An approach to information integration based on the AMN formalism

Manuk Manukyan¹ and Grigor Gevorgyan²

¹ Yerevan State University, Yerevan 0025, Armenia, 
mgm@ysu.am

² Russian-Armenian (Slavonic) University, Yerevan 0051, Armenia, 
grigor.gevorgyan@gmail.com

Abstract. An original approach to integrate information is considered. In the frame of this approach an extensible canonical model is created which is based on the algebraic model of an advanced XML data model. We consider the canonical model as an intermediate model for creating our mediator and this model has been formalized by means of the Abstract Machine Notation (AMN). For each source model we formalize it by means of the AMN and create a reversible mapping into an extension of the canonical model. After this B-technology is used to prove that the AMN semantics of the source model represents a refinement of the AMN semantics of the extended canonical model. Hereby the correctness of mapping and the ability to use extended canonical model for representation of schemas of the source model are proved. Finally, in order to illustrate our approach a mapping from relational data model to canonical model has been created.

1 Introduction

The aim of information integration is to use two or more databases (data sources) in the frame of a large database, possibly virtual, containing information from all sources, so the data can be queried as a unit. Thus, a data integration system should provide the user a unified view, called global schema, of a set of heterogeneous data sources. Problems of information integration detail were discussed in literature, for example in [7, 21]. There are two approaches for heterogeneous data sources integration: Global-as-view (GAV) and Local-as-view (LAV). According to GAV [6, 16] a global schema is defined as a view over the data sources, while LAV [6, 16] assumes that data sources are defined as a view over the global schema. GAV approach has better query processing capabilities than LAV but LAV has better extensibility than GAV. To combine advantages of the mentioned techniques a mixed approach, called GLAV, is considered in [4]. There is a certain technique for compiling a GLAV system into an equivalent GAV one [3].

The theoretical basis of our approach to information integration are the works of the SYNTHESIS group (IPi RAS) [9–11], which are pioneers in the area of justifiable data model mapping for heterogeneous databases integration. In [11]
the basic definitions of equivalence of database states, database schemas and data models were introduced to preserve operations and information while constructing of mappings of various heterogeneous structured data models into the canonical one. According to this approach, each data model was defined by syntax and semantics of two languages - data definition language (DDL) and data manipulation language (DML). The main principle of mapping of an arbitrary resource data model into the target one (the canonical data model) could be reached under the condition, that the diagram of DDL (schemas) mapping and the diagram of DML (operators) mapping are commutative. In early works, to prove a commutativity of the diagrams the denotational semantics were used as a formalism (metamodel)[11]. Later, as formalism the AMN[1] was used instead of the denotational semantics. Instead of equivalence of respective specifications, it became possible to reason on their refinement[12]. 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. B - technology for AMN provides for conducting proofs of commutativity of model mappings semi-automatically. It is assumed that the canonical model should be extensible. A kernel of the canonical model is fixed. For each specific information model M of the environment an extension of the kernel is defined axiomatically 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 models M of the environment. The resource schema refines the canonical model schema. The refinement of the schema mapping is formally checked. In the frame of the considered formalism is offered construct: 1) Ms into an extension Mt mapping; 2) AMN semantics of Mt; 3) AMN semantics of the extended Mt. After that the B - technology is applied to proof that Ms is a refinement of the extension of Mt. The approach is supported by semi-automatic tools [14] and used for model unification and GLAV-based subject mediation of heterogeneous distributed information resources[2, 13]. In the frame of the proposed formalism a subject mediation approach is considered, for instance, in [2, 13]. For information integration a canonical model is introduced. The kernel of this model is a hybrid object-oriented/frame-based information model. Method and tools for semantic mapping of the kernel into the AMN are provided [20]. For any source model a mapping into an extension of the canonical model is defined. The source schema refines the canonical model schema. The refinement of the schema mapping is formally checked.

In this paper we follow the approach provided by SYNTHESIS group and introduce an extensible canonical model for information integration, having the algebraic model of an advanced XML data model (XML application)[17] as kernel. Our choice of using XML data model as a model for information integration is explained by the possibility of modeling principal concepts of conventional data models and semi-structured data by means of XML element directly. In the case of object-oriented model there are three principal concepts: the class (or its extent), the relationship and the method. Likewise, the object-relational model has two similar concepts: the attribute type (which includes classes) and
relation. On the XML data model level we can blend these concepts, much as relational model blends entity sets and relationships. However, the motivation for the blending appears to be different in each case. While the relational model owes some of its success to the fact it facilitates the efficient implementation, interest in the XML data model is motivated primarily by its flexibility. The reasoning considered above fully coincides with the analogous reasoning for semistructured-data model proposed in [5] as an information integration model.

According to our approach to information integration, for any source model a reversible mapping into an extension of the canonical model is defined. The basic formalism for the construction of a mediator is the AMN. Data models, their properties and mappings from one model to another are represented in AMN by means of corresponding specifications. First of all the kernel of the canonical model is fixed, representing the AMN semantics for the target data model. For arbitrary data model of independent information source, in order to integrate it into a virtual database, are constructed corresponding AMN semantics and canonical model extension by concepts of the source model, which are not yet present in target model. After this B-technology is used to prove that the AMN semantics of the source model represents a refinement of the AMN semantics of extended canonical model. Hereby the correctness of mapping and the ability to use extended canonical model for representation of schemas of the source model are proved. In this case is offered construct: 1) AMN semantics of $M_s$; 2) the canonical model extension by concepts of the source model which are not yet present in target model. After that the $B$ - technology is applied to proof that $M_s$ is a refinement of the extension of $M_t$. Finally, based on the hierarchy of abstract machines we generate mapping from source model to the target model.

As in [2, 13, 15] we introduce an extensible canonical model also. In our case the kernel of the canonical model is the algebraic model of our XML data model (XDM). We define AMN semantic for XDM (mapping from XDM to AMN). Our approach assumes the creation of AMN semantics for any source model (mapping from source model to AMN) developing further the approach presented in [12]. We apply B-technology to prove the fact that source model is a refinement of the canonical model. Based on the intermediate canonical model, we construct global schema as an XML document, which is an instance of XDM DTD. The distinguishing feature of our approach is that we perform verification of mapping on the level of data models (but not on the level of schemas). Such approach is explained by the fact that if the source model is a refinement of target model then for each correct schema of the source model there exists an equivalent correct schema in the target model.

The paper organized as follows: An overview of the AMN and extensible data model is considered in Section 2. The principle of kernel extension is introduced in Section 3. An approach to information integration is proposed in Section 4. A short discussion about DML formalization principles in AMN is considered in Section 5. The conclusion is provided in Section 6.
2 Preliminaries

2.1 AMN: A formal language of specification

AMN is a state-oriented formalism for software development which consists of the following notations: the logical notation, the basic set notation, the relational notation, the mathematical object notation, and the generalized substitution notation. In the frame of AMN specifications, refinements and implementations are presented as abstract machines. AMN allows to define a context of global constraints, given constants, abstract sets (their properties), a list of state variables and operations on them, the latter having optional preconditions, parameters and results. Operations of abstract machines are based on generalized substitutions describing the conversions between states of the system. Each generalized substitution \( S \) is defined as a predicate transformer which transforms a post-condition \( R \) into the weakest precondition that guarantees the preservation of \( R \) after the operation execution. The information defining the state of a machine is contained in local variables, and general constraint about the state of a machine can be expressed by a proposition called invariant. The key concept of AMN is refinement which is formalized by means of so-called proof obligations. These proof obligations allow us to verify that:

- The new invariant condition does not contradict with the invariant condition already given;
- The initial state satisfies the invariant;
- The invariant is preserved by the operations.

Finally, large software development is supported by using several composition mechanisms, i.e. SEES, USES, INCLUDES, EXTENDS. They give different access privileges to the operations or to the local variables of an external machine.

2.2 XDM: An XML data model

XDM is a result of an extension of the XML data model semantics to support the DB concept. In this model basic and compound objects are considered. Examples of basic objects are integers, strings, variables of different sorts, symbols (for instance, reserved words). The compound objects are defined in terms of binding and application in \( \lambda \)-calculus [8]. The type system is built from a basis of types that are defined by themselves and certain recursive rules, whereby the compound types are built from simpler types. The basis consists of the conventional atomic types (for example, integer, string, boolean, etc.). We use the following type constructors for building compound types:

- **Attribution.** If \( v \) is a basic object variable and \( t \) is a typed object, then \( \text{attribution}(v, \text{type } t) \) is typed object. It denotes a variable with type \( t \).
- **Abstraction.** If \( v \) is a basic object variable and \( t, A \) are typed objects, then \( \text{binding}(\text{lambda, attribution}(v, \text{type } t), A) \) is typed object.
- **Application.** If \( F \) and \( A \) are typed objects, then \( \text{application}(F, A) \) is typed object.
• Function Space. If \( t \) and \( u \) are typed objects, and \( v \) is a basic object variable, then \( \text{binding}(PType, \text{attribution}(v, \text{type } t), u) \) is typed object. It represents the type of functions mapping an argument \( v \) of type \( t \) to a result of type \( u \).

Adding applicative syntax and semantics substantially increases the expressiveness of the XML data model. For this data model a declarative query language (element calculus)\(^3\) is developed \([17]\). We can combine the possibility of the applicative programming with element calculus to give very complex integrity constraints on the global schema level. The necessity to give such integrity constraints is arising as the data at the sources may not satisfy the constraints of the global schema. The considered approach of constructing AMN semantics for the algebraic model of XDM is based on the following definition of XDM schema \([18,19]\):

**Definition 1** We say that \( S \) is a schema, if
\[ S = \langle \text{name}, \text{type}^4, f \rangle \text{ or } S = \langle \text{name}, \text{typeOp}(S_1, S_2, \ldots, S_n), f \rangle, \text{ and } S_i \text{ is a schema}^5, \text{ where } \text{typeOp} \in \{\text{sequence, choice, all}\}^6, f \in \{?, *, +, \perp\}^7, 1 \leq i \leq n. \]

Formal definition of algebraic operations in \([18,19]\) is provided also. Finally, our algebra is closed with respect to the XDM-element \([18,19]\).

### 3 Kernel extension principle

The canonical model must be extensible. The extension of the canonical model is formed by consideration of each new data model by adding new symbols (constructions) to its DDL to define logical data dependencies of the source model in terms of the target model if necessary. In XDM symbols are used for data models concepts representation. The extension result must be equivalent to the source data model. For applying a concept on the canonical model level the following rule is proposed:

Concept ← Symbol ContextDefinition

For example, to support the concepts of referential integrity and key’s of relational data model, we have expanded the kernel with the following symbols: key, unique, foreign key, constraint, on update, on delete, cascade, and set null. In general, the kernel extension is reduced to introduce new symbols in so-called content dictionaries (XML document) of XDM and to define the context in which these symbols are applied. It is essential that we use a computationally complete language to define the context. Content dictionaries are used to assign

---

3 The detailed discussion of the query language of the mediator is beyond the topic of this paper.
4 Atomic type or function type.
5 The attributes are not considered for simplicity.
6 sequence, choice, all have similar semantics as in XML Schema.
7 ? , *, + have similar semantics as in XML, a \( \perp \) following an object means that the object may occur exactly one time.
informal and formal semantics to all concepts used in the data models. As a result of this approach usage of new symbols in the DDL does not lead to change DDL parser change.

4 AMN formalization of data models

For each data model some abstract machine (or abstract machines hierarchy) is created which represents AMN semantics for that model. Abstract machine $Mch_M$ for a data model $M$ should be constructed so that the set of schemas in model $M$ is in a bijection with the set of acceptable states of machine $Mch_M$, constrained by its invariant $I_M$. First of all we introduce an abstract machine $DatamodelContext$ in which some common concepts of data models are included:

MACHINE
DatamodelContext
SETS
OBJ; Types
ABSTRACT CONSTANTS
atomicTypes, extent, typeOf
PROPERTIES
atomicTypes $\in \mathcal{P}(Types)$ $\land$ extent $\in$ Types $\rightarrow \mathcal{P}(OBJ)$ $\land$
 typeOf $\in$ OBJ $\rightarrow$ Types
END

The $SETS$ clause contains the following abstract sets:
- $OBJ$: the set of all possible objects;
- $Types$: the set of all types.
In $ABSTRACT CONSTANTS$ clause the following abstract constants are introduced:
- $atomicTypes$: the set of atomic types, for instance, $integer$, $string$ etc;
- $extent$: an injection, meaning the extent of the type, i.e. the set of all possible values a variable of the given type can take;
- $typeOf$: a surjection, mapping to each object its type;

4.1 AMN semantics for XDM

AMN semantics for XDM data model is defined in two steps. First, abstract machine $XDM\_Context$ is constructed which contains abstract constants $ext\_XDM$, representing set of all well-formed XDM schemas according to the Definition 1, and $function\_Space$, representing the space of all possible functions. Below is given the definition of this machine:

\[\text{Without loss of generality, in order to visual simplicity, we will use pseudo-AMN for defining AMN semantics for data models.}\]
MACHINE XDM_Context
INCLUDES DatamodelContext
ABSTRACT CONSTANTS ext_XDM, functionSpace
PROPERTIES
  ext_XDM ∈ ℙ(OBJ) ∧ functionSpace = seq(Types) × Types
END

Second, we construct an abstract machine \textit{XDM\_Schema} which states represent all well-formed XDM schemas according to the Definition 1:

MACHINE XDM\_Schema
  (p\_name, p\_value, p\_operationType, p\_freq, p\_recursiveSequence)
INCLUDES XDM_Context
CONSTANTS
  frequency, operationTypes
PROPERTIES
  frequency = \{?, *, +, ⊥\} ∧ operationTypes = \{sequence, choice, all, empty\}
CONSTRAINTS
  p\_name ∈ STRING ∧ p\_value ∈ atomicTypes ∪ functionSpace ∪ seq(ext\_XDM)
  ∪ ext\_XDM ∪ ℙ(ext\_XDM) ∧ p\_operationType ∈ operationTypes ∧
  p\_freq ∈ frequency ∧ p\_recursiveSequence ∈ ℙ(ext\_XDM)
VARIABLES
  name, valueType, operationType, freq, recursiveSequence
INITIALISATION
  name := p\_name || valueType := p\_value || operationType := p\_operationType
  || freq := p\_freq || recursiveSequence := p\_recursiveSequence
INVARIANT
  name ∈ STRING ∧ valueType ∈ atomicTypes ∪ functionSpace ∪
  seq(ext\_XDM) ∪ ext\_XDM ∪ ℙ(ext\_XDM) ∧ operationType ∈ operationTypes
  ∧ freq ∈ frequency ∧ recursiveSequence ∈ ℙ(ext\_XDM) ∧
  ContentSpec(valueType, operationType, recursiveSequence)
END

According to the Definition 1, the considered variables in the XDM\_Schema machine represent correspondingly:

- the name of the schema;
- the value Type of the second parameter;
- the operation type of the second parameter when it is neither atomic type nor function type;
- the frequency of the object;
- the recursive sequence of schemas \( S_1, S_2, ..., S_n \) which is used in the case when operationType ∈ operationTypes;
- the value of construction (ContentSpec) is AMN-definition of XDM-element’s structure according to Definition 1.

4.2 The idea of mapping the source model into target model

In terms of AMN the data models mapping is performed as follows. Let for the source model $M_S$ and the target model $M_T$ already be constructed corresponding abstract machines $Mch_S$ and $Mch_T$. A new abstract machine $M_{S \to T}$-mapping is defined which contains an operation $M_{S \to T}$ describing the mapping of the state of $Mch_S$ to some state of $Mch_T$. If necessary, additional operations may be defined.

4.3 Extension of the canonical model

Let the abstract machine $Mch_T$ for the canonical model be constructed. For a source model $S$ we construct a new machine $Concepts_{ST}$, representing concepts of the source model that are new to the target model and cannot yet be represented by its means. These concepts are expressed as sets and constants of machine $Concepts_{ST}$. Also we create a machine $Mch_{ST}$, which is obtained from $Mch_T$ by adding $Concepts_{ST}$ to its SEES clause. In other words, we extend the AMN representation of canonical model with concepts intrinsic to the source model.

```
MACHINE Mch_T
SETS Sets T
CONSTANTS Const T
PROPERTIES P T
VARIABLES Var T
INITIALISATION Init T
INVARIANT I T
END

MACHINE Concepts_{ST}
SETS Sets_{ST}
CONSTANTS Const_{ST}
PROPERTIES P_{ST}
END

MACHINE Mch_{ST}
SETS Sets S
EXTENDS Mch T
SEES Concepts_{ST}
VARIABLES Var S
SEES Concepts_{ST}
INITIALISATION Init S
INVARIANT I_S \land Cond S
END
```

After this, we construct another abstract machine, $Mch_S$, which is a refinement of $Mch_{ST}$, in such a way that the set of it is states is in a bijection with the set of all valid schemas in $S$, and that bijection is provided by some mapping rules from model $S$ to model $T$ (such mapping rules can be constructed in terms of AMN as described in 4.2). This can be done by adding some additional conditions $Cond S$ to the invariant, narrowing the set of machine’s state to the necessary.

```
REFINEMENT Mch_S
REFINES Mch_{ST}
SEES Concepts_{ST}
SETS Sets S
CONSTANTS Const S
PROPERTIES P S
VARIABLES Var S
SEES Concepts_{ST}
INITIALISATION Init S
INVARIANT I_S \land Cond S
END
```

Now if the conditions of proof obligations are satisfied for this refinement, we can conclude that the constructed extended canonical model machine can be used to represent any schema of the source model.
4.4 Mediator database schema construction algorithm

The virtual integrated database is constructed iteratively. First the mediator kernel is constructed which represents AMN semantics of the canonical model. Then sequential joining of the information sources to the existing virtual integrated database in terms of AMN is performed. Each data source is integrated into the existing mediator as follows: If the source model is not yet supported by mediator, the extension of the existing virtual database specification is constructed as described in 4.3. Then the state of the extended specification corresponding to the source model and the state corresponding to the mediator schema at the moment are joined into the resulting state of the canonical model’s abstract machine. The obtained state represents the resulting integrated database schema.

4.5 AMN semantics for relational data model

We shall define AMN semantics for the relational data model via abstract machines hierarchy, consisting of specifications REL_Context, REL_Schema and REL_DB_Schema.

MACHINE
   REL_Context
   INCLUDES
   DatamodelContext
   ABSTRACT_CONSTANTS
   ext_REL_Sch, ext_REL_DB_Sch
   PROPERTIES
   ext_REL_Sch ∈ P(OBJ) ∧ ext_REL_DB_Sch ∈ P(OBJ)
END

Here ext_REL_Sch and ext_REL_DB_Sch represent sets of all correctly constructed relation schemas and schemas of relational databases correspondingly.

MACHINE
   REL_Schema
   VARIABLES
   schemaName, state
   INVARIANT
   schemaName ∈ STRING ∧ state ∈ seq(STRING × atomicTypes)
END

In the considered abstract machine REL_Schema the variable state represents the list of attributes of the schema, each element containing attribute name and type. This abstract machine represents relation schemas. Variables schemaName and state specify the state of the machine which corresponds to exactly one relation schema.
MACHINE  
REL_DB_Schema  
INCLUDES  
REL_Context  
VARIABLES  
dbSchemaName, schemas  
INVARIANT  
dbSchemaName ∈ STRING ∧ schemas ∈ seq(ext_REL_Schema)  
END  

In above considered abstract machine we include REL_Context machine to use the ext_REL_Schema set. The variable schemas represents the list of relation schemas which form the relational database schema. Finally, the abstract machine REL_DB_Schema represents relational database schemas. The state of this machine is determined by the variables dbSchemaName and schemas and uniquely identifies some correctly constructed relational database schema.

4.6 Mapping from relational data model into XDM

Mapping from relational data model to XDM is performed by defining an abstract machine REL_to_XDM_mapping, containing operations REL_attr_to_XDM, REL_Schema_to_XDM, REL_DB_Schema_to_XDM, which correspondingly represent mappings of single attribute, relation schema, and schema of relational database to XDM schemas.

MACHINE  
REL_to_XDM_mapping  
USES  
REL_Schema, REL_DB_Schema, XDM_Schema  
OPERATIONS  
xdmSchema ← REL_attr_to_XDM(attr)=  
PRE  
attr ∈ string × atomicTypes ∧ xdmSchema ∈ ext_XDM  
THEN  
xdmSchema = XDM_Schema(dom(attr), ran(attr), empty, ⊥, <>)
END

xdmSchema ← REL_Schema_to_XDM(relSchema)=  
PRE  
relSchema ∈ ext_REL_Sch ∧ xdmSchema ∈ ext_XDM
THEN  
xdmSchema = XDM_Schema(name, valueType, operationType, frequency, recursiveSequence)
END

xdmSchema ← REL_DB_Schema_to_XDM (relDbSchema)=  
PRE  
relDbSchema ∈ ext_REL_DB_Sch ∧ xdmSchema ∈ ext_XDM
5 A short discussion about DML formalization principles in AMN

To reflect data manipulation in AMN we need to have an appropriate DML representation for each model considered. In case of relational model this can be done by defining the `Relation` machine to represent relations, and `RelationalAlgebra` machine, which implements relational algebra operations as methods manipulating with `Relation` type objects:

**MACHINE**

`RelationalAlgebra`

**USES**

`Relation`

**OPERATIONS**

\[ s ← \text{union}(r, q) : \]

**PRE:**

\[ r \in \text{extRelation} \land q \in \text{extRelation} \land s \in \text{extRelation} \land \]

\[ \text{isUnionCompatibility}(r,q) \]

**THEN:**

\[ s\text{.schema} := r\text{.schema} || s\text{.tuples} = r\text{.tuples} \cup q\text{.tuples} \]

END

...

END

The rest of relational operations can be defined in the same manner as shown for `union`. Any query can be implemented by superposition these operations. For XDM model an abstract machine `ElementAlgebra` can be constructed, implementing operations on xdm-elements as proposed in [18,19]. Adding mapping rules representing source model DML operations by means of target model DML operations allows to translate queries from source model to target on AMN level. A detailed consideration of the principles of formalization of DML beyond the scope of this paper.

6 Conclusions

In this paper a mediator for arbitrary heterogeneous information sources integration which is based on the AMN formalism is presented. In the frame of this mediator an extensible canonical model is accepted. The kernel of this model is the algebraic model of XDM. The considered XML data model is a result of
extension of the conventional XML data model by means of the applicative syntax and semantics which substantially increases the expressiveness of the XML data model. We consider the canonical model as an intermediate model for creating our mediator and this model has been formalized by means of the AMN. This formalism allowed to define the model-theoretic specifications in the first order logics and to prove the fact of specification refinement. In the frame of our approach the canonical model by means of a hierarchy of abstract machines is represented. Data models, their properties and mappings from one model to another have been defined by means of these abstract machines. In the top level abstract machine common concepts of data models have been defined. The next abstract machine in this hierarchy represents AMN semantics for XDM. By means of the next abstract machines in this hierarchy we define AMN semantics for source models and mappings from source models to an extension of the canonical model. In our case the canonical model is defined as kernel + \{new concepts\}. We apply B-technology to prove the fact that the source model is a refinement of the extension of the canonical model. Herby the correctness of mapping and the ability to use extended canonical model for representation of schemas of the source model are proved. Based on the intermediate canonical model, we construct global schema as an XML document, which is an instance of XDM DTD. The distinguishing feature of our approach is that we perform verification of mapping on the level of data models (but not on the level of schemas). Such approach is explained by the fact that if the source model is a refinement of the target model then for each correct schema of the source model there exists an equivalent correct schema in the target model. Finally, in order to illustrate our approach a mapping from relational data model to canonical model has been created.

Acknowledgements

The authors would like to express their gratitude to members of SYNTHESIS group, Prof. Leonid Kalinichenko and Sergey Stupnikov, for constructive critics, tips and scientific support. As the result of our discussion the paper quality has essentially improved.

References