A Model-Based Mediator System for Scientific Data Management

Bertram Ludäscher[⋆] Amarnath Gupta[⋆] Maryann E. Martone[‡]

[⋆]San Diego Supercomputer Center, U.C. San Diego {ludaesch,gupta}@sdsc.edu

[‡]Department of Neurosciences, U.C. San Diego mmartone@ucsd.edu

Chapter 6

A Model-Based Mediator System for Scientific Data Management

A database mediator system combines information from multiple existing source databases and creates a new virtual, mediated database that comprises the integrated entities and their relationships. When mediating scientific data, the technically challenging problem of mediator query processing is further complicated by the complexity of the source data and the relationships between them. In particular, one is often confronted with "complex multiple-world scenarios", *i.e.*, in which both the semantics of individual sources as well as the knowledge to link them require a "deeper modeling" than offered by current database mediator systems. Based on experiences with federation of brain data, we present an extension called *model-based mediation* (MBM). In MBM data sources not only export raw data and schema information but *conceptual models* (CMs) including domain semantics to the mediator, effectively lifting data sources to "knowledge sources". This allows a mediation engineer to define integrated views based on (i) the local CMs of registered sources and (ii) auxiliary domain knowledge sources called *domain maps* (DMs) and *process maps* (PMs), respectively, which act as sources of "glue knowledge". For complex scientific data sources, semantically rich CMs are indeed *necessary* in order to represent and reason with scientific rationale for linking a wide variety of heterogeneous experimental assumptions, observations, and conclusions that together constitute an experimental study. We illustrate the challenges using real-world examples from a complex neuroscience integration problem and present the methodology and some of our tools, in particular the KIND[1] mediator prototype for model-based mediation of scientific data.

6.1 Background

Seamless data access and sharing, handling of large amounts of data, federation and integration of heterogeneous data, distributed query processing and application integration, data mining, and visualization are among the common and recurring broad themes of scientific data management in many disciplines. A main stream of activity in the bioinformatics domain is concerned with sequence and structural databases such as GenBank, NCBI, PDB, SwissProt, etc. and much work is devoted to algorithmic challenges stemming from problems, *e.g.*, efficient sequence alignment and structure prediction. However, in addition to the well-known challenges of main stream bioinformatics applications such as algorithmic complexity and scalability (*e.g.*, in genomics), there are other major challenges that are sometimes overlooked, particularly when considering other kinds of scientific data beyond the level of sequence and protein data, *e.g.*, brain imagery data. These challenges arise in the context of *information integration of scientific data* and have to do with the inherent semantic complexity of (i) the actual source data, and (ii) the "glue knowledge" that is necessary to link the source data in meaningful ways. We argue that traditional federated database system architectures and those of the more recent database mediators developed by the database community need to be extended in order to adequately handle information integration of complex scientific data from multiple sources. This extensions is a combination of knowledge representation and mediator technology – in a nutshell:

Model-Based Mediation = Database Mediation + Knowledge Representation

With respect to their *semantic heterogeneity* (ignoring syntactic and system aspect), we can roughly classify information integration/mediation scenarios (scientific or otherwise) along a spectrum as follows: On the one end, we have *simple one-world scenarios*, somewhere in the middle we have *simple multiple-world scenarios*, and at the other end of the spectrum, we find *complex multiple-world scenarios*. An example of a *simple one-world scenario* (*i.e.*, in which the modeled real-world entities can be related easily to one another and come from a single domain) is *comparison shopping*, say for books. A typical query is to find the cheapest price for a given book

[1] ***K***nowledge-based ***I***ntegration of ***N***euroscience ***D***ata

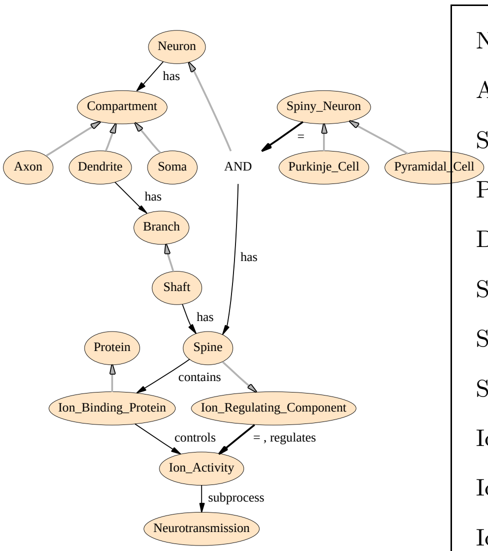
from a number of sources such as `amazon.com` and `bn.com`. An example of a *simple multiple-world scenario* is the integration of realtor and census data in order to annotate and rank real estate by neighborhood quality. Here, we combine and relate quite different kinds of information, but the relations between the multiple worlds are simple enough to be understood without deep domain knowledge. Examples of *complex multiple-world scenarios* are often found in scientific data management and are the subject of this chapter. Thus, "simple" and "complex" here refers to the degree in which specific *domain semantics* is required in order to be able to formalize or even state meaningful associations and linkages between data objects of interest – it does not mean that the database and mediation technology for realizing such mediators is simple.[2] For example, to state the problem what the result of an integrated comparison shopping view should be, a basic understanding of a "books schema" (title, authors, publisher, price, etc.) is sufficient. In particular, the association operation that links objects of interest across sources can be executed (at least in principle) as a *syntactic join* on the ISBN. Similarly, in the realtor example, data can be joined based on ZIP code, latitude/longitude, or street address, *i.e.*, by *spatial joins* which can be modeled as atomic function calls to a "spatial oracle". To understand the basic linkage of information objects, no "insight" into the details of the spatial join is required.

This is fundamentally different for complex multiple-world scenarios as found in many scientific domains. There, even if data is stored in state-of-the-art (often web-accessible) databases, significant domain knowledge is required in order to even articulate meaningful queries across disciplines (or within different micro-worlds of a single discipline); cf. the examples in the next section.

Outline. In this chapter, we illustrate these challenges using examples from our ongoing collaborations with users and providers of scientific data sets, in particular, from the Neuroscience domain (Section 6.2). We then present a methodology called *model-based mediation* which extends current database mediator technology by incorporating knowledge representation (KR) techniques in order to create explicit representations of domain experts' knowledge that can be used in various ways by mediation engineers and by the model-based mediator system itself (Section 6.3). The goal of model-based mediation could be paraphrased as

*Turning scientists' **questions** into executable database **queries**.*

[2] quite the opposite: such "simple" mediation scenarios often pose very difficult technical challenges, *e.g.*, query processing in the presence of limited source capabilities [PGH98, LYV+98].



Neuron $\sqsubseteq \exists$has.Compartment

Axon, Dendrite, Soma $\sqsubseteq$ Compartment

Spiny_Neuron $\equiv$ Neuron $\sqcap\ \exists$has.Spine

Purkinje_Cell, Pyramidal_Cell $\sqsubseteq$ Spiny_Neuron

Dendrite $\sqsubseteq \exists$has.Branch

Shaft $\sqsubseteq$ Branch $\sqcap\ \exists$has.Spine

Spine $\sqsubseteq \exists$contains.Ion_Binding_Protein

Spine $\sqsubseteq$ Ion_Regulating_Component

Ion_Activity $\sqsubseteq \exists$subprocess_of.Neurotransmission

Ion_Binding_Protein $\sqsubseteq$ Protein $\sqcap\ \exists$controls.Ion_Activity

Ion_Regulating_Component $\equiv \exists$regulates.Ion_Activity

Figure 6.1: A *domain map* for SYNAPSE and NCMIR (unlabeled, gray edges $\approx$ "isa" $\approx$ "$\sqsubseteq$") and its formalization in description logic

In Section 6.4 we introduce some of the KR formalisms, *e.g.*, for domain maps and process maps and describe their use in model-based mediation (some details are deferred to Appendix A). In Section 6.5 we present the KIND mediator prototype and other tools being developed at SDSC and UCSD, primarily in the context of the Neuroscience domain. We discuss some related work and conclude in Section 6.6.

6.2 Scientific Data Integration Across Multiple Worlds: Examples and Challenges from the Neurosciences

We illustrate some of the challenges of scientific data integration in complex multiple-world scenarios using examples that involve different "Neuroscience worlds". Such examples occur regularly when trying to federate brain data across multiple sites, scales, and even species [NPA01], and have led to new research and development projects aimed at overcoming the current limitations of biomedical data sharing and mediation [BIR01].

Example 1 (Two Neuroscience Worlds) Consider two Neuroscience laboratories SYNAPSE and NCMIR[3] 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

[3]see `synapses.bu.edu` and `www-ncmir.ucsd.edu`

thus the anatomical entities that are reconstructed from 3-dimensional serial-sections. For each entity (*e.g.*, spines, 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 a different cell type, 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 (*e.g.*, segment diameter) 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: Like pyramidal neurons, Purkinje cells also possess dendritic spines. Release of calcium in 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 neither of the sources contains information that would allow a mediator system to bridge the "semantic gap" between them. Therefore we need, independent of the observed experimental raw data of each source, *additional domain knowledge* in order to connect the two sources. The domain expert, here a neuroscientist has no problem providing us with the necessary "glue knowledge":

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

In order to capture such domain knowledge and make it available to the system, we employ two kinds of *ontologies*, called *domain maps* and *process maps*, respectively. While the former are aimed at capturing the basic domain terminology, the latter are used to model different process contexts (see below). Ontologies, such as the domain map in Figure 6.1 are often formalized in logic (here statements in *description logic* [CGL+98]; see Section 6.4.1). Together with additional inference rules (*e.g.* capturing transitivity of "`has`"), logic axioms like these formally capture the domain knowledge and allow a mediator systems to work with this knowledge (*e.g.*, a concept or class hierarchy can be used to determine whether the system should retrieve objects of class C' when the user is looking for instances of C).

Domain maps not only provide a concept-oriented browsing and data exploration tool for the end user, but – even more importantly – can be used for defining and executing integrated view definitions (IVDs) at the mediator. The above real-world example 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 in the scientific context*, which in this case is the neuroanatomy and neurophysiological setting described above. Therefore we call this *mediation across multiple worlds* and facilitate it using domain maps as the one shown.

From Terminology and Static Knowledge to "Process Context"

While domain maps are useful to put data into a terminological and thus somewhat "static" knowledge context, a different knowledge representation has to be devised when trying to put data into a dynamic or "process context". Consider, for example, groups of neuroscientists who study the science of mammalian memory and learning. Many of these groups study a phenomena called *long-term potentiation* (LTP) in nerve cells, where repeated or sustained input to nerves in specific brain regions (such as the hippocampus) conditions them in such a manner that after some time, the neuron produces a large output even with a small amount of "known" input. Given this general commonality of purpose, however, individual scientists study and collect observational data for very different aspects of the phenomena.

Example 2 (Capturing Process Knowledge) Consider a group [BST+00] that studies the role of a specific protein *N-Cadherin* in the context of *synpase formation* during *late-phase long-term potentiation* (L-LTP), a subprocess of LTP. The data collected by the group consists of measurements that illustrate how the amount of *N-Cadherin* and the number of synapses (nerve

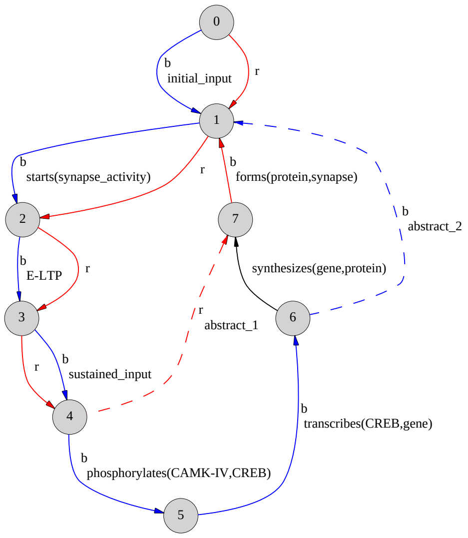


Figure 6.2: A simple *process map*. Blue and red edges (marked "b" and "r", respectively) depict processes about which two data sources/research groups have observational data; dashed edges indicate abstractions ("short cuts").

junctions) both simultaneously increase in cells during L-LTP. A different group, [KFM01], studies a new enzyme called CAMK-IV and its impact on a chemical reaction called phosphorylation of a protein called CREB. Their data is collected to show how modulating the amounts of CAMK-IV, and other related enzymes affect the amount of CREB production, and how this in turn, affects other products in the nucleus of the neurons. Ideally, the goal of "mediating" between experimental information from these two sources would be to produce an integrated view that enables an end-user scientist to get a deeper understanding of the LTP phenomena. Specifically, the end-user should be able to ask queries (and get answers) that exploit the scientific interrelationship between these experiments. In this way, the integrated access provided by a mediator system can lead to new observations and questions thus eventually driving new experiments.

At the risk of oversimplification, we can state that the first source looks at synapse-formation, and is only interested in the fact that some proteins (including *N-Cadherin*) bring about the formation of synapses. They do not look at the processes leading to the production of these proteins. The second source looks at a specific chain of events leading up to the production of the proteins, but does not identify which proteins are produced. The *semantic connection* between these two sources can be constructed in terms of the underlying "event structure" and how they elaborate on different parts of it. Figure 6.2 depicts a simplified view of the relationship explained above and shows the cyclic progression of events leading to synapse formation during LTP: Red edges situate the first source with respect to the overall process, while blue edges situate the second source. In either case, the dashed lines show the subsequence of events the sources "glossed over" or abstracted. Thus the first source does not have any information pertaining to *phosphorylates(CAMK-IV, CREB)*, and the second source does not have any data related to *forms(protein, synapse)*. Neither source has any data about the (black) edge *synthesizes(gene, protein)*. □

Domain maps allow data providers to put their source data into a static/terminological context, while process maps allow them to do the same for a dynamic/process context. Together they capture valuable "glue knowledge" that resides at the mediator and facilitates integration of hard-to-correlate sources; in particular, concept-oriented data discovery ("semantic browsing") [GLM01], view definition, and semantic query optimization [CGM90]. To make model-based mediation effective, it is also necessary to "hook" the elements of the source schema to the domain map and the process map. This process, which we call the *contextualization* mechanism, is central to the MBM framework.

6.3 Model-Based Mediation

In mediator systems, differences in syntax and data models of sources $S_1, S_2, \ldots$ are resolved by wrappers that translate the raw data into a common data format, typically XML. In most current mediator systems, all other differences, in particular schema heterogeneities are then handled by an appropriate integrated view definition (IVD) which is defined using an XML query language [LPV00, PV01]. 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). Then, the mediator's IVDs can be defined in terms of CMs (*global-as-view*) and thus make use of a semantically richer model involving class hierarchies, complex object structure, properties of relationships (relational constraints, cardinalities, ...) etc.

Model-Based Mediation: The Protagonists

The underlying methodology and procedures of MBM involve users in different roles and at different levels:

• ***Data providers*** are typically domain experts, *e.g.*, bench scientists who would like to make their data from experimental studies available to the community. In MBM, a data provider can not only export an XML-queryable version of her data, but also *domain semantics* by lifting the exported data and schema information from a structural level (*e.g.*, XML DTDs) to the level of conceptual models.[4] Allowing data providers to situate or "*contextualize*" (Example 4) their primary data themselves has significant benefits. First, data providers know best where their data fits on the glue maps. Second, even without the IVDs defined by mediation engineers, data is automatically associated across different sources via their domain/process map contexts.

• ***View providers*** specify integrated view definitions (IVDs), *i.e.*, program complex views in an expressive declarative rule-language. The IVDs are defined over the registered complex sources $\mathsf{CM}(S_1), \mathsf{CM}(S_2), \ldots$ and the "glue knowledge" sources in the mediator's repository. Thus view providers are the actual mediation engineers and bring together (as a team or individually) expertise in the application domain and in databases and knowledge-representation.

The new "fused" objects defined by an IVD can be contextualized, based on the contexts provided by the source conceptual models (see right-hand side in Figure 6.6). In this way, an integrated, virtual view exported by the mediator becomes a "first-class citizen" of the federation, *i.e.*, is considered a conceptual level source $\mathsf{CM}(M)$ itself and can be used just like any original CM-wrapped source.

• ***End users*** can start by *semantic browsing* of GMs, *i.e.*, by navigating the domain and process ontologies in the style of topic maps, where a user navigates through a concept space by following certain relationships, going up and down concept hierarchies etc. A user may also focus her view by issuing graph queries over domain or process maps which return only the subgraphs of interest. Eventually, the user can access raw data from different sources, which is (due to contextualization) automatically organized by context [GLM01], and access derived data resulting from user queries against the mediated views.

Conceptual Models and Registration of Sources at the Mediator

We can distinguish the following components of the conceptual model CM of a source S:

$$\mathsf{CM}(S) = \mathsf{OM}(S) \cup \mathsf{ONT}(S) \cup \mathsf{CON}(S)$$

The different logical components an their dependencies are depicted in Figure 6.3:

[4]XML Schema and similar efforts like RELAX NG play an intermediate role between purely structure-based models (DTDs) and richer semantic models with constraint mechanisms etc.

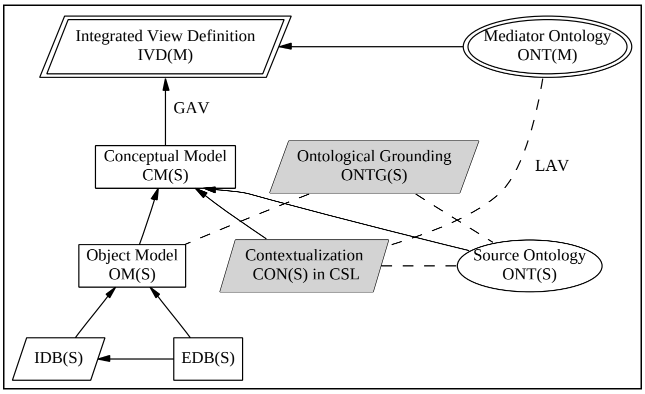


Figure 6.3: Model-based mediation: dependencies between logical components

- $\mathsf{OM}(S)$ is the *object model* of the source S and provides signatures for *classes*, *associations* between classes, and *functions*. $\mathsf{OM}(S)$ structures can be defined extensionally by facts (EDB), or intensionally via rules (IDB).

- $\mathsf{ONT}(S)$ is the *local ontology* of the source S, *i.e.*, defines concepts and their relationships from the source's perspective.

- $\mathsf{ONTG}(S)$ is the *ontological grounding* of $\mathsf{OM}(S)$ in $\mathsf{ONT}(S)$, *i.e.*, a mapping between the object model $\mathsf{OM}(S)$ (classes, attributes, associations) and the concepts and relationships of $\mathsf{ONT}(S)$.

- $\mathsf{CON}(S)$ is the *contextualization* of the local source ontology relative to a mediator ontology $\mathsf{ONT}(M)$.

- $\mathsf{IVD}(M)$ is the mediator's *integrated view definition* and comprises logic view definitions in terms of the sources' object models $\mathsf{OM}(S)$ and the mediator's ontology $\mathsf{ONT}(M)$. By posing queries against the mediator's $\mathsf{IVD}(M)$, the user has the illusion to interact with a single, semantically integrated source instead of interacting with independent, unrelated sources.

In the following, we present the local parts of $\mathsf{CM}(S)$, *i.e.*, $\mathsf{OM}(S)$, $\mathsf{ONT}(S)$, and $\mathsf{ONTG}(S)$ through a running example. For details on the contextualization $\mathsf{CON}(S)$ see Example 4 and [GLM02a].

Example 3 (Cell-Centered Database: CCDB) Figure 6.4 shows pieces of a simplified version of the conceptual model $\mathsf{CM}(\mathsf{CCDB})$ of a real-world scientific information source called the *Cell-Centered Database*, [MGW$^+$02]. The database consists of a set of EXPERIMENT objects. Each experiment collects a number of cell IMAGES from one or more instruments. For each image,

Classes in OM(CCDB)

EXPERIMENT(<u>id</u>:id, date:date, cell_type:string, images:SET(image)).

IMAGE(<u>id</u>:id, instrument:ENUM{c_microscope, e_microscope}, resolution:float, size_x:int, size_y:int, depth:int, structures:SET(structure), regions:SET(deposit)).

STRUCTURE(<u>id</u>:id, name:string, length:float, surface_area:float, volume:float, bounding_box:Cube).

DEPOSIT(<u>id</u>:id, substance_name:string, deposit_type:string, relative_intesity:ENUM{dark,normal,bright}, amount:float, bounding_box:Cube).

...

Associations in OM(CCDB)

co_localizes_with(DEPOSIT.substance_name, DEPOSIT.substance_name, STRUCTURE.name).

surrounds(s1:STRUCTURE, s2:STRUCTURE).

...

Functions in OM(CCDB)

deposit_in_structure(DEPOSIT.id) $\rightarrow$ SET(STRUCTURE.name)

...

Source Ontology – ONT(CCDB)

brain $\xrightarrow{has(co)}$ cerebellum $\xrightarrow{has(co)}$ cerebellar cortex $\xrightarrow{has(co)}$ vermis (ONT1)

dendrite $\xrightarrow{has(co)}$ spine process $\xrightarrow{has(pm)}$ spine (ONT2)

cell $\xrightarrow{projects\_to}$ brain_region

globus_pallidus $\xrightarrow{isa}$ brain_region. ... denaturation $\xrightarrow{isa}$ process. (ONT3)

$tc\_has(co)$:= **transitive_closure**($has(co)$). $tc\_has(pm)$:= **transitive_closure**($has(pm)$). (ONT4)

has_co_pm := **chain**($tc\_has(co), tc\_has(pm)$) (ONT5)

...

Ontological Grounding – ONTG(CCDB)

domain(STRUCTURE.volume) **in** [0,300]

domain(STRUCTURE.name) **in** $tc\_has(co)$(cerebellum) (OG1)

domain(EXPERIMENT.cell_type) **in** $tc\_has(co)$(cerebellum) (OG2)

EXPERIMENT.cell_type $\xrightarrow{projects\_to}$ globus_pallidus (OG3)

DENATURED_PROTEIN $\xrightarrow{exhibits}$ denaturation. (OG4)

...

Figure 6.4: Conceptual model for registering the Cell-Centered Database [MGW+02]

the scientists mark out cellular STRUCTURES in the image and perform measurements on them [MGW+02]. They also identify a second set of regions, called DEPOSITs, in images that show the deposition of molecules of proteins or genetic markers. In general, a region marked as deposit does not necessarily coincide with a region marked as a structure. □

Note that OM(CCDB) in Figure 6.4 includes classes that are instantiated with observed data, *i.e.*, the extensional database EDB(CCDB). In addition to classes, OM(CCDB) stores *associations*, which are n-ary relationships between object classes. The association co_localizes_with specifies which pairs of substances occur together in a specific structure. The object model also contains *functions*, such as the domain specific methods that can be invoked by a user as part of a query. For example, when the mediator or another client calls the function CCDB.deposit_in_structure(), and supplies the *id* of a deposit object, the function returns a set of STRUCTURE objects that spatially overlap with the specified deposit object.

Next, we describe the source's local ontology, ONT(CCDB). Here, an ontology ONT(S) consists of a set of *concepts* and inter-concept *relationships*[5], possibly augmented with additional inference rules and constraints.[6] The ontological grounding ONTG(S) links the object model OM(S) to the source ontology ONT(S). The source ontology serves a number of different purposes:

Creating a Terminological Frame of Reference. For defining the terminology of a specific scientific information source, the source declares its own controlled vocabulary through ONT(S). More precisely, ONT(S) comprises the terms (*i.e.*, *concepts*) of this vocabulary and the *relationships* among them. The concepts and relationships are often represented as nodes and edges of a directed graph, respectively. Two examples of interconcept relations are has(co) and has(pm) which are different kinds of part-whole relationships [7]. In Figure 6.4, items ONT1 and ONT2 show fragments of such a concept graph. Once a concept graph is created for a source, one may use it to define additional constraints on object classes and associations.

Semantics of Relationships. The edges in the concept graph of the source ontology represent inter-concept relationships. Often these relationships have their own semantics that have to be specified within ONT(S). Item ONT4 declares two new relationships tc_has(co) and tc_has(pm). After registration, the mediator interprets this declaration and creates the new (possibly materialized) transitive relations on top of the base relations has(co) and has(pm) provided by the source S. Similarly, the item ONT5 is interpreted by the mediator using a higher-order rule for chaining binary relations:

- chain(R1,R2)(X,Y) **if** R1(X,Z), R2(Z,Y)

With this, ONT5 creates a new relationship has_co_pm(X,Y) provided that there is a Z such that tc_has(co)(X,Z), *and* tc_has(pm)(Z,Y).

[5] most formal approaches, *e.g.*, based on description logic, consider binary relationships only

[6] *e.g.*, ONT4, ONT5 in Figure 6.4 define virtual relations such as *transitive closure* over the base relations

[7] By standards of meronyms, there are different kinds of the has relation: component-object has(co), portion-mass has(pm), member-collection has(mc), stuff-object has(so), place-area has(pa) etc. [AFGP96]

Ontological Grounding of $\mathsf{OM}(S)$. A local domain constraint specifies additional properties of the given extensional database, and thereby establishes an *ontological grounding* $\mathsf{ONTG}(S)$ between the local ontology $\mathsf{ONT}(S)$ and the object model $\mathsf{OM}(S)$ (Figure 6.3). Items (OG1–OG2) in Figure 6.4 refines the domains of the attributes EXPERIMENT.cell_type and STRUCTURE.name from the original type declaration (STRING). The refinement constrains them to take values from those nodes of the concept graph that are *descendants* of the concept cerebellum through the has(co) relationship.

This constraint illustrates an important role of the local ontology in a "conceptually lifted" source. By constraining the domain of an attribute to be concept name C, the corresponding object instance o is "semantically about" C. In addition, this also implies that o is about any ancestor concept C' of C where ancestor is defined via has(co) edges only. Similarly, if a specific instance, STRUCTURE.name has the value 'spine process', it is also about 'dendrite' (ONT2 in Figure 6.4).

In addition to linking attributes to concept names, a constraint may also involve inter-concept relationships. Let us assume projects_to(cell, brain_region) is a relationship in the source ontology ONT(CCDB). A constraint may assert that for all instances e of class EXPERIMENT, projects_to(e.cell_type, 'globus_pallidus') holds (OG3). The constraint thus *refines* the original relationship projects_to to suit the specific semantics of OM(CCDB). Such constraint-defined correspondences between $\mathsf{OM}(S)$ and $\mathsf{ONT}(S)$ are used in the contextualization process [GLM02a].

Intensional Definitions. In the CM wrapper of a source S, we can define virtual classes and associations that can be exported to the mediator as first-class, queriable items by means of an intensional database (view definition) $\mathsf{IDB}(S)$. For example, we can create a new virtual class called DENATURED_PROTEIN in IDB(CCDB) via the rule:

> DENATURED_PROTEIN(ProtName) **if**
>
> > DEPOSIT(ID, ProtName, protein, dark, _, _), deposit_in_structure(ID) $\neq \emptyset$

Thus, an instance of a DENATURED_PROTEIN is created when a "dark" protein deposit is recorded in an instance of DEPOSIT, and there is some structure in which this deposit is found. As a general principle of creating a CM wrapper, such a definition will be supplemented by additional constraints to connect it to the local ontology. For example, assume that ONT(CCDB) already contains a concept called process. Item (ONT3) defines denaturation as a specialization of process. We can now add the constraint (OG4) to complete the semantic specification about the new DENATURED_PROTEIN object.

Contextual References. It is a standard practice for scientific data sources to tag object instances with attributes from a public standard, and to use controlled vocabularies for the values of some of these attributes. For example, the source can specify that the domain of the DEPOSIT.id field can be accessed through an internal method, which, given a protein name, gets its id from a specific database. For example, we can use get_expasy_protein_id to retrieve this information from the SWISS-PROT database on the web.[8] How the source enforces this integrity constraint is internal to the source and not part of its conceptual export schema.

Interplay between Mediator and Sources

In order to address the source registration issue, we have to specify which components of an existing n-source federation can be "seen", *i.e.*, accessed by the new, $n+1^{st}$ source. A federation at the mediator consists of: (i) currently *registered conceptual models* $\mathsf{CM}(S)$ of each participating source S, (ii) one or more *global ontologies* $\mathsf{ONT}(M)$ residing at the mediator that have been used in the federation, and (iii) *integrated views* $\mathsf{IVD}(M)$ defined in a global-as-view (GAV) fashion.

Typical mediator ontologies $\mathsf{ONT}(M)$ are *public*, *i.e.*, serve as domain-specific expert knowledge and thus can be used to "glue" conceptual models from multiple sources. Examples of such ontologies are the Unified Medical Language System (UMLS) from the National Library of Medicine[9] and the Biological Process Ontology from the GeneOntology Consortium[10]. In the presence of multiple ontologies, *articulations*, *i.e.*, mappings between different source ontologies [MWK00] can be used to register with the mediator information about inter-source relationships. Note that a source S usually cannot see all of the above components (i–iii) when defining its conceptual model: While S sees the mediator's ontologies $\mathsf{ONT}(M)$ and thus can define its own conceptual model $\mathsf{CM}(S)$ relative to the mediator's ontology in a *local-as-view* (LAV) fashion, it cannot directly employ *another* source's conceptual model $\mathsf{CM}(S')$, nor can it query the mediator's integrated view $\mathsf{IVD}(M)$ which is defined *global-as-view* (GAV) on top of the sources. The former is no restriction, since S' can register $\mathsf{CM}(S')$, in particular $\mathsf{ONT}(S')$ with the mediator, at which point S can indirectly refer to registered concepts of S' via $\mathsf{ONT}(M)$. The latter guarantees that query processing in this setting does not involve "recursion through the web", *i.e.*, between a source S

[8] `http://www.expasy.ch`

[9] `http://www.nlm.nih.gov/research/umls/`; strictly speaking a metathesaurus, *i.e.*, a "semi-formal" ontology with a limited set of predefined relationships such as broader-term/narrower-term

[10] `http://www.geneontology.org/process.ontology`

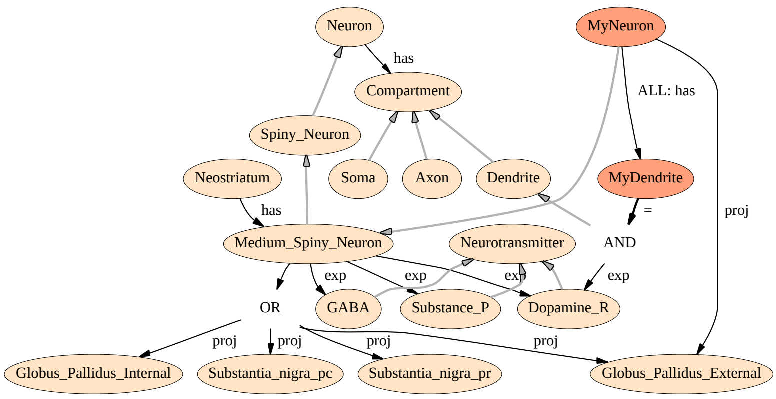


Figure 6.5: A domain map DM after situating new concepts `MyNeuron` and `MyDendrite` (dark)

and the mediator M (the dependency graph in Figure 6.3 is acyclic).[11]

Example 4 (Contextualization: Local-as-View)

Consider the domain map in Figure 6.5. Lighter-colored nodes correspond to concepts that the mediator "understands" and which a source can see. Now assume a source S wants to register information about specific neurons and their dendrites, but the mediator ontology (domain map) does not have dedicated names for those specific kinds of neurons and dendrites. In MBM this problem is solved by contextualising the new local source concepts as views on the mediator's global concepts: In Figure 6.5 the darker-colored source concepts are "hooked" to the mediator's domain map, thereby defining their meaning relative to the mediator's concepts. This is achieved by sending the following first-order axioms (here in description logic syntax) to the mediator:

$$\texttt{MyDendrite} \equiv \texttt{Dendrite} \sqcap \exists \texttt{exp.Dopamine\_R}$$

$$\begin{aligned}\texttt{MyNeuron} \sqsubseteq\ & \texttt{Medium\_Spiny\_Neuron} \\ & \sqcap \exists \texttt{proj.Globus\_pallidus\_external} \\ & \sqcap \forall \texttt{has.MyDendrite}\end{aligned}$$

Thus instances of `MyDendrite` are exactly those dendrites that express Dopamine R(eceptor), and `MyNeuron` objects are medium spiny neurons projecting to Globus Pallidus External and *only* have `MyDendrite`s. Assuming properties are *inherited* along the transitive closure of `isa`, it follows that `MyNeuron`, like any `Medium_Spiny_Neuron` projects to certain structures (`OR` in Figure 6.5). With the newly registered knowledge, it follows that `MyNeuron` definitely projects to Globus Pallidus External. If we want to specify that it *only* projects to the latter, a *nonmonotonic inheritance*, *e.g.*, using FL with well-founded semantics can be employed. □

[11] At the cost of loss of efficiency, the restriction "no recursion through the web" could be lifted.

Note that the intuitive graphical contextualization depicted in Figure 6.5 is not unique, *i.e.*, logically equivalent domain maps may have different graph representations.[12] For domain maps that can be completely axiomatized using a description logic, a reasoning system such as FaCT [Hor98] can be employed in order to compute the deductive closure, in particular, to derive a unique concept hierarchy and to check consistency of a domain map.

6.4 Knowledge Representation for Model-Based Mediation

We now take a closer look at the principal mechanisms for specifying "glue knowledge", *i.e.*, ontologies in the form of domain maps (DMs) and process maps (PMs).

6.4.1 Domain Maps

As is standard for ontologies, domain maps name and specify relevant concepts by describing the characteristic relationships among them [LGM01]. In this way, DMs provide the basic domain semantics which is needed to glue data across different sources in multiple-world scenarios. DMs can be depicted more intuitively in the form of labeled directed graphs. In contrast to many other graph-based notations, however, DMs have a solid formal semantics via a translation to logic rules (see below). The graph form of DMs is defined as follows.

Definition 1 (Domain Maps) Let $\mathcal{C}$ be a set of symbols called *concepts*, $\mathcal{R}$ a set of *roles*. A *domain map* DM is a directed labeled graph with nodes $\mathcal{C}$. A concept $C \in \mathcal{C}$ can be understood as denoting a class of objects sharing a set of common properties. In order to understand how a concept C is defined relative to other concepts, we have to inspect its outgoing edges. We write $c \in C$ to denote that c is an *instance* of concept C.[13] We distinguish the following types of edges in DMs:

1. $C \stackrel{isa}{\rightarrow} D$ (short: $C \rightarrow D$) defines that *every C isa D*, *i.e.*, $c \in C$ implies $c \in D$.

   Since this subconcept/subclass relation is very common in DMs, we usually omit the *isa* label and use the shorthand notation $C \rightarrow D$ instead.

2. $C \stackrel{ex:r}{\rightarrow} D$ defines that for every $c \in C$, there *exists some* r-related $d \in D$.

[12]Similar to the fact that the same query can have many different syntactic representations. In general, equivalence of first-order (or SQL) queries is not decidable.

[13]Thus, we can view C and D as unary predicates.

Here, $r \in \mathcal{R}$ is a *role*, *i.e.*, a *binary relation* $r(c, d)$ between instances of C and D.

3. $C \overset{all:r}{\rightarrow} D$ defines that for every $c \in C$ and *all* x which are r-related to c (*i.e.*, for which $r(c, x)$ holds), we have $x \in D$.

4. $C \xrightarrow{r} D$ defines that if $c \in C$ and $d \in D$, then they are r-related, *i.e.*, $r(c, d)$ holds.

5. $\texttt{AND}{\rightarrow}_i\{D_1, \ldots, D_n\}$, *i.e.*, an `AND`-node with n outgoing edges to $D_1, \ldots, D_n$, respectively, defines an anonymous concept, the *intersection* of concepts $D_1, \ldots, D_n$.

6. $\texttt{OR}{\rightarrow}_i\{D_1, \ldots, D_n\}$, *i.e.*, an `OR`-node with n outgoing edges to $D_1, \ldots, D_n$, respectively, defines an anonymous concept, the *union* of concepts $D_1, \ldots, D_n$.

7. $C \overset{\equiv}{\rightarrow} D$ defines that C is *equivalent* to D, *i.e.*, every C *isa* D and vice versa. We could have denoted this also as "$C{\leftrightarrow}D$", however, the directed edge keeps the distinction between C (the definiendum) and its definition D (definiens). □

Note that D can be an atomic or a defined concept. When unique, `AND` nodes are omitted and outgoing arcs directly attached to the concept being defined. In Figure 6.5, unlabeled, grey edges and edges labeled "proj" (*projects-to*) correspond to "*isa*"-edges and "*ex*:proj"-edges, respectively.

Reified Roles as Concepts. In DMs, as in description logics, it is the concepts that are being defined, whereas the roles are only a means to that end. In order capture the semantics of roles, *i.e.*, define their properties in terms of each other, we have to turn them into concepts themselves. In logic this "quoting mechanism" is known as reification.

Example 5 (Roles as Concepts) Consider a DM involving the roles *regulates*, *activates*, and *inhibits* and assume that in the given domain, *activates*(C, D) and *inhibits*(C, D) are special cases of *regulates*(C, D). Instead of introducing a special notation for "sub-roles"[14], and then defining the mechanics of how roles can be related to one another, we turn roles into "first-class citizens" by making them concepts using a operator *mc* ("make-concept"). Now we can apply the modeling capabilities of DMs to roles and, *e.g.*, simply state that $mc(activates) \overset{isa}{\rightarrow} mc(regulates)$. □

By modeling roles a concepts, more domain semantics can be formalized, leading to better "knowledge engineering". In particular, during *query processing*, such formalized knowledge can be automatically employed by the system: Given a DM (formalized as logic rules), an MBM query

[14] RDF(S) has such a notion called *subproperty*.

or view definition involving *activates* and *regulates* "knows" that the former is a subconcept of the later. If during query processing, *e.g.*, a goal `regulates('cAMP',Protein)` is evaluated, the logic rules corresponding to the DM knowledge allow the system to deduce that any result for `activates('cAMP',Protein)` is also an answer for `regulates('cAMP',Protein)`. This is correct since a *substitutability principle* holds which allows the system to replace a concept D with any of its subconcepts C, *i.e.*, for which $C \overset{isa}{\to} D$ holds.

Generating the Role Hierarchy. When making a role into a concept, the *isa* hierarchy[15] on concepts induces an *isa* hierarchy on roles.

Example 6 (Roles as Concepts, *Cont'd*) Consider a DM stating that `NProt` $\overset{isa}{\to}$ `Protein`, `NProt` `regulates` some `Gene`, and `cfos` $\overset{isa}{\to}$ `Gene`.[16] The role `regulates` is conceptualized by asserting $mc(\texttt{regulates})$. When making its hidden arguments visible, we note that $mc(\texttt{regulates}(C, D))$ really denotes a *family* of `regulates` concepts. The *isa* hierarchy on `regulates` concepts is derived from the *isa* hierarchy of its arguments. For example, we have:

$$mc(\texttt{regulates(NProt,cfos)}) \overset{isa}{\to} mc(\texttt{regulates(NProt,Gene)}) \overset{isa}{\to} mc(\texttt{regulates(Protein,Gene)}) \quad \square$$

Deriving the Role Hierarchy. Above we introduced the unary operator *mc* which turns role literals into concepts. We implement it in FL as a subclass of the (meta-)class `concept` by asserting "`mc :: concept`" and adding further rules for deriving the role hierarchy from the concept hierarchy, given as set of *mc*-declarations such as $\texttt{r(C,D):mc}$ by the user:

- $\texttt{r(C,D):mc}, \texttt{r(C}',\texttt{D}'\texttt{):mc}, \texttt{r(C,D)} :: \texttt{r(C}',\texttt{D}'\texttt{)}$ **if** $(\texttt{r(C,D):mc} \lor \texttt{r(C}',\texttt{D}'\texttt{):mc}), \texttt{C} :: \texttt{C}', \texttt{D} :: \texttt{D}'$ *(up/down)*
- $\texttt{r(C,D):mc}, \texttt{r(C}',\texttt{D}'\texttt{):mc}, \texttt{r(C,D)} :: \texttt{r(C}',\texttt{D}'\texttt{)}$ **if** $(\texttt{r(C,D}'\texttt{):mc} \lor \texttt{r(C}',\texttt{D):mc}), \texttt{C} :: \texttt{C}', \texttt{D} :: \texttt{D}'$ *(mixed)*

Observe that with the rules, we get indeed the desired result in Example 6.

Recursive Concepts. Consider the part-of relationship `has_a` and its interaction with `isa`. For example, since `MyNeuron isa Medium_Spiny_Neuron` which in turn `has_a Neostriatum`, we conclude that `MyNeuron has_a Neostriatum` (Figure 6.5). In the general case, this gives rise to a recursive rule "if $C \overset{isa}{\to} D$ and $D \overset{has\_a}{\to} E$ then $C \overset{has\_a}{\to} E$". Similarly, one can define that `isa` and `has_a` are independently transitive and that `isa` is antisymmetric. For such recursive definitions, an intuitive graph notation can be devised, *e.g.*, using a dashed edge for the concept being defined

[15] Strictly speaking, the *isa* does not have to be a hierarchy but can be any directed acyclic graph.

[16] Here, `NProt` stands for *nuclear protein.*

(cf. [LHL+98, pp.601]) to its recursive definition. In a declarative rule-based query language like FL, an executable specification is:

- $\texttt{has\_a(X,Z)}$ **if** $\texttt{X :: Y, has\_a(Y,Z)}$.

Note that here $\texttt{X,Y,Z}$ are concept variables. Such FL rules can also be used at the mediator to handle inductive definitions such as (ONT4) in Figure 6.4, in particular, when the source does not have the capability to evaluate recursive definitions.

Parameterized Roles and Concepts. Part-of relationships like `has_a` come in different flavors F, *e.g.*, $F \in \{\mathit{member/collection}, \mathit{portion/mass}, \mathit{phase/activity}, \ldots\}$ and transitivity does not necessarily carry over across flavors [AFGP96].[17] This is most naturally modeled by a *parameterized role* $\texttt{has\_a(}F\texttt{)}$ which is transitive *within* each flavor F but which may interact in other ways *across* flavors. It is straightforward to extend Definition 2 for parameterized roles and concepts: *e.g.*, assume the parameterized role $\texttt{has\_a(}F\texttt{)}$ should hold between concepts C and D only for some flavors F satisfying a condition $\varphi(F)$. We can extend Ψ and compile such a parameterized DM edge into FL as follows:

- $\Psi(C \overset{has\_a(F) \| \varphi(F)}{\longrightarrow} D) = \{\texttt{has\_a(F)(c,d)}\ \textbf{if}\ \texttt{c:C, d:}\Phi(D), \varphi(\texttt{F})\} \cup \Psi(D)$.

Note that a parameterized role like `has_a(F)` has a first-order semantics in FL despite its higher-order syntax [KLW95].

6.4.2 Process Maps

Process maps (PMs) provide abstractions of "process knowledge", *i.e.*, temporal and/or causal relationships between events that can be used for situating and linking data across different sources. Like DMs, PMs are directed labeled graphs, albeit with a very different semantics: Nodes are used to model *states* while edges correspond to *state transitions* which are labeled with a process name describing the transition. In this way, data providers, *e.g.*, bench scientists can not only hook their raw data to the (given or refined) DMs, but also to processes that are witnessed in their experimental studies databases (cf. Figure 6.2 and Figure 6.8).

Initial Process Semantics $\mathbf{PM}_0$. Intuitively, an edge of the form $e_\pi = s \overset{\{\varphi\}\pi\{\psi\}}{\longrightarrow} s'$ of a PM means that the process π leads from state s to s'; φ is a necessary *precondition* that must hold in

[17] For example, *orchestra has_a musician* and *musician has_a arm*, but not *orchestra has_a arm*.

s for π to happen, ψ is a *postcondition* which holds in s' as a result of π. By PM_0 we denote the set of all initial process semantics.

We call the edge e_π of a PM, a *process occurrence* of π in PM. Thus, a process occurrence specifies where in a PM a process occurs, and which pre- and postconditions φ and ψ this occurrence satisfies. In addition to the semantics implied by the occurrence of e_π in PM, a process π can have an *initial semantics* associated with the process name π.

To allow for *parameterization* of processes, we consider edge labels where process names are *first-order atoms*, *i.e.*, $\pi = \pi(T_1, \ldots, T_n)$ where each term T_i is a logic variable or constant. For example, consider $\pi = \mathtt{opens}(\mathtt{Channel})$ describing the opening process of an ion channel. We can define its initial semantics by the expression

$$\{\neg\, \mathtt{open}(\mathtt{Channel})\}\ \mathtt{opens}(\mathtt{Channel})\ \{\mathtt{open}(\mathtt{Channel})\}$$

meaning that *any* transition along a process occurrence of $\pi = \mathtt{opens}(\mathtt{Channel})$ in a PM must be from a state where $\mathtt{open}(\mathtt{Channel})$ was *false*. In the successor state, however, *i.e.*, after π has happened, we have that $\mathtt{open}(\mathtt{Channel})$ is *true*.

From Process Maps to Domain Maps. We call the first-order predicates occurring in φ and ψ like "$\mathtt{open}(\mathtt{Channel})$" *fluents*, since their truth is state dependent. We require that the set of *fluent predicate symbols* $\mathcal{F}$ is disjoint from the set $\mathcal{P}$ of *process names* and the sets of concept and role names $\mathcal{C}$ and $\mathcal{R}$, respectively. In contrast, the constant parameters used in process occurrences, like "$\mathtt{Channel}$" *are* allowed to be concepts from $\mathcal{C}$.

For example, a DM may have that $\mathtt{NMDA\_receptor} \stackrel{isa}{\to} \mathtt{Calcium\_channel} \stackrel{isa}{\to} \mathtt{Channel}$ in which case the process knowledge about the opening of channels and the "static" knowledge from a DM are directly linked through the common concept $\mathtt{Channel}$.

Similarly, just as we have made roles "first-class" citizens by reifying them into concepts, we can do the same for processes, thereby being able to specify additional semantics of processes using domain maps.

Example 7 (Processes as Concepts) Consider the $\mathtt{binds\_to}(\mathtt{X}, \mathtt{Y})$ process with the initial semantics

$$\{\neg\, \mathtt{bound}(\mathtt{X}, \mathtt{Y})\}\ \mathtt{binds\_to}(\mathtt{X}, \mathtt{Y})\ \{\mathtt{bound}(\mathtt{X}, \mathtt{Y})\}$$

Now consider a DM in which we have reified processes as concepts as follows:

$$\mathtt{dimerizes}(\mathtt{X}) \xrightarrow{isa \| \mathtt{X}=\mathtt{Y}} \mathtt{binds\_to}(\mathtt{X}, \mathtt{Y})$$

It is easy to see that this (parameterized) DM edge, when translated into FL, allows the system to conclude in the combined knowledge base DM $\cup$ PM$_0$ that

$$\{\neg\,\texttt{bound(X,X)}\}\ \texttt{dimerizes(X)}\ \{\texttt{bound(X,X)}\}.$$

□

Process Elaboration and Abstraction. The edge e_π of a process occurrence can be seen as an *abstraction* of a real process. In addition to its initial semantics PM$_0$, and the semantics induced by its concrete occurrence in a specific PM, we can *elaborate* this abstraction by replacing the e_π with a (sub-)process map $\mathsf{elab}(e_\pi)$ whose initial and final states are s and s'. The newly created nodes and edges of the elaboration $\mathsf{elab}(e_\pi)$ are annotated with the same unique *elaboration identifier eid*. The *eid* includes at least a reference to e_π indicating the edge being elaborated, and the *author* (*e.g.*, data provider) of the elaboration.

The converse of elaboration, *abstraction*, takes a connected subgraph $\Pi(S, s_0, s_f, E)$ with nodes S, edges E, and distinguished nodes $s_0, s_f \in S$ (initial and final state), and abstracts Π into a single edge $e_\pi = \mathsf{abstract}(\Pi(S, s_0, s_f, E))$. The abstracted edges E of Π are marked with a unique *abstraction identifier aid*, which includes a reference to the new abstraction edge e_π, and the author of the abstraction. See Appendix A.2 for further details on process maps.

6.5 Model-Based Mediator System and Tools

At the core of the model-based mediation framework is the KIND mediator system. Other important components are the SMART Atlas for annotating, displaying, and relating data with brain atlases, the Cell-Centered Database CCDB as our primary source of experimental data, and the Know-ME tool for concept-based navigation of source and mediated views. For a description of the latter see [QLMG02]; the other components are described next.

The KIND Mediator Prototype

The architecture of the KIND (Knowledge-based Integration of Neuroscience Data) mediator system is depicted on the left in Figure 6.6. On the right a snapshot of the prototype execution is shown: after the user issues a query against the integrated view, the system situates the results on a domain map, in this case ANATOM (simple ontology of brain anatomy). By clicking on the orange "diamonds", the user can retrieve the actual result objects, grouped by concept (foreground).

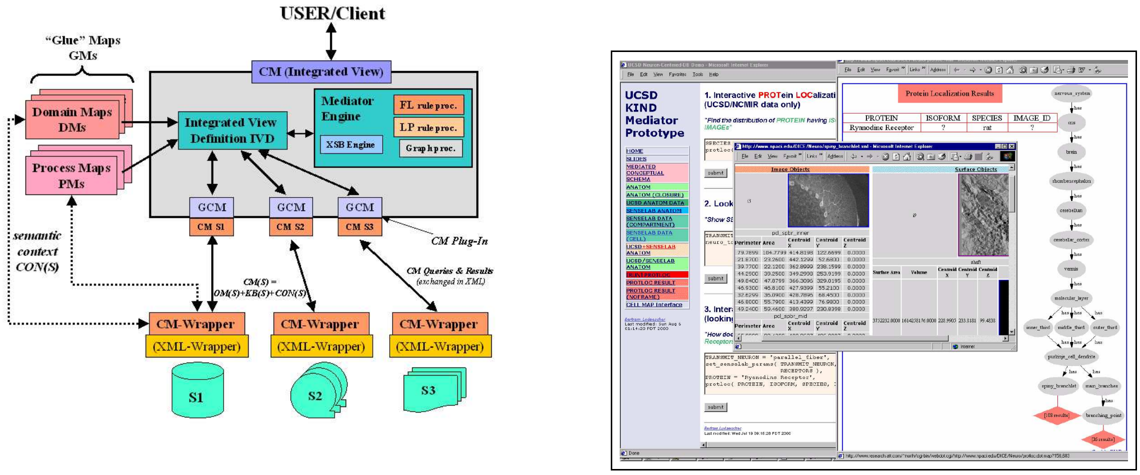


Figure 6.6: **Left**: architecture of the KIND model-based mediator. **Right**: snapshot of the prototype: *background left*: meditor shell for issuing ad-hoc queries against CM(M); *background right*: generated subgraph having the requested result data shown in its anatomical context; clicking on a (diamond) result node retrieves the actual result data (*foreground center*).

In our first prototype [LGM00, GLM01] we have used the F-logic implementation Flora [YK00] as the only query processing and deduction engine. As part of a large collaborative project [BIR01] we are currently reimplementing our prototype as a modular, distributed mediator system that includes several additional components, for example:

- *Logic plan generator*: Given a user query Q and an integrated view definition IVD, $Q \circ \mathsf{IVD}$ is translated into a plan generator program $PG(Q \circ \mathsf{IVD})$ which, when executed, produces an initial logic query plan for $Q \circ \mathsf{IVD}$.

- *Query rewriter*: This module takes a logic query plan and rewrites it into an executable, distributed plan based on the capabilities of a source (*e.g.*, conjunctive queries with binding patterns or complete SQL).

- *Execution plan compiler*: For final execution, the rewritten plan is compiled into a logic program whose runtime execution sends the corresponding subqueries to wrapped sources, retrieves results, and post-processes them (*e.g.*, joins, group-bys, and unions across sources) before sending them to the user.

- *SQL plan generator*: For relational sources, *i.e.*, having SQL query capabilities, this wrapper module translates a logic query plan into an equivalent SQL statement, similar to [Dra92].

A preliminary version of this new system has been recently demonstrated [GLM02a] and includes all of the above modules. Plan generation and rewriting is implemented using logic programming technology [Lud02], the SQL plan generator has been implemented in Java. It is planned that the final system will include specialized inference engines such as FLORA and XSB [SSW94] for handling deductive and object-oriented database capabilities, and FaCT [Hor98] for reasoning tasks over domain maps which are formalized in description logics.

The Cell-Centered Database and SMART Atlas: Retrieval and Navigation Through Multi-Scale Data

The Cell-Centered Database (CCDB) mentioned earlier in Example 3 houses different types of high resolution 3D light and electron microscopic reconstructions of cells and subcellular structures produced at the National Center for Microscopy and Imaging Research[18] [MGW$^+$02]. It contains structural and protein distribution information derived from confocal, multiphoton and electron microscopy, including correlated microscopy. Many of the data sets are derived from electron tomography, a powerful technique for deriving 3D information from electron microscopic specimens. Electron tomography is similar in concept to medical imaging techniques like CAT scans and MRI in that it derives a 3D volume from a series of 2D projections through a structure. In this case, the structures are contained in sections prepared for electron microscopy, which are tilted through a limited angular range. On the left of Figure 6.7 examples of datasets in the CCDB are shown.

A screenshot of the SMART ATLAS (Spatial Markup and Rendering Tool) is shown on the right of Figure 6.7. The tool is based on a geographic mapping tool [Zas00] and allows users to define polygons on a series of 2D vector images and annotate them with names, relationships, and concept IDs from an ontology such as UMLS. This tool provides another kind of "glue map" (in addition to domain and process maps); here, in the literal sense: First, a brain atlas such as Paxinos and Watson [PW98] is translated into a spatial format (*e.g.*, Scalable Vector Graphics: SVG). The user then "marks up" the atlas using the SMART ATLAS tool, *e.g.*, with concepts names from UMLS. Once the atlas has been (partially) marked up, it can be queried from the same browser: Clicking on any point in the atlas will return the stereotaxic coordinates; clicking on a brain region will return the name of that region, along with any synonyms, and highlight all planes containing that structure. The SMART ATLAS can now be used to register a researcher's

[18] `http://www.ncmir.ucsd.edu`

Figure 6.7: **Left**: Examples of tomographic data sets in the CCDB. A and B show a selectively stained spiny dendrite from a Purkinje cell. A is a projection of the volume reconstruction (dendrite appears as white against dark background). B is the segmented dendrite. C and D show a tomographic reconstruction of the node of Ranvier. C is a single computed slice through the volume. D is a surface reconstruction of the various components comprising the node. Scale bar in B= $1\mu m$; in C = $0.5\mu m$. **Right**: Registration of a data set with the Smart Atlas. The user draws a polygon representing the location of a data set, in this case a filled Purkinje neuron. The user specifies the data base containing this data, then enters an annotation and selects a concept from the UMLS or some other ontology. The concept ID is stored back in the database.

data to a specific spatial location. This also links the registered data automatically to the UMLS ontology by virtue of the earlier semantic markup of spatial objects. To register source data, the user draws an arbitrary polygon representing the approximate data location on one of the atlas planes (Figure 6.7: right). The user is then presented with a form which can be used to add annotations or to provide additional links to concepts of an ontology. Although the UMLS is used in the examples shown here, the user will eventually be able to use multiple ontologies, including those of their own creation, for semantically indexing data. Tools are also being developed to define new terms and relationships in existing ontologies. In [GLM$^+$02b] we have shown how spatial and conceptual information can be used together in a mediator system; see also [MGL$^+$02] for further details on the use of the SMART ATLAS.

6.6 Related Work and Conclusions

Related Work

Significant progress has been made in the general area of data mediation in recent years, and several prototype mediator architectures have been designed by projects like TSIMMIS [GMPQ+95], SIMS [KMA+98], Information Manifold [LRO96], Garlic [HKWY97], and MIX [BGL+99]. While these approaches focus mostly on structural and schema aspects, the problem of *semantic mediation* has also been addressed: In the DIKE system [PTU00], the focus is on automatic extraction of mappings between semantically analogous elements from different schemas. A global schema is defined in terms of a conceptual model (SDR network) where the nodes represent concepts and the (directed) edge labels represent their semantic distances and a score called *semantic relevance* that measures the number of instances of the target node that are also instances of the source node. The correspondence between objects are defined in terms of *synonymies*, *homonymies* and *sub-source similarities*, defined by finding maximal matching between the two graphs.

ODB-Tools [BB01] is a system developed on top of the MOMIS [BCV99] system for modeling and reasoning about the common knowledge between two to-be-integrated schemas. They present the object-oriented language ODL_{I^3} derived from a description logic (OCDL). The language allows a user to create complex objects with finite nesting of values, union and intersection types, integrity constraints and quantified paths. These constructs are used to define a class in one schema as a *generalization*, *aggregation*, or *equivalent* with respect to another; *subsumption* of a class by another can be inferred. An integrated schema is obtained by clustering schema elements that are close to one another in terms of an affinity metric.

Calvanese et al. [CCG+01] perform semantic information integration using an LAV approach by expressing the conceptual schema by a description logic language called $\mathcal{DLR}$, and subsequently defining non-recursive Datalog views to express source data elements in terms of the conceptual model. The language $\mathcal{DLR}$ represents concepts C, relations R, and a set of assertions of the form $C_1 \sqsubset C_2$ or $R_1 \subset R_2$, where R_1, R_2 are $\mathcal{DLR}$ relations with the same arity. Mediation is accomplished by defining "reconciliation correspondences", specifications that a query rewriter uses to match a conceptual level term to data in different sources.

Recently Peim et al. [PFPG02] have proposed an extension to the well-known TAMBIS system [GSN+01]. Their approach is similar to ours [GLM00, LGM01] in that a logic-based ontology (in their case the $\mathcal{ALCQI}$ description logic) interfaces with an "object-wrapped" source. While we use F-logic [KLW95] as the internal knowledge representation and query language, their work focuses on how a query on the ontology is transformed to monoid comprehensions for semantic query optimization.

Summary: Model-Based Mediation and "Reason-able" Metadata

We have presented model-based mediation as a methodology that supports information integration of scientific data across complex multiple-world scenarios as found, *e.g.*, in the Neuroscience domain. In this framework, object-oriented and conceptual models, domain maps and process maps all provide means to capture more domain semantics and thus can act as "glue knowledge sources" to link hard-to-correlate sources. We have presented mechanisms to formally contextualize source data. The graph structures thus constructed have been shown to be useful for navigating across related concepts and querying local data during navigation [QLMG02].

Logic formalizations of domain and process maps can be seen as "reason-able" or "executable" metadata (cf. [Hor02]): Unlike conventional, descriptive metadata which is primarily used for data discovery, formal ontologies such as domain maps and process maps can support much more versatile computational tasks in a mediator system as illustrated in this chapter. For example, different and apparently unrelated data objects can be associated and retrieved together or even "fused" by the mediator's integrated view definition (IVD), since IVDs can be defined as deductive rules over domain maps and process maps (Figure 6.3). In this way, in model-based mediation, logic rules play the role of "executable" or "computational" metadata for scientific data integration. The latter is a challenging application and benchmark for combined database and knowledge representation techniques.

Acknowledgements. This work has been supported by NIH/NCRR 3 P41 RR08605-08S1 (BIRN) and NSF-NPACI Neurosciences Thrust ACI 9619020. The authors thank their colleagues and students involved in the BIRN project for their contributions, in particular, Xufei Qian, Edward Ross, Joshua Tran, and Ilya Zaslavsky.

Bibliography

[AFGP96] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole Relations in Object-Centered Systems: An Overview. *Data & Knowledge Engineering*, 20:347–383, 1996.

[BB01] D. Beneventano and S. Bergamaschi. Extensional Knowledge for semantic query optimization in a mediator based system. In *Intl. Workshop on Foundations of Models for Information Integration (FMII-2001)*, 2001.

[BCV99] S. Bergamaschi, S. Castano, and M. Vincini. Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record*, 28(1):54–59, 1999.

[BGL+99] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-Based Information Mediation with MIX. In *ACM Intl. Conference on Management of Data (SIGMOD)*, pp. 597–599, Philadelphia, PA, 1999. exhibition program.

[BIR01] Biomedical Informatics Research Network Coordinating Center (BIRN-CC), University of California, San Diego. `http://nbirn.net/`, 2001.

[BST+00] O. Bozdagi, W. Shan, H. Tanaka, D. Benson, and G. Huntley. Increasing Numbers of Synaptic Puncta During Late-Phase LTP: N-Cadherin Is Synthesized, Recruited to Synaptic Sites, and Required for Potentiation. *Neuron*, 28(1):245–259, 2000.

[CCG+01] D. Calvanese, S. Castano, F. Guerra, D. Lembo, M. Melchiori, G. Terracina, D. Ursino, and M. Vincini. Towards a Comprehensive Methodological Framework for Semantic Integration of Heterogeneous Data Sources. In *8th Int. Workshop on Knowledge Representation meets Databases (KRDB)*, 2001.

[CGL+98] D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description Logic Framework for Information Integration. In *Principles of Knowledge Representation and Reasoning*, pp. 2–13, 1998.

[CGM90] S. Chakravarthy, J. Grant, and J. Minker. Logic-Based Approach to Semantic Query Optimization. *ACM Transactions on Database Systems (TODS)*, 15(2):162–207, 1990.

[Dra92] C. Draxler. A Powerful Prolog to SQL Compiler. Technical report, Centre for Information and Language Processing, Ludwigs-Maximillians-Universität München, 1992.

[FLOa] FLORA Homepage. `www.cs.sunysb.edu/~sbprolog/flora/`.

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

[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)*, pp. 39–52, Berlin, Germany, July 2000. IEEE Computer Society.

[GLM01] A. Gupta, B. Ludäscher, and M. E. Martone. An Extensible Model-Based Mediator System with Domain Maps. In *17th Intl. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, 2001. exhibition program.

[GLM02a] A. Gupta, B. Ludäscher, and M. E. Martone. Registering Scientific Information Sources for Semantic Mediation. In *21st Intl. Conference on Conceptual Modeling (ER)*, Tampere, Finland, 2002.

[GLM+02b] A. Gupta, B. Ludäscher, M. E. Martone, X. Qian, E. Ross, J. Tran, and I. Zaslavsky. A System for Managing Alternate Models in Model-based Mediation. In 19^{th} *British Natl. Conf. on Databases (BNCOD)*, LNCS, Sheffield, UK, July 2002. Springer.

[GMPQ+95] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages (Extended Abstract). In *Next Generation Information Technologies and Systems*, 1995.

[GSN+01] C. Goble, R. Stevens, G. Ng, S. Bechhofer, N. Paton, P. Baker, M. Peim, and A. Brass. Transparent Access to Multiple Bioinformatics Information Sources. *IBM Systems Journal*, 40(2):534–551, 2001.

[HKWY97] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing Queries Across Diverse Data Sources. In *Intl. Conference on Very Large Data Bases (VLDB)*, pp. 276–285, Athens, Greece, 1997.

[HLLS98] R. Himmeröder, G. Lausen, B. Ludäscher, and C. Schlepphorst. FLORID: A DOOD-System for Querying the Web. In *Demonstration Session at EDBT*, Valencia, Spain, 1998.

[Hor98] I. R. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pp. 636–645. Morgan Kaufmann, San Francisco, California, 1998.

[Hor02] I. Horrocks. DAML+OIL: A Reason-able Web Ontology Language. In *Intl. Conference on Extending Database Technology (EDBT)*, pp. 2–13, 2002. keynote talk.

[KFM01] J. Kasahara, K. Fukunaga, and E. Miyamoto. Activation of Calcium/Calmodulin-dependent Protein Kinase IV in Long Term Potentiation in the Rat Hippocampal CA1 Region. *The Journal of Biological Chemistry*, 276(26):24044–24050, 2001.

[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.

[KMA+98] C. A. Knoblock, S. Minton, J. L. Ambite, P. J. M. N. Ashish, I. Muslea, A. G. Philpot, and S. Tejada. Modeling Web Sources for Information Integration". In *Proc. Fifteenth National Conference on Artificial Intelligence*, 1998.

[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. Model-Based Information Integration in a Neuroscience Mediator System. In *26th Intl. Conf. on Very Large Data Bases (VLDB)*, pp. 639–642, Cairo, Egypt, 2000. Morgan Kaufmann. demonstration session.

[LGM01] B. Ludäscher, A. Gupta, and M. E. Martone. Model-Based Mediation with Domain Maps. In *17th Intl. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, 2001. IEEE Computer Society.

[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. Elsevier/Pergamon.

[LLM98] G. Lausen, B. Ludäscher, and W. May. On Active Deductive Databases: The Statelog Approach. In B. Freitag, H. Decker, M. Kifer, and A. Voronkov, editors, *Transactions and Change in Logic Databases*, number 1472 in LNCS. Springer, 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, pp. 150–165, Konstanz, Germany, 2000. Springer.

[LRO96] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Intl. Conference on Very Large Data Bases (VLDB)*, pp. 251–262, 1996.

[Lud02] B. Ludäscher. Mediator Query Processing with Prolog Technology. Technical Note, 2002. in preparation, Biomedical Informatics Research Network; preliminary version: BIRN-DI-TN-2002-01.

[LYV+98] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. D. Ullman, and M. Valiveti. Capability Based Mediation in TSIMMIS. In *ACM Intl. Conference on Management of Data (SIGMOD)*, pp. 564–566, 1998.

[MGL+02] M. E. Martone, A. Gupta, B. Ludäscher, I. Zaslavsky, and M. H. Ellisman. Federation of Brain Data through Knowledge-Guided Mediation. In R. Kötter, editor, *Neuroscience Databases – A Practical Guide*. Kluwer Academic Publishers, 2002. to appear.

[MGW+02] M. E. Martone, A. Gupta, M. Wong, X. Qian, G. Sosinsky, S. Lamont, B. Ludäscher, and M. H. Ellisman. A Cell-Centered Database for Electron Tomographic Data. *Journal of Structural Biology*, 138:145–155, 2002. see also `http://ncmir.ucsd.edu/CCDB/`.

[MWK00] P. Mitra, G. Wiederhold, and M. L. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Intl. Conference on Extending Database Technology (EDBT)*, pp. 86–100, 2000.

[NPA01] National Partnership for Computational Infrastructure (NPACI) – Neuroscience Thrust Area, 2001. `http://www.npaci.edu/Thrusts/Neuro/`.

[PFPG02] M. Peim, E. Franconi, N. Paton, and C. Goble. Query Processing with Description Logic Ontologies Over Object-Wrapped Databases. In *Intl. Conference on Scientific and Statistical Database Management (SSDBM)*, 2002.

[PGH98] Y. Papakonstantinou, A. Gupta, and L. M. Haas. Capabilities-Based Query Rewriting in Mediator Systems. *Distributed and Parallel Databases*, 6(1):73–110, 1998.

[PTU00] L. Palopoli, G. Terracina, and D. Ursino. The System DIKE: Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses. In *Proc. ADBIS-DASFAA Symposium*, pp. 108–117, 2000.

[PV01] Y. Papakonstantinou and V. Vassalos. The Enosys Markets Data Integration Platform: Lessons from the Trenches. In *Intl. Conference on Information and Knowledge Management (CIKM)*, 2001.

[PW98] G. Paxinos and C. Watson. The Rat Brain in Stereotaxic Coordinates. Academic Press, San Diego, 1998.

[QLMG02] X. Qian, B. Ludäscher, M. E. Martone, and A. Gupta. Navigating Virtual Information Sources with Know-ME. In *EDBT*, number 2287 in LNCS, Prague, Czech Republic, March 2002.

[SSW94] K. F. Sagonas, T. Swift, and D. S. Warren. XSB as an Efficient Deductive Database Engine. In *ACM Intl. Conference on Management of Data (SIGMOD)*, pp. 442–453, 1994.

[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.

[Zas00] I. Zaslavsky. A New Technology for Interactive Online Mapping. *Cartographic Perspectives*, (37):65–77, 2000.

Appendix A

A.1 Domain Maps as Logic Rules

Domain maps borrow from description logics [CGL$^+$98] the notions of *concept* and *roles*. Indeed, while some of the above constructs of DMs have equivalent formalizations in description logic [LGM01], the fact that we need additional mechanisms like *roles as concepts*, *recursive* and *parameterized* roles and concepts, and the fact that we want "executable" DMs during query processing, requires a translation into a more general logic framework.

In the following, we formalize DMs in a minimal subset of F-logic (FL) [KLW95]. The semantics of DMs could be formalized in other languages, in particular in other deductive database languages. The use of FL is convenient since a small subset of it already matches nicely the minimal requirements established for a model-based mediator system [LGM01]. Moreover, implementations of FL are readily available [FLOa, FLOb] and have been used by the authors in different mediator prototypes before [LGM00, HLLS98, LHL$^+$98].

In FL, "$c\colon C$" and "$C :: D$" denote class membership ($c \in C$) and subclassing ($C \subseteq D$), respectively. Thus, there are logic rules of the form "*head* **if** *body*" that express the FL semantics of "**:**" and "**::**", say that "**::**" is a reflexive, transitive, and antisymmetric[19] relation.

Definition 2 (Compilation of Domain Maps) The mapping Ψ : DM $\rightarrow$ FL of domain maps to F-logic is defined as follows:

1. $\Psi(C) := \{\texttt{C}\colon \texttt{concept}\}$, for all atomic concepts $C \in \mathcal{C}$
2. $\Psi(r) := \{\texttt{r}\colon \texttt{role}\}$, for all roles $r \in \mathcal{R}$
3. $\Psi(C \overset{isa}{\rightarrow} D) := \{\texttt{C} :: \Phi(D)\} \cup \Psi(D)$
4. $\Psi(C \overset{ex:r}{\rightarrow} D) :=$

[19]Since concepts are implemented as FL classes, this avoids terminological cycles.

(a) $\{\mathtt{r}(\mathtt{c}, \_\mathtt{d}), \_\mathtt{d}\colon \Phi(D) \textsf{ if } \mathtt{c}\colon \mathtt{C}, \_\mathtt{d} = \mathtt{skol_D}(\mathtt{c})\} \cup \Psi(D)$

(b) $\{\mathtt{False} \textsf{ if } \mathtt{c}\colon \mathtt{C}, \neg\, (\mathtt{r}(\mathtt{c}, \_\mathtt{d}), \_\mathtt{d}\colon \Phi(D))\} \cup \Psi(D)$

5. $\Psi(C \stackrel{all:r}{\rightarrow} D) :=$

   (a) $\{\mathtt{d}\colon \Phi(D) \textsf{ if } \mathtt{c}\colon \mathtt{C}, \mathtt{r}(\mathtt{c}, \mathtt{d})\} \cup \Psi(D)$

   (b) $\{\mathtt{False} \textsf{ if } \mathtt{c}\colon \mathtt{C}, \mathtt{r}(\mathtt{c}, \mathtt{d}), \neg\, \mathtt{d}\colon \Phi(D)\} \cup \Psi(D)$

6. $\Psi(C \stackrel{r}{\rightarrow} D) := \{\mathtt{r}(\mathtt{c}, \mathtt{d}) \textsf{ if } \mathtt{c}\colon \mathtt{C}, \mathtt{d}\colon \Phi(D)\} \cup \Psi(D)$

7. $\Psi(\mathtt{AND}{\rightarrow_i}\{D_1, \ldots, D_n\}) := \{\mathtt{d}\colon \mathtt{skol_{AND}} \textsf{ if } \mathtt{d}\colon \Phi(D_1), \ldots, \mathtt{d}\colon \Phi(D_n)\} \cup \Psi(D_1) \cup \cdots \cup \Psi(D_n)$

8. $\Psi(\mathtt{OR}{\rightarrow_i}\{D_1, \ldots, D_n\}) := \{\mathtt{d}\colon \mathtt{skol_{OR}} \textsf{ if } \mathtt{d}\colon \Phi(D_1) \vee \ldots \vee \mathtt{d}\colon \Phi(D_n)\} \cup \Psi(D_1) \cup \cdots \cup \Psi(D_n)$

9. $\Psi(C \stackrel{=}{\rightarrow} D) := \{\mathtt{C} :: \Phi(D), \Phi(D) :: \mathtt{C} \textsf{ if } \Phi(D)\} \cup \Psi(D)$ □

Remarks. Here, $\Phi(D)$ is defined similar to $\Psi(D)$, but returns for a compound concept description D, a new auxiliary symbol $\Phi(D)$ representing the compound. For atomic D, we simply have $\Phi(D) = \Psi(D)$. The symbols "$\mathtt{skol}_X$" produce new Skolem function symbols everytime they are used in the translation Ψ: *e.g.*, in (4a), we invent a symbolic representation for the existentially quantified variable "$\_\mathtt{d}$". Note that $\mathtt{c}, \mathtt{d}, \_\mathtt{d}$ are logic *variables*, while $\mathtt{C}, \mathtt{D}, \mathtt{D_i}, \mathtt{False}$ are *constants*.[20] The different variants (a) and (b) in the translations of DMs correspond to different intended uses: in (4a), we *create* an anonymous object for the $\exists$-quantified variable, in (5a), we *type coerce* all $C.r$ objects into instances of D. In contrast, the (b) translations only *check* whether the constraints induced by the DM edges are indeed satisfied, and signal an inconsistency ("$\mathtt{False}$") otherwise.

A.2 Process Maps

Definition 3 (Process Maps) A *process map* $\Pi(S, s_0, s_f, E)$ is a connected, directed graph with nodes S, labeled edges E, and initial and final states $s_0, s_f \in S$. The edges e_π of E are of the form

- $s \stackrel{\{\varphi\}\pi\{\psi\}}{\longrightarrow} s'$ $\qquad (e_\pi)$

where the *process name* π is a first-order atom and φ and ψ are first-order formulas, called the *precondition* and *postcondition* of e_π, respectively.

Given an edge $e = s_a \stackrel{\cdots}{\rightarrow} s_b$ of a process map $\Pi(S, s_0, s_f, E)$, the *elaboration* $\mathsf{elab}(e)$ of e is a process map $\Pi'(S', s_a, s_b, E')$ such that (i) the initial and final states are s_a, s_b, (ii) $S' \cap S = \{s_a, s_b\}$, and (iii) all $e' \in E'$ are linked to e via a common, unique identifier $eid(e', e)$.

[20]This is reversed from the usual convention used in the rest of the paper in order to match our DM notation.

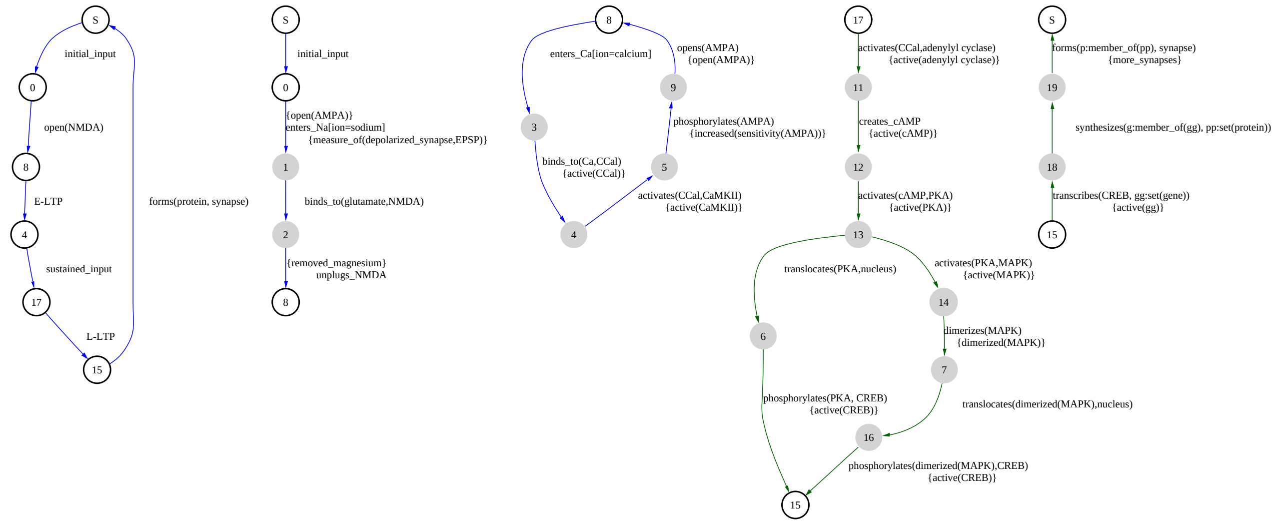


Figure 6.8: Process maps with elaborations and abstractions

A connected subgraph of a PM with distinguished initial and final state is called a *subprocess map* (sub-PM). Given a PM $\Pi(S, s_0, s_f, E)$, the *abstraction* of a sub-PM $\Pi'(S', s_a, s_b, E')$ of Π, denoted $\mathsf{abstract}(\Pi')$, is a new edge $e_{\pi'} = s_a \overset{...}{\to} s_b$, where (i) $e_{\pi'} \notin E$, and (ii) all $e' \in E'$ are linked to $e_{\pi'}$ via a common, unique identifier $aid(e', e_{\pi'})$. $\square$

Marking edges with elaboration and abstraction identifiers, guarantees one-to-one mappings between an edge and its elaboration, and similarly, between a subprocess map and its abstraction. In this way, a data provider can "double-click" on an edge e_π and elaborate the processes into a PM Π in order to provide more precise links to her data. Conversely, she may "collapse" a subprocess map Π into a single edge e_π, if her data does not provide information at the detailed level of Π, hence is more adequately hooked to the overall process e_π.

Process Maps as Logic Rules

Similar to DMs, we can translate PMs into a logic representation $\Psi(\text{PM})$. The difference is that for DMs our formalization in description logic or F-logic yields a first-order logic semantics, whose unique minimal model $\mathcal{M}(\text{DM})$, interprets concepts and roles as unary and binary predicates over a set of individuals. The model $\mathcal{M}$ implies that data objects, which are linked as concept instances to a DM, have the properties defined by the domain map (*e.g.*, the neurons in the images linked to `MyNeuron` in Example 4 project to `Globus_Pallidus_External`). In constrast, the logic representation of a PM specifies only some process properties via pre- and postconditions in

PM, and PM's graph structure. We omit the details of the semantics, due to lack of space. The basic idea is that the graph structure of PMs (with its embedded hierarchy of elaborations and abstractions) is formalized via a nested Kripke structure in which the nodes of PM (states) have associated first-order models, and in which labeled process edges specify a *temporal accessibility relation* between states.[21] In particular, a process elaboration of an edge e_π adds to the initial semantics PM_0, and the semantics of the pre- and postconditions of the concrete occurrence of e_π in PM, an *elaboration semantics*, *i.e.*, a sequence of intermediate states with first-order constraints along the paths of the elaboration.

[21]see, *e.g.*, [LLM98, Sec. 6] for a formalization of hierarchical processes using nested Kripke structures.