Model-Based Mediation with Domain Maps

Bertram Ludäscher* Amarnath Gupta* Maryann E. Martone‡

*San Diego Supercomputer Center, {ludaesch,gupta}@sdsc.edu
‡Department of Neurosciences, mmartone@ucsd.edu
University of California San Diego

Abstract

We propose an extension to current view-based mediator systems called *model-based mediation*, in which views are defined *and executed* at the level of conceptual models (CMs) rather than at the structural level. Structural integration and lifting of data to the conceptual level is "pushed down" from the mediator to wrappers which in our system export classes, associations, constraints, and query capabilities of a source. Another novel feature of our architecture is the use of *domain maps*, semantic nets of concepts and relationships that are used to mediate across sources from *multiple worlds* (i.e., whose data are related in *indirect* and often complex ways). As part of registering a source's CM with the mediator, the wrapper creates a "semantic index" of its data into the domain map. We show that these indexes not only semantically correlate the multiple worlds data and thereby support the definition of the integrated CM, but that they are also useful during query processing, for example, to select relevant sources. A first prototype of the system has been implemented for a complex Neuroscience mediation problem.

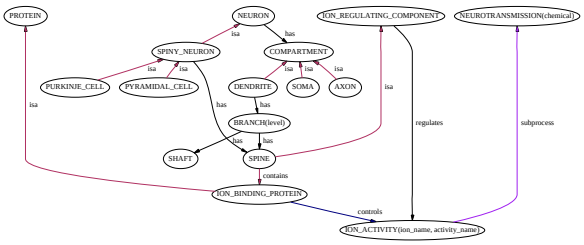
1 Introduction

Mediator systems federate and integrate data from disparate sources in order to elicit information that the individual sources cannot provide independently. Currently, the "standard" mediator architecture employs wrappers that translate heterogenous source data into a common (often semistructured) data model like XML. A "mediation engineer" provides an integrated view definition (IVD) on the wrapped XML sources. In such a system, an IVD is ideally expressed in a declarative query language for XML or semistructured data. When developing the IVD, an XML query language provides the mediation engineer only with a *tree-structured* model of the source, i.e., the names and possible nesting structure of XML elements as defined by an XML DTD, but gives no hint on semantic relationships, class structures, not to mention application domain specific constraints.

We argue that such a mediator architecture based solely on an XML-like semistructured data model, while very powerful and useful in simple "one world scenarios" (say comparison shopping with `amazon.com` and `barnesandnoble.com`), is not adequate when mediating across complex sources whose data comes from seemingly disjoint "worlds":

Example 1 (Two Neuroscience Worlds) Consider two Neuroscience laboratories[1] that perform experiments on two different brain regions. The first laboratory, SYNAPSE, studies dendritic spines of pyramidal cells in the hippocampus. The primary schema elements are

[1] see `synapses.bu.edu` (SYNAPSE) and `www-ncmir.ucsd.edu` (NCMIR)



AXON, DENDRITE, SOMA $\sqsubset \exists$isa.COMPARTMENT
ION_BINDING_PROTEIN $\sqsubset \exists$isa.PROTEIN
ION_REGULATING_COMPONENT $\sqsubset \exists$isa.SPINE
PURKINJE_CELL, PYRAMIDAL_CELL $\sqsubset \exists$isa.SPINY_NEURON
SPINY_NEURON $\sqsubset \exists$ isa.NEURON

BRANCH(...) $\sqsubset \exists$has.{SHAFT,SPINE}
DENDRITE $\sqsubset \exists$has.BRANCH(...)
ION_ACTIVITY(...) $\sqsubset \exists$subprocess.NEUROTRANSMISSION(...)
ION_BINDING_PROTEIN $\sqsubset \exists$controls.ION_ACTIVITY(...)
ION_REGULATING_COMPONENT $\sqsubset \exists$regulates.ION_ACTIVITY(...)
NEURON $\sqsubset \exists$has.COMPARTMENT
SPINE $\sqsubset \exists$contains.ION_BINDING_PROTEIN
SPINY_NEURON $\sqsubset \exists$has.SPINE

Figure 1: A *domain map* of additional knowledge for linking SYNAPSE and NCMIR

thus the anatomical entites that are reconstructed from 3-dimensional serial-sections. For each entity (such as spines, post-synaptic density, shafts and dendrites), researchers make a number of measurements, and study how these measurements change across age and species under several experimental conditions.

In contrast, the NCMIR laboratory studies the Purkinje Cells of the cerebellum, inspecting the branching patterns from the dendrites of filled neurons, and localization of various proteins in neuron compartments. The schema used by this group consists of a number of measurements of the dendrite branches (such as diameter of a segment, thickness of branching points, number of branches at every split-level of the dendrites) and the amount of different proteins found in each of these subdivisions. Let us assume each of the two schemas has a class C having a location attribute which can have a value like "Pyramidal Cell dendrite" and "Purkinje Cell", respectively. □

How are the schemas of SYNAPSE and NCMIR related? Evidently they carry distinctly different information and do not even enter the purview of the schema conflicts usually studied in databases [KS96]. To the scientist however, they are related because of the following reason: Release of calcium from spiny dendrites occurs as a result of neurotransmission and results in changes in spine morphology (sizes and shapes obtained from SYNAPSE). Propagation of calcium signals throughout a neuron depends upon the morphology of the dendrites, the distribution of calcium stores in a neuron and the distribution of calcium binding proteins, whose subcellular distribution for Purkinje cells are measured by NCMIR.

Thus, a researcher who wanted to model the effects of neurotransmission in hippocampal spines would get structural information on hippocampal spines from SYNAPSE and information about the types of calcium binding proteins found in spines from NCMIR. Note that in order to connect the two sources, we need, independent of the observed experimental source data, *additional domain knowledge* like the following:

Pyramidal cells and Purkinje cells have dendrites that have higher-order branches that contain spines. Spiny dendrites are ion (calcium) regulating instruments. Spines have ion (calcium) binding proteins. Neurotransmission involves ionic (calcium) activity (release). Ion-binding proteins control ion (calcium) activity (propagation) in a cell. Ion-regulating components of cells affect ionic (calcium) activity (release).

Figure 1 shows a representation of such knowledge in the form of a *domain map* (left), a kind of semantic net that is used for defining and executing IVDs at the mediator; its logical semantics is shown on the right (cf. Definition 1). The above example from a real-world integration scenario illustrates a fundamental difference in the nature of information integration as studied in most of the database literature and as necessary for scientific data management. In the latter, seemingly unconnected schema can be semantically close *when situated*

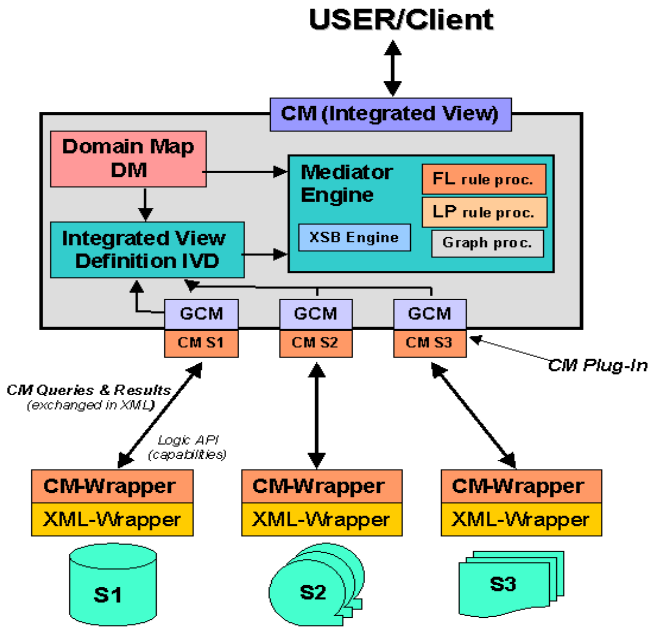


Figure 2: Architecture of a model-based mediator

in the scientific context, which in this case is the neuroanatomy and neurophysiological setting described above. We call this *mediation across multiple worlds*.

In this paper we develop a novel model-based mediator architecture for such mediation scenarios, define its formal framework, and sketch query processing in our system prototype.

The outline and main contributions are as follows: Section 2 presents the overall architecture: schema definition *and data access* of sources is lifted from the structural to the *conceptual level*, thereby facilitating the mediation engineer's task of developing IVDs. The architecture is independent of a specific formalism for conceptual models (CMs) due to a special CM *plug-in mechanism*, which is based on the underlying XML transport syntax and a generic meta-model for CMs (Section 3). Mediation across multiple worlds is facilitated by incorporating an expert knowledge base called *domain map* (DM) at the mediator. DMs are related to semantic nets and ontologies, but (i) have a formal semantics, (ii) are more powerful due to rule-based extensions, and (iii) can be "executed" during query execution of IVDs (Section 5). Section 6 concludes with a discussion on related work.

2 Model-Based Mediation Architecture

Figure 2 depicts a system architecture for model-based mediation: As is standard for mediator architectures, differences in the sources' data models are resolved by wrappers that translate the raw data into a common generic data format (XML) on which current mediator systems would directly define the integrated views using an XML query language [LPV00]. We extend this architecture by lifting exported source data from the level of uninterpreted, semistructured data in XML syntax to the semantically rich level of *conceptual models with domain knowledge* (CMs). In this way, the mediator's complex task of defining integrated views over heterogeneous sources becomes more manageable since class hierarchies, object structure, properties of relationships (relational constraints, cardinalities, ...), and in particular domain specific constraints of sources all become accessible for view definition at the mediator. The mediator's view definition language in such a model-based architecture is

not only a semistructured query language (like Lorel, YATL, XML-QL, Quilt, ...), but also a declarative, *query language for conceptual models* that can express *complex schema and instance level transformations* and check *logical constraints*.

Rather than reinventing yet another variant of conceptual models, we use a simple, generic conceptual model GCM[2] at the mediator level. The wrappers can export their "CM-lifted" source data either directly in GCM, or in any standard CM formalism like (E)ER, UML, OMR, RDF, etc. for which a CM-to-GCM *plug-in* has been provided. Syntactically all information (queries, CM signatures and data, mediator/wrapper dialogues, etc.) goes "over the wire" in XML syntax. Therefore the mediator also includes an XML sublanguage for translating between XML and the mediator's local GCM representation making the system completely independent of the chosen XML syntax for exchanging CMs. The KIND mediator prototype (Section 5) is based on an object-oriented logic programming system for realizing practically all of the above tasks.

The Mediator System at Work. At runtime, a wrapped source S can join the mediated system by *registering* its conceptual model CM(S) with the mediator M. This requires that S sends the mediator descriptions of the exported *class schemas*, *relationship schemas*, and *semantic rules* that are evaluable at the mediator (e.g., for defining virtual classes and relationships, CM-specific constraints, or arbitrary domain-specific constraints). Apart from this schema level information, S also transmits a description of its *query capabilities* to M, which is a (usually very limited) CM query language that can be seen as the "logical API" for retrieving actual object instances of CM(S). The query capability descriptions minimally specify means (e.g., primary keys) for browsing through all instances of exported classes and relations, and optionally declare further capabilities as *binding patterns* or *query templates* which allow the mediator to optimize query evaluation by "pushing down" subqueries and computations to the wrapper.

The exported objects of a CM(S) can have special anchor and context attributes[3] that provide the "semantic coordinates" of the data in the mediator's *domain map* DM(M). Recall from the introduction that although the mediator M "sees" all source data at the conceptual level CM(S_i), for multiple world scenarios, there is typically little or no overlap in the concepts of CM(S_i). The role of the domain map DM(M) is thus to provide a declarative means for specifying *additional knowledge* that is *not present* in the sources but that can be used (like a road map) to navigate through and interrelate the multiple source worlds.

CM Plug-In Mechanism. One goal of our model-based mediation architecture is to make the mediator independent of a source's current or future choice of CM formalism for communicating conceptual-level schema and data.

As a first step, we require that sources export all CM information (i.e., at the schema and instance level) in XML. For example, CMs formalized in XML Schema or RDF Schema come directly in XML syntax.[4] For other formalisms like (E)ER, ORM, UML class diagrams etc. XML exchange formats are available or can be easily defined. For each such format the mediator system needs to have a specific system component and interface, including say an XMI-API for handling UML models expressed in XMI [XMI99].

[2]Strictly speaking, GCM is a meta-model for CMs.

[3]or *methods*, i.e., *derived attributes* which are computed on demand at the mediator.

[4]see `www.w3.org/TR/xmlschema-{0,1,2}` and `www.w3.org/TR/rdf-schema/`

A second and crucial step for the plug-in mechanism is to devise a meta-model called GCM (*Generic Conceptual Model*) that is *universal* in the sense that any conceivable CM formalism can be expressed in it. Now the crux of the plug-in mechanism is that the mediator no longer needs one module for each CM formalism. Instead a new CM formalism say UXF [SY98] is added to the system by simply plugging an UXF-2-GCM translator into the mediator. Essentially such a translator is nothing more than a *complex XML query expression that a source sends once to the mediator* when a new CM is introduced. For example, a UXF-2-GCM translator is an XML query that maps XML documents conforming to the UXF DTD to their equivalent GCM representations thereby providing the desired GCM view on UXF. Hence, in this architecture the mediator needs only *a single GCM engine* for handling *arbitrary* CMs.

3 Formal Framework

Our requirements for the generic conceptual (meta-)model GCM are derived from the following typical features of conceptual models: Elements of the domain, called *objects* (or *entities*) are organized into *classes* (*entity types*), based on similarity or common features. The available *methods* (*attributes*, *slots*) and their result types determine the structure of the objects of a class. Classes can be organized hierarchically via a *subclass relation*. The latter induces a notion of *inheritance*, for example, *structural* and *value inheritance* (instances of a class inherit their "slot structure" and possibly some default values from the direct superclass[5]), but also *behavior* (via derived or computed methods) may be inherited as in OOP. Objects can participate in *relationships* (or *associations*) which can be further constrained to be *aggregations*, *compositions*, or other whole/part relationships with a specific semantics [Ode94]. Additional semantics of relations can also be expressed using *cardinality constraints*, *value constraints*, *functional dependencies*, etc. Finally, arbitrary non-structural *domain-specific constraints* are often expressed in a more or less formal constraint language like OCL (UML's object constraint language). From the above, we derive the following minimal requirements for the GCM.

GCM Core Expressions. The GCM should allow the following atomic schema- and instance-level declarations:

- $\mathsf{instance}(X, C)$ specifying that the object named X is an *instance of* class C (INST)
- $\mathsf{subclass}(C_1, C_2)$ specifying that C_1 *is a (subclass of)* C_2, i.e., instances of C_1 are also instances of C_2 [6] (SUB)
- $\mathsf{method}(C, M, C_M)$ specifying that *method* M is applicable to objects in C yielding (zero or more) objects in C_M. A concrete instance (method result) can be denoted similarly, say for $\mathsf{instance}(x, C), \mathsf{instance}(y, C_M)$ as $\mathsf{method}_{inst}(x, M, y)$ (METH)
- $\mathsf{relation}(R, A_1/C_1, \ldots, A_n/C_n)$ specifying an n-ary *relationship* between objects of classes $C_1, \ldots, C_n$. The corresponding *association roles* $A_1, \ldots, A_n$ facilitate a tuple-calculus notation: for $i = 1, \ldots, n$ the term $R.A_i$ denotes the X_i for which there is an instance: $\mathsf{relation}_{inst}(R, A_1/X_1, \ldots, A_n/X_n)$ in R. (REL)

[5] A *multiple inheriance problem* can arise if a class has several direct superclasses.

[6] sometimes C_1, C_2 are called *specializations* or *generalizations* of each other

Note that (SUB) means that subclass is reflexive and transitive. Often antisymmetry is also assumed, implying that subclass is a partial order, which prevents cycles in the class hierarchy.

GCM Extension Mechanism: Logic Rules. The actual syntactic and semantic extension mechanism of GCM is given by a suitable language for *logic rules*, i.e.,

- a *syntax* for rules in the style "*head* **if** *body*" deriving new information (=*head*) provided *body* is true, and (RULES)
- an associated logical *semantics*. (SEM)

The form of the rule head determines what is being defined, e.g., new instance or schema information for objects, classes, and relationships using the core expressions above. Certain logic rules, called *integrity constraints* do not derive "regular object information" but check the consistency of a CM instance. We express a logic integrity constraint φ that should hold for all instances of a CM as a *denial* $\psi := \neg\varphi$. Hence if ψ can be derived then the CM instance violates φ and an inconsistency is detected. We extend this basic functionality of boolean denials ψ by requiring, that when a violation is detected

- a denial ψ can add a *failure witness* w_ψ to a distinguished *inconsistency class* **ic**. (IC)

Example 2 shows how such witness objects are inserted into **ic**.

GCM Expressiveness. There are several candidate formalisms that satisfy the above GCM requirements. For example one may just use first-order calculus FO or some Datalog variant for specifying extensions to the GCM core part. Indeed FO can already express *all* common constraints for relational models including key constraints, inclusion dependencies, aggregation and cardinality constraints etc. However, CMs often contain "inductive" properties and constraints like an *inheritance semantics* that relates the meaning of "subclass" to that of "instance", or a *closure property* for certain whole/part relationships. Such properties are *not* expressible by FO formulas but in appropriate extensions of FO with fixpoint operators like FO(LFP) or Datalog. Therefore our final requirement is that

- GCM expressiveness should extend FO and include *inductive properties*. (EXPR)

For example, if we pick FO(LFP) as the GCM rule language, we know that *all* PTIME properties can be computed on finite ordered instances of a CM. However, FO(LFP) and similar fixpoint logics from finite model theory do not have an intuitive, declarative semantics, hence are not adequate as a specification language for CMs. A declarative rule language with an intuitive semantics that expresses precisely FO(LFP) is Datalog with well-founded negation.

F-Logic as GCM

As the concrete GCM for the formalization and implementation of our model-based mediator system, we use F-Logic (short: FL) [KLW95], an object-oriented extension of well-founded Datalog. The choice of FL is partly for convenience, since FL natively contains all of the above-mentioned GCM concepts (and several others) due to its roots in knowledge representation and deductive, object-oriented databases. Hence with FL we get a GCM formalism "for free"

GCM expression	FL expression	FL axioms
$\mathsf{instance}(X, C)$	$X : C$	...
$\mathsf{subclass}(C_1, C_2)$	$C_1 :: C_2$	$C :: C$:– $C :$ **class**
$\mathsf{method}(C, M, C_M)$	$C[M \Rightarrow\!\!> C_M]$	$C1 :: C2$:– $C1 :: C3, C3 :: C2$
$\mathsf{method}_{inst}(X, M, Y)$	$X[M \rightarrow\!\!> Y]$	$X : C2$:– $X : C1, C1 :: C2$
$\mathsf{relation}(R, A_1/C_1, \ldots, A_n/C_n)$	$R[A_1 \Rightarrow C_1; \ldots; A_n \Rightarrow C_n]$	...
$\mathsf{relation}_{inst}(R, A_1/X_1, \ldots, A_n/X_n)$	$r(X_1, \ldots, X_n) : R[A_1 \rightarrow X_1; \ldots; A_n \rightarrow X_n]$	...

Table 1: F-logic fragment for the generic conceptual model GCM

and avoid indirect Datalog encodings at the user level. In particular, the flexible, higher-order FL syntax turns out to be extremly useful in the real system.[7] Last but not least, FL implementations like FLORA [YK00] and FLORID [FLO] are readily available and have been successfully used in related areas like querying of semistructured data [LHL+98], mediation of Web sources [MHLL99], and in an earlier version of our Neuroscience mediator [GLM00].

As the GCM, we use a fragment of FL that can "host" all standard CM formalisms as "logic plug-ins". Table 1 shows the equivalent FL syntax for the GCM core expressions and a minimial set of FL axioms specifying the reflexive and transitive closure of "**::**", and the upward propagation of "**:**" wrt. "**::**". As the GCM extension mechanism we use FL rules with well-founded negation semantics, i.e., expressions of the form *head* :– *body* where *head* is an FL atom that becomes true if *body*, a conjunction of FL atoms or negated atoms, is true. Recall that the GCM and its FL incarnation only specify a minimal core model, but additional constraints can be easily added using logic rules: For example, assume the CM requires that the subclass relation "**::**" or any other relation R is a partial order. The following example illustrates how this is formalized.

Example 2 (IC Checking of Inductive Properties) The following integrity constraints test whether a binary relation R is a partial order on a class C: rule (1) finds all X in C for which R is not reflexive. Similarly, (2) reports missing transitive edges, and (3) derives node object pairs that violate R's antisymmetry on C. Thus, R is a partial order on C iff (1–3) do *not* insert a failure witness into **ic**:

(1) w_{rc}(C,R,X) **:ic** :– X **:** C, **not** R(X,X) .
(2) w_{tc}(C,R,X,Z,Y) **:ic** :– X,Y,Z **:** C, R(X,Z), R(Z,Y), **not** R(X,Y) .
(3) w_{as}(C,R,X,Y) **:ic** :– X **:** C, R(X,Y), R(Y,X), X≠Y .

If we assign "**::**" and the meta-class "**class**" (holding all class names) to the relation variable R and class variable C respectively, the above rules test whether "**::**" is indeed a partial order. This example also shows the power of schema reasoning in FL. □

Example 3 (Cardinality Constraints) In real-life applications, aggregation and cardinality constraints are ubiquitous. Consider the GCM declaration $\mathsf{relation}(R, A/C_1, B/C_2)$ and assume the CM at hand specifies that the cardinalities of roles A and B satisfy the conditions $card_A(N) := (N = 1)$ and $card_B(N) := (N \leq 2)$. Applied to has(neuron,axon) this says that a neuron can have ≤ 2 axons and an axon is contained in exactly one neuron. This can be expressed as follows:

[7] for schema-level deductions, XPath queries, rule maintenance (e.g., when object parameters are dropped or added, the "variable-arity" frame notation is more robust wrt. the changes), etc.

$w_{\neq 1}$(R,VB,N) **:ic** :– N = **count**{VA[VB]; R(VA,VB)}, **not** N=1.
$w_{>2}$(R,VA,N) **:ic** :– N = **count**{VB[VA]; _**:** R[A→VA; B→VB]}, **not** N ≤ 2.

The body of the first rule counts for each value VB of B the number N of values VA. If $N \neq 1$ a cardinality violation is detected and the witness $w_{\neq 1}$ gives the violating triple R, VB, N. The second rule illustrates a different FL syntax for tuple objects and checks $N \leq 2$ for B by grouping on VA. □

4 Model-Based Mediation with Domain Maps

Domain maps (DMs) formalize expert knowledge that is needed to mediate across multiple world scenarios. In our architecture, DMs are special conceptual models whose classes we call *concepts*. Concepts provide the semantic anchor points from which sources can "hang off" their data. Concepts can be linked via binary relations called *roles*. Intuitively, a labeled edge $C \xrightarrow{r} D$ of a DM means that if $c \in C$ then there is some $d \in D$ such that $r(c,d)$ holds. For example, $\mathsf{spiny\_neuron} \xrightarrow{\mathsf{has}} \mathsf{spine}$ means that every spiny neuron must have some spine. This can be formalized in description logic by the statement $\mathsf{spiny\_neuron} \sqsubset \exists \mathsf{has.spine}$ [CLN98] (see Figure 1 for a DM map and its description logic equivalent).

In principle, a DM can make use of the full expressive power underlying the GCM, i.e., concepts and roles could be defined via rules and not just as atomic facts. This results in a very expressive formalism and is sometimes useful in the definition or execution of mediated views (cf. Section 5). On the other hand, expressiveness is paid for with complexity and it is easy to see that the requirement (EXPR) above can make certain *reasoning* tasks about concepts undecidable:

Proposition 1
Subsumption and satisfiability are undecidable for unrestricted GCM domain maps. □

Here, *subsumption* means to decide whether membership in a concept class C implies membership in another class D, for *all* logic interpretations (i.e., instances of the DM) $\mathcal{I}$ that satisfy a given domain map DM. *Satisfiability* is the question whether such an $\mathcal{I}$ exists.

However, note that in a mediator system some specific DM is given which is used for navigating the multiple worlds domain and for defining and *executing* integrated views. Thus *reasoning about* the DM may be required only to a limited extent. Also real-life DMs often do not require the full expressive power of GCM in which case subsumption and satisfiability can be decided. Indeed, consider the ANATOM domain map in Figure 4 which is used in our mediator prototype. Using standard deduction techniques one can show:

Proposition 2 ANATOM *is satisfiable.* □

The formal semantics of DMs like ANATOM is defined as follows.

Definition 1 (Semantics of Domain Maps) Let $\mathcal{C}$ and $\mathcal{R}$ be finite sets of *concepts* and *roles*, respectively. A *domain map* $\mathcal{DM} \subseteq \mathcal{C} \times \mathcal{R} \times \mathcal{C}$ is an edge-labeled digraph over $\mathcal{C}$ and $\mathcal{R}$. The *semantics* of an edge $e = (C \xrightarrow{r} D) \in \mathcal{DM}$ is defined by

- $\forall x\ (C(x) \rightarrow \exists y\ (D(y) \wedge r(x,y)\ )$ SEM(e)

The semantics of $\mathcal{DM}$ is the conjunction of all SEM(e), for $e \in \mathcal{DM}$. □

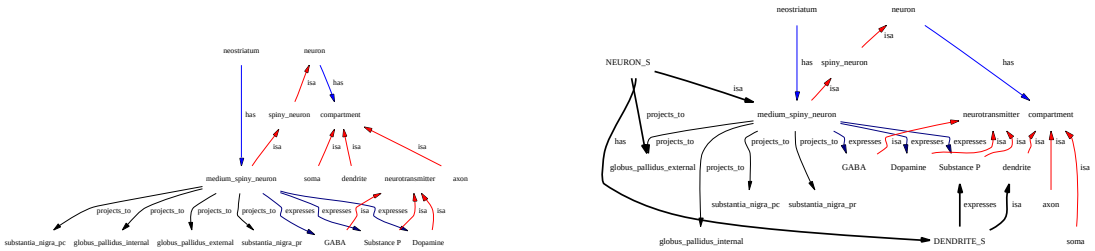


Figure 3: Domain map before (left) and after (right) registering source data

The equivalent of the FO sentence SEM(e) in FL and DL (description logic) are:

- $\exists Y\ (\ Y{:}D, r(X,Y)\ )\ \text{:-}\ X{:}C\ .$ FL(e)
- $C \sqsubset \exists r.D$ DL(e)

There are two quite different ways in which we could "execute" the axiom FL(e) at the mediator, i.e., as an integrity constraint or as an assertion. The translation of FL(e) as an *integrity constraint* is

$\mathsf{w}_{C,r,D}(\mathsf{X}):\mathsf{ic}\ \text{:-}\ \mathsf{X}:\mathsf{C},\ \textbf{not}\ (\ \mathsf{Y}:\mathsf{D},\ \mathsf{r}(\mathsf{X},\mathsf{Y})\ )$

and tests whether the mediator's object base contains for each $\mathsf{X}:\mathsf{C}$ a corresponding $\mathsf{Y}:\mathsf{D}$; otherwise a violation is reported. Such an integrity constraint is useful when the mediated object base is required to be *complete* wrt. $C \xrightarrow{r} D$.

The other, more frequent case is to view FL(e) as an *assertion* that in the real world (but not necessarily in the object base) the corresponding target object y exists. The following assertion creates a virtual placeholder object $\mathsf{f}_{C,r,D}(x)$ if the object base does not contain y:

$\mathsf{Y}:\mathsf{D},\ \mathsf{r}(\mathsf{X},\mathsf{Y})\ \text{:-}\ \mathsf{X}:\mathsf{C},\ \textbf{not}\ (\ \_\mathsf{Z}:\mathsf{D},\ \mathsf{r}(\mathsf{X},\_\mathsf{Z})\ ),\ \mathsf{Y} = \mathsf{f}_{C,r,D}(\mathsf{X})\ .$

Registering Source Data

When a source S registers with the mediator M, it can "anchor" the objects of CM(S) by sending the equivalent of the following FL expressions to M:

$o[\mathsf{anchor}{\twoheadrightarrow}\{c_1,\ldots,c_n\};\ \mathsf{context}{\twoheadrightarrow}\{r_1(d_1,e_1),\ldots,r_k(d_k,e_k)\}]\ .$

This expression declares that the object $o \in$ CM(S) should be anchored simultaneously at the concepts $c_1,\ldots,c_n$ of the mediator's DM. The object o's *context map* $o.\mathsf{context}$ is a connected subgraph of $\mathcal{C} \times \mathcal{R} \times \mathcal{C}$ and can be used to register additional knowledge about o with the mediator. This asserts the following new information at the mediator's domain map:

$o:\mathsf{subc}(c_1,\ldots,c_n):\mathsf{concept}(S).\qquad \mathsf{subc}(c_1,\ldots,c_n)::c_1.\quad\ldots\quad \mathsf{subc}(c_1,\ldots,c_n)::c_n.$

$r_1(d_1,e_1):\mathsf{role}(S).\quad\ldots\quad r_k(d_k,e_k):\mathsf{role}(S).$

This creates a new concept class subc and makes it a subclass of all concepts $c_1,\ldots,c_n$. The source object o becomes and instance of the new concept. Additionally, o's context map $o.\mathsf{context}$ is added to the mediators DM. Observe that by parameterizing with S, the mediator can keep track of which source has inserted a particular new concept or context into DM.

Example 4 (Anchoring Source Data) Figure 3 shows a part of the mediators domain map, before and after registering two classes of source data, "NEURON(S)" and "DENDRITE(S)".

□

Integrated View Definition Using Domain Maps

In Example 1, the two sources could be related simply because their data was anchored at concepts that were linked via relationships in the domain map. This is an example of a loose *federation* of correlated data where no new *integrated objects* are computed at the mediator. Instead, the integrated view is just a union view on the sources but — due to the model-based architecture — with the advantage that data can be navigated an correlated at the conceptual level using a domain map. The following example goes beyond this loose federation and extends the global-as-view integration paradigm to define integrated views not only over classes from information sources, but over a combination of information sources and the domain map:

Example 5 (Integration with DMs) The following is part of our CM declarations for the SENSELAB source that provides neurotransmission data:[8]

```
neuron[name⇒string; compartments⇒⇒compartment; organism⇒string].
neurotransmitter[name⇒string; chemical_type⇒string; role⇒string].
neurotransmission[transmitting_neuron⇒neuron; receiving_neuron⇒neuron;
        neurotransmitter⇒neurotransmitter].
```

The NCMIR source has data about protein distributions:

```
protein[name⇒string; isoform⇒string; antibody⇒string; raised_in⇒animal].
animal[genus⇒string; species⇒string].
protein_amount[protein⇒protein; neuron⇒string; regions⇒⇒region; amount⇒float].
region[region_number⇒integer; compartment⇒string; fraction⇒float].
```

The intuitive idea is that a confocal microscope image is broken down into regions. Each region may have a number of compartments. Proteins are measured per region instead of per compartment (since it is very hard to isolate each compartment when the dendrites are very narrow). The protein amount is thus assessed by the estimated fraction occupied by each compartment. □

The integrated view definition IVD uses the ANATOM domain map whose base relations isa and has_a are depicted on the left of Figure 4. In order for the subsequent computation on the IVD to be correct, we have to *infer* from the base relations, for example, that "Purkinje cells have axons", since axons are compartments of neurons, and Purkinje cells are neurons. This is accomplished by applying certain *graph operations* on the domain map DM. In this case, we need to derive the *deductive closure* "has_a_star" of "has_a" wrt. "isa":

```
tc(R)(X,Y) :- R(X,Y).                    dc(R)(X,Y) :- tc(isa)(X,Z), R(Z,Y).
tc(R)(X,Y) :- tc(R)(X,Z), tc(R)(Z,Y).    dc(R)(X,Y) :- R(X,Z), tc(isa)(Z,Y).

has_a_star(X,Y) :- dc(has_a)(X,Y).
```

The first pair of rules defines the transitive closure $\mathsf{tc}(R)$ for any binary relation R. The second pair defines the deductive closure $\mathsf{dc}(R)$ of a relation R wrt. the transitive closure of isa. Intuitively, R links are propagated up and down the isa chains. The last rule derives a new relation "has_a_star" that contains all inferable *direct* "has_a" links. Note that "has_a_star" itself is *not* transitive (Figure 4). Indeed, it would be wasteful to compute the much larger tc(has_a_star) when evaluating the IVD since a recursive traversal of the direct links is sufficient.

[8] senselab.med.yale.edu

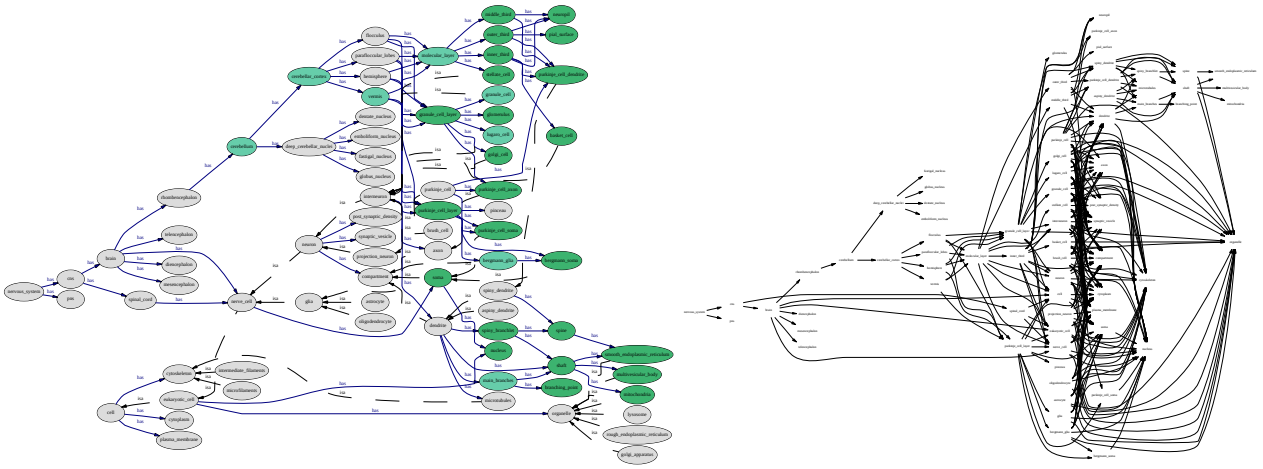


Figure 4: ANATOM domain map; *left*: isa ∪ has_a (shades indicate absence or (in)direct presence of data; *right*: has_a_star, the deductive-closure of isa and has_a)

Example 5 (Cont'd) Using all of the above we can construct the IVD for the mediated class protein_distribution and populate it in the following manner:

```
protein_distribution[protein_name⇒string; ion_bound⇒string; distribution_root⇒string;
        distribution⇒distribution].
distribution[anatomic_structure⇒string; protein_amount⇒float; distribution⇒distribution].

D:protein_distribution[protein_name→Y; animal→Z; distribution_root→P; distribution→D] :–
    'NCMIR'.protein.name=Y,
    'SENSELAB'.neuron.organism=Z,
    contains('ANATOM'.nervous_system.has_a_star, P),
    compute_aggregate(Y,'NCMIR'.protein_amount.amount,has_a_star,P,D).
```

The function compute_aggregate recursively traverses a binary relation R (here: has_a_star) starting from node P, and computes the aggregate of the specified attribute at each level of the relation R. The result for the computation for P="cerebellum", Z="rat", and Y="Ryanodine Receptor" can be seen in the system snapshot in Figure 5.

5 Query Processing in the KIND Mediator Prototype

A prototypical implementation of the model-based mediator architecture called KIND[9] [LGM00] has been developed, based on the FLORA system [YK00] as the deductive engine. The development of the architecture and the system was driven by the need to mediate real data coming from largely disjoint Neuroscience "worlds". The following example is taken from this mediation scenario and illustrates how generic operations on the domain map are useful to formulate and execute complex queries at the mediator.

Consider the classes protein_distribution and distribution from above and the class neurotransmission:

```
neurotransmission[organism⇒string; transmitting_neuron⇒string; transmitting_compartment⇒string;
        receiving_neuron⇒string; receiving_compartment⇒string; neurotransmitter⇒string].
```

[9] ***K**nowledge-based **I**ntegration of **N**euroscience Data*, `www.npaci.edu/DICE/Neuro/kind01.html`

Based on these mediated classes, we can now answer a query like:

> *"What is the distribution of those calcium-binding proteins that are found in neurons that receive signals from parallel fibers in rat brains?"*

In terms of the given views, this user query can be written as

```
answer(P, D) :–
        neurotransmission[organism→'rat'; transmitting_compartment→'parallel fiber';
                receiving_neuron→X; receiving_compartment→Y],
        D : protein_distribution[protein_name→P; ion_bound→↠{calcium}; distribution_root→_].
```

This is a typical query of a scientist who studies *neurotransmission* (and produces the data of SENSELAB above), and needs information gathered by groups that study *protein localization* (like NCMIR). Note that the user does not specify the distribution root, forcing the mediator to provide a "reasonable" root for the neuron-compartment pairs that satisfy the first condition. The following are the main steps of the query plan:

1. *push selections* ('rat', 'parallel fiber') to SENSELAB and *get bindings* for X and Y
2. using the domain map DM(M), *select sources* that have data anchored for the neuron/compartment pairs X,Y from step (1); in our case, only NCMIR is returned
3. *push selections* given by the X,Y locations to NCMIR, and *retrieve* only proteins P that are found in X,Y

Now we need to compute the actual distribution of each protein P from NCMIR at the mediator. But to do this using the view defined earlier, we first must determine which brain_region of the neuron should serve as the root of the distribution. This is accomplished by computing the *least upper bound* (*lub*) of locations in the domain map.

4. with the *lub* as the root node, *compute the view* protein_distribution at the mediator as described before. Note that this involves a *downward closure* along the has_a_star relation.

The last two operations filter out a segment in the domain map as the "region of correspondence" between the two information sources, and demonstrate how graph operations on the domain map can be actively used to compute conceptual mappings between sources.

6 Discussion and Conclusions

We have presented a novel mediator architecture for complex multiple world scenarios, which require additional knowledge in order to federate or integrate across the data. The additional domain knowledge is made available to the mediator in the form of a high-level *domain map* acting as a "semantic coordinate system" that can be used by sources to situate their data in the global context. The complexity of scientific domains like the Neurosciences also requires that view definitions are expressed at the semantically rich *conceptual level* and not just at the level of semistructured data (XML) as in current mediator systems. Our architecture is "immune" to the formalism for conceptual models as used by the sources due to a plug-in mechanism that maps other CMs, expressed in XML syntax, via complex XML queries, to a generic conceptual model GCM. A prototype has been implemented using an underlying

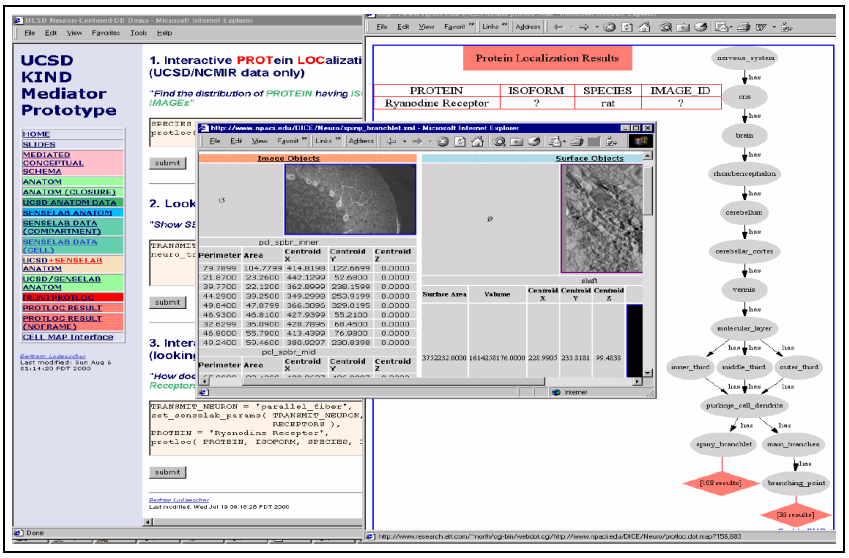


Figure 5: Snapshot of the KIND mediator prototype; *left*: meditor shell for issuing ad-hoc queries against CM(M); *right*: generated subgraph of ANATOM having the requested result data; clicking on a (diamond) result node retrieves the actual result data (*center*)

F-logic engine for evaluating queries and views in the GCM, graph operations on the domain map (e.g., *lub* and deductive closures), and even I/O operations like XML transformations (as needed for CM plug-ins) and generation of DM graphs for the user interface [LGM00].

Related Work. [CDGL$^+$98] present an architecture that uses conceptual models to support information integration. While we use an FL version of GCM, they employ a description logic called $\mathcal{DLR}$ to formalize ER diagrams and other CMs. Note that the focus in description logics is on *reasoning* about CMs at the *schema level* and not on deriving new information about a populated *instance* of a CM as in our case. Therefore description logics are designed such that problems like *satisfiability*, *subsumption*, and *equivalence* of concepts remain decidable at the schema level. Since already FL without object creation (i.e., function symbols) can express all FO queries, reasoning about CMs in our GCM model is undecidable in general. However, in our architecture we use only a limited amount of reasoning about CMs and the focus is on *execution* (evaluation) of logic rules on given object instances of CMs, i.e., a much more tractable problem. Moreover, in real application scenarios like our Neuroscience domain, restricted and decidable fragments like the ANATOM domain map are often sufficient.

[FRV96] present a method for rewriting and decomposing queries in a cooperative information system using "semantic knowledge". However, their work does not deal at all with mediation at the conceptual level, or the use of domain knowledge to mediate across multiple world scenarios. Rather "semantic knowledge" in their setting means OQL rewrite rules of the form $Q_1 \rightsquigarrow Q_2$ that can be applied for query reformulation.

A system architecture developed by experts from the Neuroscience domain is described in [NLC$^+$99]. Like many generic models, their EAV/CR model is based on a ternary entity-attribute-value representation, extended with classes and relationships. However their approach deals with the "data part" of integration only. In particular, there is no suitable declarative rule language for defining complex integrated views or queries.

The importance of semantics in information exchange is also witnessed by the recent interest in XML Schema and RDF. Indeed RDF, when used with a rule language like F-logic can be used as GCM (and so could XML Schema, but XML Schema is quite "heavy-weight" compared with the lean and elegant approach possible with RDF+rules).

At least two decades of prior research exists in the general area of information integration. Sheth, in a recent overview [She98], classified information integration research into three generations. In our architecture, similar to second generation approaches like TSIMMIS [GMPQ+97], integrated views are defined using the so-called *global-as-view* (GAV) approach, in which an integrated view definition IVD of the global view is defined in terms of local views on the sources. However, our system specifies (and executes!) IVDs at the level of *conceptual models* exported by the sources and thus falls into the category of third generation approaches which focus on semantic integration. Moreover, unlike other GAV systems our use of a domain map allows us to define global views on the sources in conjunction with the map. This *knowledge-based mediation* allows us to construct views over data that could not have been joined directly.

Since domain maps correspond to edge-labeled graphs, our global views involve *complex recursive operations*. On the other hand, *local-as-view* (LAV) approaches like SIMS [SIM98] define each local source as a view on the more pervasive global schema. For answering a user query on the global schema, an inverse operation is used to map the query to appropriate local schemata. Often, such inverse operations may not, and in the case of our complex, recursive views, do not exist.

COIN [GBMS99] performs integration by creating a domain model as a universe of primitive and semantic types, where a semantic type can take a different *value* in every context that uses it. The system allows the mapping of source-specific values to the same semantic type and permits axioms that convert between value domains for the same semantic type. Thus COIN's notion of domain knowledge or ontology effectively resolves *attribute domain conflicts*, and does not address the problem of mediating semantically distinct schema by any schema-based integration operation.

References

[CDGL+98] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information Integration: Conceptual Modeling and Reasoning Support. In *Intl. Conference on Cooperative Information Systems (CoopIS)*, pp. 280–291, 1998.

[CLN98] D. Calvanese, M. Lenzerini, and D. Nardi. Description Logics for Conceptual Data Modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pp. 229–263. Kluwer Academic Publisher, 1998.

[FLO] FLORID Homepage. `www.informatik.uni-freiburg.de/~dbis/florid/`.

[FRV96] D. Florescu, L. Rashid, and P. Valduriez. A Methodology for Query Reformulation in CIS Using Semantic Knowledge. *Intl. Journal of Cooperative Information Systems*, 5(4), 1996.

[GBMS99] C. H. Goh, S. Bressan, S. E. Madnick, and M. D. Siegel. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems (TOIS)*, 17(3):279–293, 1999.

[GLM00] A. Gupta, B. Ludäscher, and M. E. Martone. Knowledge-Based Integration of Neuroscience Data Sources. In *12th Intl. Conference on Scientific and Statistical Database Management (SSDBM)*, Berlin, Germany, July 2000. IEEE Computer Society.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, 8(2), 1997.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, July 1995.

[KS96] V. Kashyap and A. Sheth. Semantic and Schematic Similarities between Database Objects: A Context-based Approach. *VLDB Journal*, 5(4):276–304, 1996.

[LGM00] B. Ludäscher, A. Gupta, and M. E. Martone. A Mediator System for Model-Based Information Integration. In *26th Conf. on Very Large Data Bases (VLDB)*, Cairo, Egypt, 2000. Morgan Kaufmann. demonstration.

[LHL+98] B. Ludäscher, R. Himmeröder, G. Lausen, W. May, and C. Schlepphorst. Managing Semistructured Data with FLORID: A Deductive Object-Oriented Perspective. *Information Systems*, 23(8):589–613, 1998.

[LPV00] B. Ludäscher, Y. Papakonstantinou, and P. Velikhov. Navigation-Driven Evaluation of Virtual Mediated Views. In *Intl. Conference on Extending Database Technology (EDBT)*, LNCS 1777, Konstanz, Germany, 2000. Springer.

[MHLL99] W. May, R. Himmeröder, G. Lausen, and B. Ludäscher. A Unified Framework for Wrapping, Mediating and Restructuring Information from the Web. In *Intl. Workshop on the World-Wide Web and Conceptual Modeling (WWWCM'99)*, LNCS 1727, Paris, France, 1999. Springer.

[NLC+99] P. M. Nadkarni, L. M. L, R. Chen, E. Skoufos, G. Shepherd, and P. Miller. Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. *Journal of the American Medical Informatics Association*, 6(6):478–493, 1999.

[Ode94] J. J. Odell. Six Different Kinds of Composition. *Journal of Object-Oriented Programming*, 5(8), 1994.

[She98] A. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman, editors, *Interoperating Geographic Information Systems*. Kluwer, 1998.

[SIM98] SIMS Homepage. `www.isi.edu/sims/`, 1998.

[SY98] J. Suzuki and Y. Yamamoto. Making UML Models Interoperable with UXF. In *Intl. Workshop ≪UML'98≫: Beyond the Notation*, LNCS 1618, Mulhouse, France, 1998.

[XMI99] OMG XML Metadata Interchange (XMI). `www.omg.org/cgi-bin/doc?ad/99-10-02`, 1999.

[YK00] G. Yang and M. Kifer. FLORA: Implementing an Efficient DOOD System Using a Tabling Logic Engine. In *6th International Conference on Rules and Objects in Databases (DOOD)*, 2000.