An Anatomy of the Information Resource Semantic Abstraction

Leonid Kalinichenko
Institute of Problems of Informatics
USSR Academy of Sciences
Vavilova 30/6, Moscow, V-334, 117900

Abstract

Semantic abstraction mapping establishing a correspondence between an information resource and an application is considered to be a basic notion providing for the study of semantic interoperability of heterogeneous information resources. The structure and necessary properties of a resource class application abstraction are considered. Intensional model-based properties of the abstraction mapping are introduced as provable conditions of a class assertion abstraction and an operation concretization.

1 Introduction

This paper is concentrated on one of the basic issues of the interoperable heterogeneous information resource environment design [5] - the application semantics of a resource.

To make the interoperable information resource environment a reality a combination of several outstanding abilities should be reached : an ability to homogenize heterogeneous information resource representations, an ability to organize proper reuse of existing resources, an ability to be sure that a given resource is applicable to a particular problem, that a combination of resources is composable in a consistently integrated collection and an ability of the dynamic design of an interoperable system capable to solve a given problem by means of a proper executive.

The way of an applicability problem solving is seen in keeping of a clean separate description of a real world (application domain) model as the basic semantic reference point. Such a real world model (or an application model) may be given in a spirit of a specification of requirements [3]. The interpretation of this model is a mapping to a set of states abstracted from real world states. From the other side, a generalized information resource description is considered which is interpreted as a mapping to a set of states of an information resource (e. g., of a database). An information resource semantics should be defined relatively to an application model. Semantic abstraction mappings (*Sem* and *Abs*) establishes a correspondence between an information resource and an application. *Sem* is a mapping of a generalized information resource description into an application model. *Abs* is a mapping of an information resource space of states into the space of states abstracted from a real world. Such information resource abstraction should provide for analysis of semantics discrepancies between different resources and between an application and a resource to be reused, for the semantic integration of a resource abstraction and of the application model, for the information system conceptual design on the basis of the pre-existing heterogeneous information resources.

The approach is not restricted to a single application model: semantic abstraction mappings may be introduced for different application domains simultaneously.

To make the process of analysis of the applicability formal and well-grounded we follow model-based database specifications [2] developed in Z-notation [8], Abstract Machines [1] to build a model of an application and of a resource representing its statics and dynamics. Using this formalism we provide a basis for provable reasoning on such notions as consistency of operations (proved by preservation of the invariants [2]), application/resource semantic assertion subsumption [7] (the subsumption theorem should be proved), application/resource functions (transactions) concretization (the concretization conditions based on the corresponding predicative specifications should be proved), and finally on the notion of concretization of an application by a resource.

Here we concentrate on the semantic abstraction of classes taking into account that an application as well as a resource are modelled in terms of classes of objects and their relationships that develop over time and are restricted by various constraints. The application model should be as abstract and as declarative as possible to be used as a reference point. This model is separated from the homogeneous description of the information resources though both are described within the framework of a single language (e.g. *SYNTHESIS* [4]) used in different styles. The styles differ mainly in the level of conceptual abstraction used and in the level of declarativity which should be generally much higher in the application model.

The paper is organized as follows. First we consider a structure of a resource class abstraction mapping taking into account pure syntactical issues. Then necessary properties of the constituents of the mapping are summarized in order that such mapping may be considered as a class abstraction. Finally intensional model-based abstraction properties of the mapping are introduced establishing provable conditions of a class

assertion subsumption [7] and an operation concretization.

2 A structure of a class abstraction mapping

In the information resource description or in the application model we consider types, classes and attributes to be the basic entities. We shall distinguish between object (or value) typed attributes and functional attributes (specifying class methods). A comprehensive set of basic types may be used including type of a tuple, type of a set, type of a union of types as well as an abstract data type constructor.

Behaviorally we consider separately a collection of assertions connected to an attribute and a collection of class assertions, each assertion being a predicate.

Operations and assertions we shall characterize by signatures of functions (predicates) and by predicative formal specifications [2,8].

Generally a class abstraction is a mapping $abs_c : C_R \rightarrow C_A$ (everywhere a lower index R denotes belonging to a resource entity, as well as a lower index A - belonging to an application entity). This mapping is a collection of functions estabishing the correspondence between resource level entities and their application counterparts: object identifier correspondence, class attribute correspondence, behavioral abstraction (class operation and assertion correspondence). For each pair of corresponding attributes of the resource and application level the functions estabishing attribute type, operation and assertion correspondence are introduced.

3 Properties of an application abstraction of a class

We consider separately structural abstraction properties, value abstraction properties and behavioral abstraction properties which should be satisfied in order for a class c_A to become an abstraction of a class c_R.

The structural abstraction properties state that 1) each attribute of c_A should have a corresponding attribute in c_R ; 2) for all pairs of corresponding nonfunctional attributes of c_R and c_A type of the resource class attribute should be a specialization (subtype) of the application class attribute type; 3) for all pairs of corresponding functional attributes of c_R and c_A the resource class function signature should be a specialization of the application class attribute function signature.

The value abstraction properties (which should be satisfied for every point in time) establish the relationship between extensions of corresponding classes or types (E. g., relationship between collections of the application and resource class object instances may be required to be bijective, injective, surjective, etc., leading to equivalence, inclusion or overlapping of the class extensions. The analogeous conditions should be established for the domain of each attribute type and its corresponding abstracion).

Th behavioral abstraction properties state that resource and application class assertions (operations) should be in one-to-one correspondence and for each pair of corresponding assertions (operations) the assertion (operation) of the resource should be a concretization of the assertion (operation) of the application.

4 Operation and assertion abstraction conditions

Semantic interpretation of an information resource is developed in frame of the model-based specifications using pre-, postconditions or predicate transformers. One of the basic ideas of such models consists in conceptual separation of the specification of data from the specification of a program. For a class it is possible to define the mathematical model of its data and of its operations. Specification of data makes possible to define static aspects of a class notion, including states of the class and invariants keeped by a class on a transfer from one state to another. The invariants are defined formally by means of the first-order logic and set theory.

Specification of the dynamic aspects includes possible operations leading to the changes of states. The specification of an operation is a strict description of properties which should be satisfied as a result of modification of the class attributes during the operations.

The condition that a possible state should exist after an operation related to the state before the operation is called the pre-condition of the operation. A state variable decorated with a dash denotes the state after the operation. The pre-condition of the operation Op is given by the expression [8]:

$$preOp = \exists s'_c \bullet spc(Op)$$

where $spc(Op)$ is a predicative specification written in the way of Z as one predicate combining both pre- and postcondition. s_c is the state of the class c [8].

To show that an operation of a resource class O_{a_R} is a concretization of the corresponding operation O_{a_A} of the application class when a resource class state and an application class state are related by an Abs mapping two following conditions (analogeous to those of [8]) should be proved. (Notice that an abstraction of an operation here is an inverse of its concretization).

The first condition should state that resource operation should terminate whenever an application operation is guaranteed to terminate:

$$\forall s_{c_R}, s_{c_A} \bullet preO_{a_A} \wedge Abs \Rightarrow preO_{a_R}$$

The second condition ensures that the state after the resource operation represents one of those abstract

states in which an application operation could terminate :

$$\forall s_{c_A}, s_{c_R}, s'_{c_R} \bullet preO_{a_A} \wedge Abs \wedge spc(O_{a_R}) \Rightarrow \exists s'_{c_A} \bullet Abs \wedge spc(O_{a_A})$$

For a resource assertion to be a concretization of an application assertion the following condition should be proved:

$$\forall s_{c_A}, s_{c_R} \bullet spc(p_{c_A}) \wedge Abs \Rightarrow spc(p_{c_R})$$

5 Class abstraction example

We use here an idea of the example presented in [7]. Several application class descriptions (instrum_A1 and instrum_A2) and a resource class instrum_R are considered. These classes have the same trade_price attribute with different semantics described in terms of trade_price_status and currency.

Simplified class descriptions are given in the *SYNTHESIS* language. It is sufficient to comment here that in *SYNTHESIS* every description is represented by a frame with a specialized slots. In the example class specifications are presented by frames.

For semantic description the specifications are enriched by metaclass price_sem introducing an *attribute category* with the same name characterized by attributes trade_price_ status and currency. Additionally in all classes the attributes trade_price by *in* attribute (in metaslot) are prescribed to belong to the attribute category introduced. The semantics of the attribute is enriched by an assertion related to that category in metaslot. In the assertions the attributes of the class and metaclass are used.

In this simple example for a resource to be an abstraction of an application it is sufficient to prove that an assertion related to trade_price in application class implies an assertion related to trade_price in resource class. Corresponding theorems are presented below.

Metaclass

```
{price_sem;
in : metaclass;
trade_price_status : string;
currency : string
};
```

Application class 1 description:

```
{instrum_A1;
in: class;
instrum_type : string;
instrum_name : string;
exchange : string;
trade_price : real;
metaslot
in : price_sem, {A1_price_assertion:
{instrum_type = 'Equity' & exchange = 'Madrid' →
trade_price_status = 'latest_nominal_price' &
currency = 'pesetas'}}
end;
};
```

Application class 2 description:

```
{instrum_A2;
in: class;
instrum_type : string;
instrum_name : string;
exchange : string;
trade_price: real;
metaslot
in: price_sem, {A2_price_assertion:
{exchange = 'Madrid' →
trade_price_status = 'latest_nominal_price' &
currency = 'pesetas' }}
end;
};
```

Resource class description :

```
{instrum_R;
in: class;
instrum_type : string;
instrum_name : string;
exchange : string;
trade_price: real;
metaslot
in: price_sem, {R_price_assertion:
{ (instrum_type='Equity'& exchange = 'Madrid' →
trade_price_status = 'latest_nominal_price' &
currency = 'pesetas') ∨
(instrum_type = 'Equity' & ¬ exchange = 'Madrid' →
trade_price_status = 'latest_price' &
currency = exchange.currency) ∨
(¬ (instrum_type = 'Equity') →
trade_price_status = 'latest_price' &
currency = exchange.currency)
}}
end;
};
```

Conditions of the resource assertion abstraction are formulated as the following theorems :

For an application class 1:

instrum_A1.trade_price.in.A1_price_assertion ⇒
instrum_R.trade_price.in.R_price_assertion

is proved to be true.

For an application class 2:

instrum_A2.trade_price.in.A2_price_assertion ⇒
instrum_R.trade_price.in.R_price_assertion

is false.

6 Conclusion

We introduced the notion of an application abstraction of an information resource modelled by a class. This notion is considered to be a basic issue for the study of the semantic interoperability of the heterogeneous information resources. A separately keeped application model is introduced as a primary semantic reference point. We showed how to construct (to prove) an application abstraction of an information resource preserving extensional and intensional properties of an application.

Following model-based specification methodology we use the notion of the class operation and assertion concretization (abstraction) which create the ground for the analysis of the resource behavioral semantic discrepancy and reconciliation.

The proposed approach is planned to use in the *SYNTHESIS* project for the resource semantic analysis and semantic enrichment, semantic conflicts and differences resolving, applicaton/resource schema transformation and integration as well as for the consistent conceptual design.

This approach may lead also to the development of well defined terminology agreed inside of the community conducting a research in the area of heterogeneous multidatabase interoperability. In particular, the notions of concretization, subsumption, specialization, equivalence of application/resource entities (and the like) may be strictly defined.

References

1. Abrial J.R. A formal approach to large software construction. In Mathematics of program construction, Proceedings of the International Cnoference, LNCS 375, Springer-Verlag, 1989.

2. Borgida A., Mertikas M., Schmidt J.W., Wetzel I. Specification and refinement of databases and transactions. DAIDA Deliverable, ESPRIT 892, Universitaet Hamburg, Germany, 1990.

3. Borgida A. et. al. Support for data-intensive applications: conceptual design and software development. University of Toronto, Technical Note CSRI-51, September 1989, p. 1 - 25.

4. L.A.Kalinichenko. SYNTHESIS : towards a query facility for generalized information resources. Proc. of the First East-West Workshop on Database Technology. Kiev, October 1990, LNCS 504, Springer Verlag, 1991

5. Kalinichenko L. The interoperable environment of heterogeneous information resources : a generalization perspective. Proc. of the First International Workshop on Interoperability in Multidatabase System, April 1991, Kyoto, Japan, p. 196 - 199

6. Lecluse C., Richard P., Velez F. O2, an object-oriented data model. ACM SIGMOD Record. 1988. Vol. 17, N. 3. p. 424-433.

7. Siegel M., Madnick S.E. Identification and reconciliation of semantic conflicts using metadata. MIT WP N 3102-89 MSA, November 1989

8. Spivey J.M. The Z Notation. A reference manual. Prentice-Hall, 1989