An Extensible Model-Based Mediator System with Domain Maps

Amarnath Gupta* Bertram Ludäscher* Maryann E. Martone‡

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

1 Background

Mediator systems federate and integrate data from disparate sources in order to elicit information that the individual sources cannot provide independently. 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. Indeed, as shown in [4, 1, 3], mediation should be lifted to the conceptual level when mediating across complex sources whose data comes from seemingly disjoint "worlds", *e.g.*, two neuroscience labs creating information on neurotransmitters and protein distributions, respectively.[1]

To this end, we present a mediator prototype system whose main novel features are: (i) mediated views are defined *and executed* at the level of *conceptual models* (CMs) rather than at the usual structural level, (ii) *domain maps* (DMs) – labeled graphs of concepts and relationships with a formal logic semantics – are used to bridge the semantic gap between source data from "multiple worlds", and (iii) a *plug-in mechanism* for CMs and DMs is provided which allows the mediator system to be easily extended when new formalisms for CMs and DMs are used by sources. We illustrate these features using an example from a complex neuroscience mediation problem.

For details of model-based mediation with domain maps, including their formal semantics, see [4].

[1] See senselab.med.yale.edu and www-ncmir.ucsd.edu.

2 Model-Based Mediation

Figure 1 depicts our system architecture for model-based mediation: Differences in the sources' data models are resolved by wrappers that translate the raw data into a common generic data format (XML). Current mediator systems directly define the integrated views on the wrapped XML sources using an XML query language. We extend this architecture by lifting exported source data to the semantically richer level of conceptual models with domain knowledge. Thus, the integrated view definition IVD at the mediator is aware of class hierarchies, object structure, properties of relationships (inclusion dependencies, cardinalities, ...), and in particular *domain specific constraints* of sources. Consequently, the mediator's view definition language in this architecture must not only act as query language for semistructured data, but also for conceptual models including the definition of complex schema and instance level transformations and checking of logical constraints.

2.1 Generic Conceptual Model (GCM)

To facilitate extensibility, we use GCM, a generic conceptual model, at the mediator level. Like RDF, GCM is a minimalist object-oriented model that allows specification of objects at the schema and instance level (*e.g.*, method($class_A$,$meth$,$class_B$) and method$_{inst}$(oid_A,$meth$,oid_B)), similar for relations, and – most importantly – a *rule-based extension mechanism* for axiomatizing additional CM constructs and constraints. The formal model of GCM is a fragment of F-logic (short: FL) [2] with well-founded negation semantics. In this way, GCM is *universal* for CMs since all first-order constraints (cardinality constraints, range constraints, inclusion dependencies, *etc.*) and the usual inductive constraints (*e.g.*, transitivity of the class hierarchy) are expressible in the GCM formalism. The choice of a FL for our GCM is partly for convenience, since FL already includes all required GCM features and we thus get a GCM for-

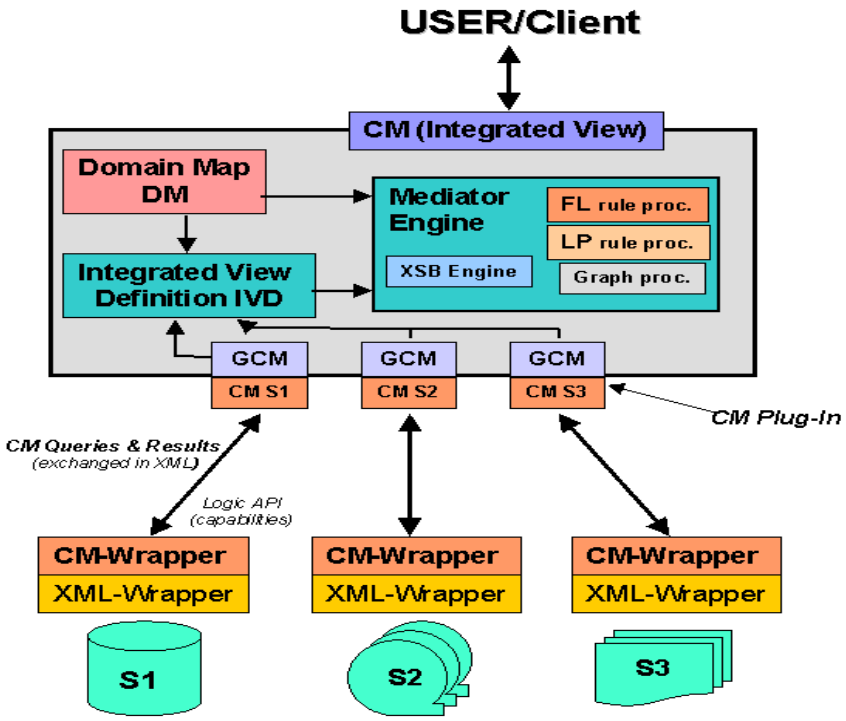

Figure 1. Architecture of a model-based mediator with CM plug-ins

malism "for free", without having to encode it indirectly say as in Datalog. In particular, the flexible, higher-order FL syntax turned out to be extremly useful in the real system. Finally, FL implementations like FLORA [6] are readily available and have been successfully used in an earlier version of our neuroscience mediator [1].[2]

2.2 CM Plug-In Mechanism

A goal of our extensible architecture is to make the mediator independent of a source's choice of CM formalism (like (E)ER or UML diagrams) for communicating CM schema and data. As a first step, all information (queries, CM signatures and data, mediator/wrapper dialogues, *etc.*) goes "over the wire" as XML. The second crucial step of the plug-in mechanism relies on the fact that GCM is universal and expressive enough to formalize any other CM formalism: The crux is that a new CM formalism say UXF [5] is added to the system simply by 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.

2.3 Domain Maps

(DMs) are used as "semantic road maps" in the mediation process and formalize expert knowledge that is needed to mediate across multiple world scenarios. In our system, DMs are special conceptual models whose classes are called *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. The formal semantics of DMs is given by a certain description logic,[3] *i.e.*, a decidable fragment of first-order logic (and thus of FL). At the top of Figure 2, the DM used for mediation of our neuroscience domain is shown: nodes cor-

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

[3] *e.g.*, $semantics(C \xrightarrow{r} D) := \forall x\,(C(x) \rightarrow \exists y\,(D(y) \wedge r(x, y)))$

respond to anatomical entities of the brain, edges correspond to relationships like is-a and has-a. Moreover, different shades indicate absence or (in)direct presence of data. Like CMs, DMs may have additional rules, in this case, that has-a needs to be closed wrt. the transitive is-a relation; the result is shown at the bottom of Figure 2.

2.4 Query Processing

At runtime, a wrapped source S joins the mediation 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 (ultimately expressed in FL) that are evaluable at the mediator (either using GCM, or any CM formalism for which a plug-in is available). The exported objects of a CM(S) can have special context attributes that provide the "semantic coordinates" of the data in the mediator's *domain map* DM(M). In particular, the context attributes can create new concept classes at M as a result of the source's registration process.

At the mediator, the user query is executed against the IVD. For example, a plan for the user query *"What is the distribution of those calcium-binding proteins that are found in neurons that receive signals from parallel fibers in rat brains?"* involves the following steps: (1) *push selections* ('rat', 'parallel fiber') to the SENSELAB source and *get bindings* for neuron/compartment pairs X and Y; (2) using the domain map DM(M), *select sources* that have data anchored at X,Y from step (1) – in our case, NCMIR; (3) *push selections* given by the X,Y locations to NCMIR, and *retrieve* only proteins P that are found in X,Y; (4) based on the *least upper bound* of locations in the domain map, *compute the view* protein_distribution at the mediator (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 (cf. Figure 3).

For the full

References

[1] A. Gupta, B. Ludäscher, and M. E. Martone. Knowledge-Based Integration of Neuroscience Data Sources. *12th Intl. Conf. on Scientific and Statistical Database Management (SSDBM)*, July 2000.

[2] 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.

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

[4] B. Ludäscher, A. Gupta, and M. E. Martone. Model-Based Mediation with Domain Maps. In *Intl. Conf. on Data Engineering (ICDE)*, `www.sdsc.edu/~ludaesch/Paper/icde01.html`, 2001.

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

[6] 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.

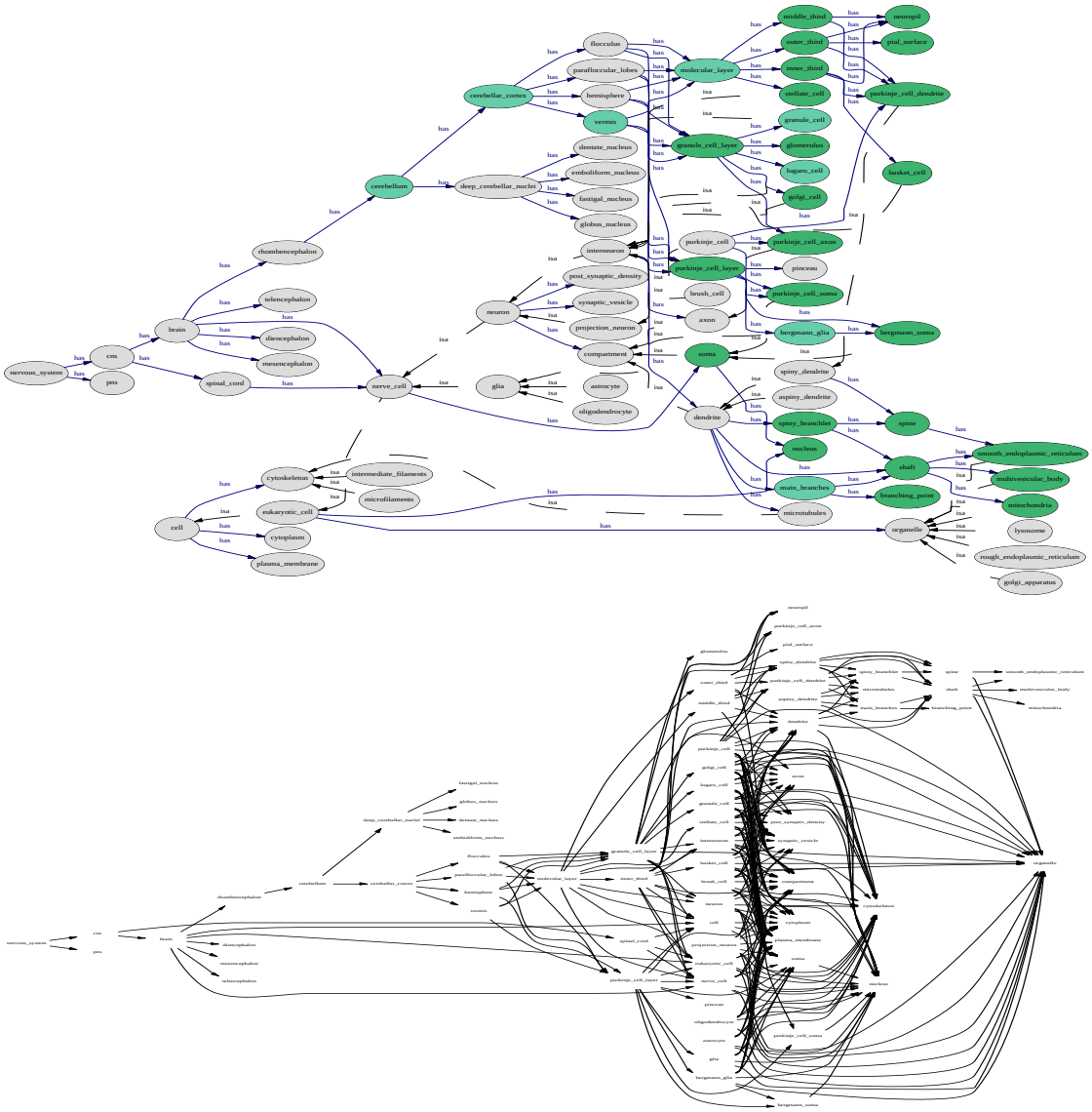

Figure 2. ANATOM **domain map:** is-a ∪ has-a **(***above***), and** has-a-star**, the deductive-closure of** is-a **and** has-a **(***below***)**

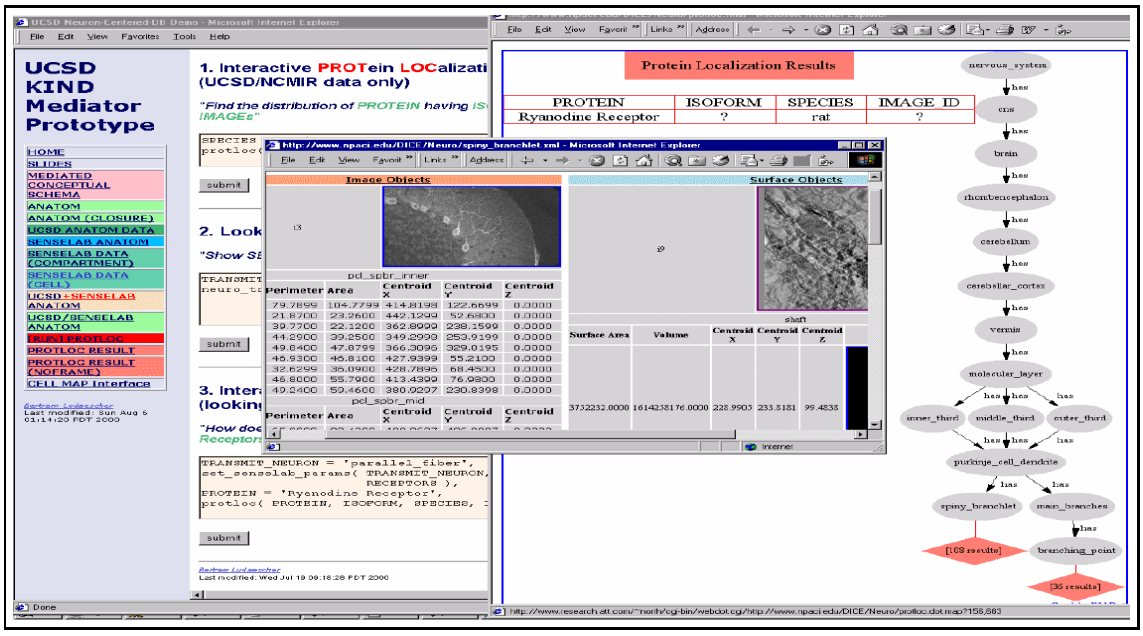

Figure 3. Snapshot of the mediator prototype; *background left*: **meditor shell for issuing ad-hoc queries against CM(**M**);** *background right*: **generated subgraph of** ANATOM **having the requested result data; clicking on a (diamond) result node retrieves the actual result data (***foreground center***)**