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
Fax : (095) 310 - 7050
E-mail: leonidk@ipian15.ipian.msk.su

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. An 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 [4] - 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. Then an information resource semantics may be defined relatively to such real world model (fig. 1).

```
                                        Real World
                                          States
                                             |
                                             | Abstraction
---------------                       _______V________
| Application |                       |   Abstract   |
|             |   AM Interpretation   |              |
|   model     |---------------------->|    states    |
|_____________|                       |______________|
       ^                                      ^
       |  Sem                                 |  Abs
_______|_______                       ________|________
| Generalized |                       |   Information |
| Information |   IR Interpretation   |    Resource   |
| Resource    |---------------------->|     States    |
| Description |                       |_______________|
|_____________|
```

Figure 1: An Information Resource Application Abstraction

Semantic abstraction mapping (Sem and Abs) establishes a correspondence between an information resource and an application. 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.

To make the process of analysis of the applicability formal and well-grounded we follow model-based specifications [2] developed in Z-notation [6], 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 [5] (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 in frame of one and the same language (e.g. *SYNTHESIS* [3]) used in different styles.

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 an intensional model-based abstraction properties of the mapping are introduced establishing provable conditions of a class assertion subsumption [5] and an operation concretization. A simple case of an application abstraction of a resource class is demonstrated by the example.

2 A structure of a class abstraction mapping

In the information resource desription or in the aplication model the following collections of basic entities are considered: 1) types $T = \{t_1, t_2, ...\}$, 2) classes $C = \{c_1, c_2, ...\}$, 3) attributes $A = \{a_1, a_2, ...\}$. To each class c a set Id_c of unique object identifiers (oid) corresponds.

An attribute a_i of a class c is considered to be a function $a_i : Id_c \rightarrow \{\mathcal{E}(t_i) \cup \mathcal{E}(c_i)\}$ where $\mathcal{E}$ denotes a set of value (an extension) of a type or a class (for a class c $\mathcal{E}$(c) is considered to be equivalent to Id_c). Functional attributes (class methods) are denoted by $a_i : Id_c \rightarrow O_i$, where O_i is a type of a function (operation), or a particular function description (O_i in this case is treated as a single-element set).

A particular set of type costructors here is not important, though it is assumed that a comprehensive set of basic types is 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 $P_a = (p_{a_1}, p_{a_2}, ...)$ and a collection of class assertions $P_c = (p_{c_1}, p_{c_2}, ...)$, each assertion being a predicate.

Operations and assertions we shall characterize by signatures of functions (predicates) denoted by *sign(o)* or *sign(p)* and by predicative formal specifications [2,6] denoted by *spc(o)* and by *spc(p)* .

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). The structure of this mapping is explained below.

For a particular pair of classes $(c_R, c_A) \in abs_c$ an attribute abstraction function is $abs_a : A_R \rightarrow A_A$, mapping a collection of the c_R attrtibutes into the collection of the c_A attributes. For the pair of classes an oid abstraction is $f_i : Id_{c_R} \rightarrow Id_{c_A}$.

For a pair of attributes $(a_R, a_A) \in abs_a$ we get :

- an abstraction of the value domain of an attribute type t_R:

  $f_d^t : D_{t_R} \rightarrow D_{t_A}$,

- an abstraction of the operations related to the type t_R:

  $f_o^t : O_{t_R} \rightarrow O_{t_A}$,

- an abstraction of the assertions related to an attribute a_R:

  $f_p^a : P_{a_R} \rightarrow P_{a_A}$.

For the sake of simplicity we do not make here any distinction between assertions having different enforcement conditions treating all the assertions as an invariants. In real situations it might be necessary to consider separate assertion abstractions for each kind of an enforcement condition.

A behavioral abstraction of a class c is represented by :

- an operational abstraction of the class (an abstraction of functions - values of functional attributes):

  $f_o^c : O_{a_R} \rightarrow O_{a_A}$,

- an abstraction of the class assertions :

  $f_p^c : P_{c_R} \rightarrow P_{c_A}$.

3 Properties of an application abstraction of a class

Here we postulate a conditions which should be satisfied in order that a class c_A becomes an abstraction of a class c_R.

The structural abstraction properties :

1. abs_a^{-1} should be a total function;
2. for all nonfunctional value-typed attributes such that $(a_R, a_A) \in abs_a$, type t_{a_R} should be a specialization (subtype) of a type t_{a_A} (various spezializations are explained in the appendix);
3. for all nonfunctional class-typed attributes such that $(a_R, a_A) \in abs_a$ class c_{a_R} should be a specialization (subclass) of a class c_{a_A};
4. for all functional attributes such that $(a_R, a_A) \in abs_a$ a signature $sign(o_{a_R})$ should be a specialization of a signature $sign(o_{a_A})$;
5. for all assertions such that $(p_{a_R}, p_{a_A}) \in f_p^a$ a signature $sign(p_{a_R})$ should be a specialization of a signature $sign(p_{a_A})$;
6. for all assertions such that $(p_{c_R}, p_{c_A}) \in f_p^c$ a signature $sign(p_{c_R})$ should be a specialization of a signature $sign(p_{c_A})$.

Value abstraction properties (should be satisfied for every point in time):

- f_i function should be bijective;

  In this case we emphasize an equivalence between the application and the resource object instances. Another relationships of collections of object instances may also be considered (e.g., total function (injective), partial function (surjective) leading to an inclusion or an overlapping of the class extensions).
- f_d^t function should be bijective (the same comments as to the item above are applicable here);
- all operations $(o_{t_R}, o_{t_A}) \in f_o^t$ should conform to the following condition: $f_d^t \circ o_{t_R} = o_{t_A} \circ f_d^t$.

Behavioral abstraction properties (should be satisfied for every point in time):

- $(f_o^c)^{-1}$ should be a total function;
- f_p^c should be bijective;

  This one and the next condition are stated here for simplicity reason. We assume here that application and resource assertions are equivalently structured. It is possible to get rid of this restriction considering more general abstractions of the conjunctive compositions of assertions related to an attribute or to a class.

- f_p^a should be bijective;
- for each pair of assertions $(p_{c_R}, p_{c_A}) \in f_p^c$ or $(p_{a_R}, p_{a_A}) \in f_p^a$ an assertion p_R should be a concretization of p_A (section 4);
- for each pair of operations $(o_{a_R}, o_{a_A}) \in f_o^c$ operation o_{a_R} should be a concretization of o_{a_A}(section 4).

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 operation related to the state before it 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 [6]:

$$preOp = \exists s_c' \bullet spc(Op)$$

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 [6].

To show that each operation o_{a_R} is a concretization of an operation o_{a_A} ($(o_{a_R}, o_{a_A}) \in f_o^c$) when a resource class state and an application class state are related by an Abs mapping (Abs mapping is given by f_i and a collection of f_d^t abstractions) two following conditions (similar to those formulated in [6]) should be proved. (Notice that an abstraction of the operations is an inverse of their 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 [5]. 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 as 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 the assertions the attributes of 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 descriptions:

class 1:

```
    {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;
};
```

class 2:

```
    {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. 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, Springer Verlag, 1991

4. 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

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

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

Appendix

Definition of structural specialization relationships

1. Type specialization

A type t_1 is a specialization of a type t_2 (written $t_1 \leq t_2$) iff one of the following conditions holds:

- t_1, t_2 are basic types and $t_1 \leq t_2$;
- t_1, t_2 are classes and t_1 is a subclass of t_2 (class specialization is defined below);

- t_1 has the form $[A_1 : t_{11}, ..., A_k : t_{1k}, ..., A_{k+p} : t_{k+p}]$, t_2 has the form $[A_1 : t_{21}, ..., A_k : t_{2k}]$ and $t_{1i} \leq t_{2i}$ for $1 \leq i \leq k$;
- t_1 has the form of set type $\{ t_{11} \}$, t_2 has the form $\{ t_{22} \}$ and $t_{11} \leq t_{22}$.

2. Class specialization

A class c is a structural specialization of class c_1 if to each attribute a_i of c an attribute a_j of c_1 corresponds such that a type of a_i is a specialization of a type of a_j or a signature of a_i (in case of a functional attribute) is a specialization of a signature of a_j.

3. Function signature specialization

A function signature is an expression of the form: $z = t_1 \times t_2 \times ... \times t_n \rightarrow t$, where $t_1, t_2, ..., t_n, t$ are types or classes. A function $m =< n, z >$, where n is a function name, z - is a signature. A signature model is a set of all partial functions of the form: $M(t_1) \times ... \times M(t_n) \rightarrow M(t)$ where $M(t_k)$ is an interpretation of a type t_k.

If f and g are two signatures, f is a specialization of g (or $f \leq g$) iff the model of f is included into the model of g .

4. Assertion signature specialization

An assertion signature is an expression of the form: $z = t_1 \times t_2 \times ... \times t_n \rightarrow Boolean$, where $t_1, t_2, ..., t_n$ are types or classes. A predicate $p =< n, z >$, where n is a predicate name, z - is a signature. A signature model is a set of all partial functions of the form: $M(t_1) \times ... \times M(t_n) \rightarrow \{true, false\}$ where $M(t_k)$ is an interpretation of a type t_k.

If f and g are two assertion signatures, f is a specialization of g (or $f \leq g$) iff the model of f is included into the model of g .