasSugar$
INVARIANT
$\quad wine \in POW(ext\_Wine) \wedge dessertWine \in POW(ext\_DessertWine) \wedge$
$\quad dessertWine \subseteq wine \wedge hasSugar \in ext\_Wine \rightarrow WineSugar \wedge$
$\quad \forall\, w(w \in ext\_DessertWine \Rightarrow hasSugar(w) \in \{OffDry, Sweet\}) \wedge$
OPERATIONS
$include\_Wine(obj) = \ldots$
$include\_DessertWine(obj) = \ldots$
$set\_hasSugar(obj, val) =$
PRE
$\quad obj \in wine \wedge val \in WineFlavor \wedge$
$\quad obj \in dessertWine \Rightarrow val \in \{OffDry, Sweet\}$
THEN
$\quad hasSugar(obj) := val$
END

Set of all abstract values (values of all abstract data types) is represented in AMN by $AVAL$ set. Set of object type values is represented by Obj — subset of $AVAL$. Types are represented by their extents — sets of admissible values (for instance, type *Wine* is represented by extent ext_Wine). Extents of object types are typed in PROPERTIES clause as subsets of Obj. Type — subtype relation is represented by set — subset relation over extents.

Classes (for instance *wine*) are represented by variables typed in INVARIANT clause as subsets of extents of instance types. Class — subclass relation is represented by set — subset relation over respective variables.

Type attributes (for instance *hasSugar*) are represented by variables typed in INVARIANT clause as functions having type extents as domains.

Every class is represented also by an operation adding a new instance to the class. Every attribute is represented also by an operation modifying the attribute value of some object. In the example the body of only one operation (related to the attribute *hasSugar*) is considered. The operation modifies the value of the attribute *hasSugar* of the object *obj* to the value *val* if and only if the precondition of the operation holds. The precondition guarantees preserving of the invariant after the execution of the operation.

AMN-specifications related to OWL wine ontology and its canonical specification were used as input for the tool of refinement proving (B-Toolkit 5.4.1. [9]). It automatically formulated 49 theorems, expressing the fact of specification refinement. Large number of theorems is explained by automatically subdividing complex theorems by the tool into simpler ones to prove them independently. 11 theorems were proved automatically, the rest were proved interactively. In the table 2 total number of theorems formulated and number of theorems automatically proved are shown.

Table 2. The number of theorems

	Number of theorems	Number of automatically proved theorems
Theorem of non-emptiness of the joint state	1	0
Theorems of refinement for operation *include_Wine*	17	5
Theorems of refinement for operation *include_DessertWine*	20	5
Theorems of refinement for operation *set_hasSugar*	11	1
Total number of theorems	49	11

4 Related Work

During last ten years the most close research are carried out by the Microsoft Research and some universities of Europe and USA. The aim of the research consists in the development of unified methods and tools for metadata manipulation in database area primarily. These methods and tools are concentrated on mapping of a database schema expressed in one data model into *respective* database schema expressed in the other model [12].

The main idea of *ModelGen* approach [13] is using a metamodel – a set of constructions applied for definition of data models that are instances of the metamodel. For the definition of this set of constructions the unified, independent of any data model metaconstructions are used. Metaconstructions were classified by Hull and King [14] applying some categories.

Every model is identified by a subset of such constructions and respective metaconstructions. Mapping of a schema expressed in one data model into the other is defined in terms of metaconstruction transformation. *Supermodel* is a data model containing constructions that corresponds to all metaconstructions known to system. Every data model is a *specialization* of the supermodel, so a schema expressed in any data model is a schema in supermodel.

Mapping includes the following steps:

1. Translation of source schema into supermodel.
2. Translation of the result into target schema realized in frame of the supermodel.
3. Translation of the target schema into target model.

First and third steps are said to be labor-intensive and straightforward because every model (source or target) *is subsumed* by supermodel. Transformation coded by the authors relates only to the second step.

Problems of translation of schemas expressed in source data model (e.g. object, relational, XML-oriented) into target one are considered in [15]. A prototype for interactive generation of relational schemas from object-oriented ones embedded into Microsoft Visual Studio 2005 is mentioned. For the schema translation specific rules are used, intended to eliminate those constructions of source schema that are not presented in target data model.

The authors of [16] propose to use extensible model. Specialization and refinement are to be used for extending the model metaconstructions of which are organized as inheritance hierarchy (lattice) of predefined concepts. The approach is considered as an alternative for the supermodel metaconstruction approach.

A result of research of IBM Almaden Center — CLIO [17] system — allows to map source database schemas into target ones expressed in a restricted subset of structured information models.

Briefly a comparison of approach proposed in this paper *(SYNTHESIS)* and *ModelGen* approach, developed independently, includes the following points.

Expressibility of the supremodel and the canonical model. SYNTHESIS canonical model is a language of specification of subject mediators as applications. Supermodel of *ModelGen* is not a complete language and serve as auxiliary instrument for the schema mapping. Metaconstructions of the canonical model kernel (SYNTHESIS language [6]) are not restricted by structured data models as *ModelGen.* The set of metaconstructions is much more comprehensive and includes object and nonobject abstract data types (ADT), frames and typed values constituting hybrid strictly typed and semistructured model, functions specified by signatures, pre- and postconditions, classes as sets of objects of specific type, subtype and subclass relations, associations between instances of ADT, metaclasses, type and class invariants expressed by logic formulae, processes. Supermodel constructions are easily included into constructions of the canonical model. Note that the proposed approach of information model mapping construction is quite general and does not depend on the specific canonical model kernel.

Extending of models. Extending of the canonical model essentially differs from extending of supermodel. Extending of the canonical model is a semantic process of introduction into model of new patterns of parameterized closed logical formulae expressing data dependencies, parameterized generic data types, metaframes annotating additional properties of constructions. A process of *ModelGen* supermodel extending is mainly a mechanical introduction of new metaconstructions.

Preserving of information in the process of model mapping. In *SYNTHESIS* the source information models are considered to be defined by respective languages with their syntax and semantics. *ModelGen* approach uses only data structure specifications. Detailed analysis of language semantics, integrity constraints, functions are not considered. This leads to information loss while data model mapping (for instance, during mapping of object models into relational one). *SYNTHESIS* approach is based on formal definition of semantics of complete schemas in source and target models. Thus proving of refinement of target schema by source schema becomes possible. Refinement relation is formally defined and its use allows to preserve the information and operations during the information models mapping and to avoid non-formally defined concepts (such as correspondence, specialization, subsuming of schemas).

Architecture differences. The Model Unifier infrastructure is developed in accordance with mentioned differences of *SYNTHESIS* and *ModelGen.* The process of mapping of source model into the canonical model includes definition of required extension of the kernel based on matching of constructions of source and canonical models, proving of correctness of the mapping, and construction of translator from source model into the canonical one with the help of metacompilation tools. Matching of constructions of source and canonical models as well as kernel extension are interactive creative processes realized by an expert supported by reference schemas related to constructions of source and target models. Structure and verbal annotations of reference schemas allow to establish similarity of constructions.

It should be clear that the approach proposed is much more general than ModelGen. Instead of choosing of a very limited set of structural data model facilities (ModelGen), it is required to select the whole information description and manipulation languages as the kernel of the target canonical model. We show that the level of generality of our approach is much greater than that of the ModelGen and other known approaches.

Alongside with the Meta Environment used in the current version of the Unifier it is possible to use other tools, namely implementations of QVT language. QVT (Query/View/Transformation) [18] is the OMG standard for model transformation having several analogues, for instance, ATL (ATLAS Transformation Language) [19]. In this framework source and target models should be expressed in M3 MOF metamodel or its analogues, for instance, Ecore of Eclipse Modeling Framework or KM3 — Kernel Meta Meta Model, developed by ATLAS INRIA group[19]). Model transformation is expressed by an expert using declarative–imperative language. Also transformation is a model itself expressed in M3 MOF (Ecore, KM3) metamodel.

Consequently existing approaches are characterized by lack (insufficiency) of accurate specifications of language semantics, lack of information model mapping verification (in MDA for instance), lack of synthesis of extensible unifying (canonical) model. Methods developed by the authors differ from existing ones by basing on extensible canonical model, constructed in a modular way extending the kernel, having strictly defined formal foundations, widely applying of refinement while unifying model transformation.

5 Conclusion

Complexity of various information models semantics makes integration and composition of heterogeneous resources hardly realizable. The only practical way is to map heterogeneous specifications to common, unifying model called the canonical one. Over a long period of time the authors have been developing the methods of canonical models synthesis for the wide range of real information models: structured, object, service, process including their arbitrary combinations. The methods developed preserve information and operations according to the refinement principle while mapping source models into the canonical one. With regard to large variety of information models the manual application of developed methods for real systems becomes inefficient.

New approach for heterogeneous information models mapping into the unified target model for the information resources integration in I-systems is proposed. The approach provides semi-automatic generation of provably correct refining mappings as respective compilers from the source information models into the target extensible canonical information model. Information model refinement as an important property of the mapping and extensibility of the canonical model kernel for obtaining refinement mappings are fundamental new contribution of our approach. The approach is independent of a canonical model kernel chosen. Selection of a kernel may depend on the purpose and environment of a particular I-system.

The paper presents architecture and functions of the Model Unifier allowing to reduce heterogeneous sources information models to canonical representation. The process of provable mapping of source information model into the canonical one is illustrated by the example of fragments of the Web Ontology Language (OWL). The kernel of the canonical model is constituted by hybrid object-frame language SYNTHESIS. The Model Unifier is being tested on various kinds of source information models. The Unifier is considered as a constituent part of the subject mediator middleware.

References

1. *Kalinichenko L. A., Briukhov D. O., Martynov D. O., Skvortsov N. A., Stupnikov S. A.* Mediation Framework for Enterprise Information System Infrastructures

// The 9th International Conference on Enterprise Information Systems (ICEIS). 2007.

2. *Kalinichenko L.A.* Method for Data Models Integration in the Common Paradigm // Advances in Databases and Information Systems: Proc. of the First East-European Conference. – St. Petersburg: Nevsky Dialekt, 1997. P. 275–284.
3. *Kalinichenko L. A.* Methods and tools for equivalent data model mapping construction // Proc. EDBT'90 Conference. – Springer-Verlag, 1990. P. 92–119.
4. *Kalinichenko L. A., Stupnikov S. A., Zemtsov N. A.* Extensible Canonical Process Model Synthesis Applying Formal Interpretation // East-European Conference ADBIS'05. – Springer, 2005.
5. *Kalinichenko L. A.* Canonical model development techniques aimed at semantic interoperability in the heterogeneous world of information modeling // Knowledge and model driven information systems engineering for networked organizations: Proc. of the CAiSE INTEROP Workshop. – Riga: Riga Technical University, 2004. P. 101–116.
6. *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. – M.: IPI RAS, 2007. – 171 p. – http://synthesis.ipi.ac.ru/synthesis/publications/07synthesis
7. *Van den Brand M. G. J. et al.* The ASF+SDF meta-environment: a component-based language development environment // Compiler Construction 2001 / Ed. by R. Wilhelm. – Springer, 2001. P. 365–370.
8. *Abrial J.-R.* The B-Book: Assigning Programs to Meanings. – Cambridge: Cambridge University Press, 1996.
9. The B-Toolkit. – http://www.b-kernel.com/ONLINEDOC/BToolkit.html
10. OWL Web Ontology Language Reference. W3C Recommendation. – http://www.w3.org/TR/owl-ref/, 2004.
11. *Patel-Schneider P. F., Hayes P., Horrocks I.* OWL Web Ontology Language Semantics and Abstract Syntax // W3C Recommendation, http://www.w3.org/TR/owl-semantics/, 2004.
12. *Atzeni P.* Schema and data translation: A personal perspective // 11th East European Conference ADBIS 2007. – Springer, 2007.
13. *Atzeni P., Cappellari P., Bernstein P.* ModelGen: Model Independent Schema Translation // 21st International Conference on Data Engineering, 2005.
14. *Hull R., King R.* Semantic database modeling: Survey, applications and research issues // ACM Computing Surveys, 1987, V. 19, 3.
15. *Bernstein P., Melnik S., Mork P.* Interactive Schema Translation with Instance-Level Mappings // 31st VLDB Conference, 2005.
16. *Barsalou T., Gangopadhyay D.* M(dm): an open framework for interoperation of multimodel multidatabase systems // ICDE 1992. – Los Alamitos: IEEE Computer Society Press, 1992.
17. *Haas L., et al.* Clio, 2005. Grows Up: From Research Prototype to Industrial Tool // Proc. of the ACM SIGMOD Conference, 2005, Baltimore, Maryland, USA.
18. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification // http://www.omg.org/cgi-bin/doc?ptc/2007-07-07, 2007.
19. ATL Project // http://www.eclipse.org/m2m/atl/