Knowledge-Based Integration of Neuroscience Data Sources

Amarnath Gupta[1] Bertram Ludäscher[1] Maryann E. Martone[2,*]

[1] San Diego Supercomputer Center, UCSD {gupta,ludaesch}@sdsc.edu
[2] Department of Neurosciences, UCSD mmartone@ucsd.edu

Abstract

The need for information integration is paramount in many biological disciplines, because of the large heterogeneity in both the types of data involved and in the diversity of approaches (physiological, anatomical, biochemical, etc.) taken by biologists to study the same or correlated phenomena. However, the very heterogeneity makes the task of information integration very difficult since two approaches studying different aspects of the same phenomena may not even share common attributes in their schema description. This paper develops a wrapper-mediator architecture which extends the conventional data- and view-oriented information mediation approach by incorporating additional knowledge-modules that bridge the gap between the heterogeneous data sources. The semantic integration of the disparate local data sources employs F-logic *as a data and knowledge representation and reasoning formalism. We show that the rich object-oriented modeling features of F-logic together with its declarative rule language and the uniform treatment of data and metadata (schema information) make it an ideal candidate for complex integration tasks. We substantiate this claim by elaborating on our integration architecture and illustrating the approach using real world examples from the neuroscience domain. The complete integration framework is currently under development; a first prototype establishing the viability of the approach is operational.*

1 Introduction

A grand goal in many disciplines of biological research is to understand the workings of a biological organ like the brain and how the interplay of different structural, chemical and electrical signals in the biological tissues gives rise to natural and disease processes [KH96]. To achieve such a goal, however, it is essential to develop an integrated understanding of very different, but conceptually correlated studies and data produced from diverse biological subdisciplines. Most importantly:

- Biologists assess different animal models to study different aspects of the same biological function. Thus, for a given research problem, they may wish to integrate information about the cytoarchitecture of sensory cortex from the somatosensory cortex of the rat, the brain areas involved in vision from the primate, the physiology of receptive fields from the cat, the distribution of key proteins involved from the rat, and the molecular underpinnings of synaptic plasticity from the mouse.

- Biologists study the same biological system from multiple perspectives. For example, in the study of calcium regulation, researcher A may take a physiological approach, using patch electrodes to study calcium currents; researcher B may take an anatomical approach, mapping the distribution of different isoforms of calcium regulatory proteins and the organelles that express them; a biochemist C may study signal transduction cascades and levels of protein activity using Western blots and assay systems, a pharmacologist D may use a panel of channel blockers, agonists or antagonists to study the response in single cells or the whole animal to alterations in calcium regulation.

The goal of this paper is to present an architecture to integrate different studies and analyses conducted by biologists performing different experiments, such that the integrated body of information can be queried and navigated across. Once such information is integrated, the practicing biologist can use the system to discover biologically significant correlations and use the discovery to plan future work in the context of available data. The integration challenge is that source data cannot be joined using simple term-matching or comparison operators. Even more sophisticated approaches which use ontologies to enumerate joinable terms [Kas96] are often not sufficient. Instead a join should be performed based on whether the objects satisfy

* Supported by NIH grants RR04050 and DC03192.

some application-specific condition. For complex integration scenarios as our neuroscience application, a more expressive formalism is necessary to specify these "semantic join conditions". In particular, the formalism should have inferencing mechanisms to reason over domain knowledge if necessary.

The contributions of the paper are as follows: We develop an architecture called KIND (***K**nowledge-based **I**ntegration of **N**euroscience **D**ata*) that extends the conventional wrapper-mediator architecture with one or more domain *knowledge bases* that provide the "semantic glue" between sources through facts and rules from the application domain. Thus our mediator enhances view-based information integration with deductive capabilities. Data manipulation and restructuring operations for integration can be performed not only on the base data from the sources but also on intensional data derivable from the knowledge bases. To this end, we employ the deductive object-oriented language F-logic and demonstrate that it can handle the given complex integration problems. An implementation of the system is underway; first experiments with a preliminary (centralized, non-distributed) prototype have proven the viability of the approach [KIN00].

We like to emphasize that the integration problem we address is different from the problems addressed in *database federation* (or *multidatabases*) [SL90, BE96, PS98]. There a huge body of work has dealt with issues like schema integration, resolving conflicts and mismatches (structural, extensional, naming, etc.), global query processing in the presence of local autonomy etc. Those heterogeneities are between different representations of essentially the *same* (or very similar) real world entities. In contrast, we deal with sources containing *inherently different* (but related through "expert knowledge") information so these conflict resolution techniques are not applicable.[1]

The rest of the paper is organized as follows. In Section 2 we provide a brief introduction to F-logic to clarify the notation and concepts used subsequently. Section 3 presents a motivating example that illustrates the nature of the information integration task for the given problem domain. In Section 4 we present our architecture and explain the role of F-logic in the representation of schema, knowledge and in the inference mechanism. Section 5 illustrates a particular instance of our architecture, i.e., presents several elements of INSM, the ***I**ntegrated **N**euro**S**cience **M**odel*. In Section 6 we show in more detail how integrated views are defined and queried in INSM using semantic information. Finally, Section 7 contains a discussion including a comparison with related approaches and an outlook on future improvements and optimizations of the architecture.

[1] This does not preclude the possibility that our integration scenarios *also* involve such problems and hence will benefit from this previous work.

2 F-Logic in a Nutshell

Since our integration approach is based on F-logic, we briefly introduce the syntax and basic concepts of F-logic; see [KLW95] for a full coverage of all features, in particular details of the F-logic semantics (or [LHL+98] for a gentle introduction with a focus on the management of semistructured data and querying the Web).

While there are other formalisms that could possibly be used, we chose F-logic for several reasons: F-logic is a declarative language with rich modeling capabilities (class hierarchy, complex objects, inheritance, etc.) and a powerful rule language. It has its roots in AI (frame-based knowledge representation) and deductive object-oriented databases. Apart from "pure" database modeling and querying, it has been applied in several related (but different) areas, including schema transformation [CRD94], information integration [GBMS99], querying the Web [LHL+98, MHLL99, DFKR99], knowledge representation/reasoning with ontologies [DEFS99], and management of semistructured data [LHL+98]. F-logic extends Datalog and first-order logic (including Skolem functions). In particular, well-known transformations can be used to map arbitrary first-order constraints to equivalent stratified Datalog (and thus F-logic) rules. Finally, F-logic query evaluation engines such as FLORA [FLOa], FLORID [FLOb], and SILRI [DEFS99] are readily available (and continue to be improved).

F-logic Syntax and Object Model

- *Symbols*: The F-logic alphabet comprises sets $\mathcal{F}$, $\mathcal{P}$, and $\mathcal{V}$ of *object constructors* (i.e., function symbols), *predicate symbols* (including $\doteq$), and *variables*, respectively. Variables are denoted by capitalized symbols (X, Name, ...), whereas constants and function symbols (0-ary and n-ary object constructors) are denoted in lowercase (cerebellum, foo(bar,baz), ...) unless quoted ('Cerebellum'). An expression is *ground* if it involves no variables. In addition to the usual first-order symbols, there are special symbols[2]:], [, }, {, $\rightarrow$, $\twoheadrightarrow$, $\Rightarrow$, $\Rightarrow\!\!\!\!\twoheadrightarrow$, :, ::.

- *Id-Terms/Object-Ids (Oids)*:

  (0) First-order terms over $\mathcal{F}$ and $\mathcal{V}$ are called *id-terms*, and are used to name objects, methods, and classes. Ground id-terms correspond to *logical object identifiers* (*oid*s). In particular, *constants* and *strings* ("cerebellum") are oids; the latter are conceived as character *lists*, i.e., nested ground terms.

[2] We do not deal with inheritable methods here, so we omit the corresponding symbols; cf. [KLW95].

- *Atoms*: Let O, M, R_i, X_i, C, D, T be id-terms. In addition to the usual first-order atoms like $p(X_1, \ldots, X_n)$, there are the following basic types of atoms:

  (1) $O[M \rightarrow R_0]$ (*single-valued meth. app.*)
  (2) $O[M \twoheadrightarrow \{R_1, .., R_n\}]$ (*multi-valued meth. app.*)
  (3) $C[M \Rightarrow T]$ (*single-valued class signature*)
  (4) $C[M \Rightarrow\!\!\!\!\twoheadrightarrow T]$. (*multi-valued class signature*)

  (1) and (2) are *data atoms* and specify (at the instance level) that the application of *method* M to the object with oid O yields the result object with oid R_i. In (1), M is *single-valued* (or *scalar*), i.e., there is at most one R_0 such that $O[M \rightarrow R_0]$ holds. In contrast, in (2), M is *multi-valued*, so there may be several result objects R_i. For $n = 1$ the braces may be omitted.
  (3) and (4) denote *signature atoms* and declare that the (single/multi-valued) method M applied to objects of *class* C yields instances of *type* (i.e., class) T.

  The organization of objects into classes is specified by *isa-atoms*:[3]

  (5) $O : C$ (O *is an instance of class* C)
  (6) $C :: D$. (C *is a subclass of* D)

- *Path Expressions*: F-logic supports path expressions to simplify object navigation along single-valued and multi-valued method applications and to avoid explicit join conditions. The following *path expressions* are allowed in place of id-terms:

  (7) $O.M$ (*single-valued path expression*)
  (8) $O..M$ (*multi-valued path expression*)

  The path expression (7) is *single-valued* and refers to the unique object R_0 for which $O[M \rightarrow R_0]$ holds, whereas (8) is *multi-valued* and refers to each R_i for which $O[M \twoheadrightarrow \{R_i\}]$ holds. O and M may be id-terms or path expressions. Although not part of the core syntax, *generalized path expressions*[4] can be defined by means of rules [LHL+98].

- *Parameters*: Methods may be *parameterized*, so

  $$M@(X_1, \ldots, X_k)$$

  is allowed in (1–4) and (7–8).

  Example: o_{rat}[name@(scientific)→"Rattus rattus"].

- *Rules*: A *rule* is of the form

  Head **IF** *Body*

  where *Head* and *Body* are conjunctions of F-logic atoms (read *"if a ground instance satisfies Body then also the Head"*); a *program* is a set of rules.

F-molecules are a concise notation for several atoms specifying properties of the same object: for example, instead of o_{rat} : taxon $\wedge$ o_{rat}[name@(common)→"rat"] $\wedge$ o_{rat}[order→"Rodentia"] we can simply write

o_{rat} : taxon[name@(common)→"rat"; order→"Rodentia"]

In F-logic rules "," is shorthand for "$\wedge$".

Object Model

An *F-logic database (instance)* is a set of ground F-logic atoms. The basic relations among objects ($\rightarrow$, $\twoheadrightarrow$, $\Rightarrow$, $\Rightarrow\!\!\!\!\twoheadrightarrow$) in this model can be represented as a labeled graph where nodes are oids and where edges are labeled with the corresponding arrow and the method name. From base facts additional facts can be derived by means of rules.

Example 1 (Fragment of ANATOM) The following is a fragment of ground F-logic atoms and molecules that make up the anatomical knowledge base ANATOM (Section 4.1):

```
nervous_system[has@(struct)→→{cns,pns}].
cns[has@(struct)→→{brain, spinal_cord}].
brain[has@(struct)→→{telencephalon, diencephalon,
      mesencephalon,rhombencephalon}].
  ...
cerebellar_cortex[ has@(func)→→{hemisphere, vermis,
      flocculus, parafloccular_lobes}]

eukaryotic_cell :: cell.
brain_cell :: eukaryotic_cell.
neuron :: brain_cell.
glia :: brain_cell. projection_neuron :: neuron.
interneuron :: neuron.
purkinje_cell :: projection_neuron.
  ...
schwan_cell :: glia.
```

The first two groups of facts describe *has-a* relationships which are either structural or functional, the third group specifies the *is-a* hierarchy of brain cells using F-logic's subclass connective " **::** ".

Based on such a fact base, rules are used to specify intensional knowledge (which is derivable on-demand at runtime). For example, the transitive closure of *all* has-a relationships (structural, functional, ...) can be expressed by a *single* recursive rule

```
X[has@(P)→→{Y}] IF X..has@(P)..has@(P) = Y.
```

thereby illustrating the expressive power of rules using parameterized methods, path expressions, and recursion. □

[3] In KR parlance, for o_{53} : medium_spiny_neuron :: neuron :: cell we say "a medium_spiny_neuron isa neuron isa cell" and "o_{53} is an instance of medium_spiny_neuron". In F-logic parlance, we say "subclass" instead of "isa".

[4] aka *regular path expressions* and similar to XML's XPath expressions [XPa99]

3 Integration Across Multiple Worlds: Motivating Example

Consider three research groups: Group A studies neuroanatomy of rodents, group B studies calcium regulatory proteins in vertebrates and group C studies neurotransmission phenomena in mammals.

Group A ("Neuroanatomy"). Let us assume that group A has a database of studies, where a study consists of a number of experiments on a specific research problem. Each experiment records a number of experimental parameters in the form of (*name*,*value*) pairs, and produces a number of images. We focus on a specific image class called *protein labeling images*. For each protein label image the anatomical parameters (i.e., which anatomical region the image represents) and the protein used are recorded. Each image is segmented into a number of segments, based upon the amount of protein staining. The image is also represented as a collection of named anatomical structures visible in the image. Each anatomical structure is modeled as a collection of segments, such that aggregate features like the distribution of stain within an anatomical structure may be computed. Very often a single biological study involves a number of experiments conducted at *different granularity levels* in the animal. For example, experimenters may try to localize a protein in a tissue, a cell, specific cellular compartments and in intracellular substructures. In this case, the anatomical parameters of an image at any level are *semantically* related to those of an image at the next coarser level although this relation may not be directly visible from the schemas. We will explain how this is modeled in Section 6. In some experiments, specific anatomical structures from a stack of confocal images or a series of electron micrographs are reconstructed into volumetric objects modeled as geometric entities, and specific 3D properties such as the surface to volume ratio are measured. The volumetric information is stored in a separate database DENDREC. The 3D anatomical models are related to the images from which the reconstruction was made. In our example, DENDREC contains the reconstruction of spiny dendrites in the rat neostriatum.

Group B ("Calcium-Binding Proteins"). Next, let us assume group B to have a database of calcium-binding proteins[5] where each protein is identified by its reference number in the PDB[6] and/or the reference number in the SWISS-PROT[7] database. Otherwise it is identified by an internal identifier. A protein has a molecular weight, an amino acid sequence, the number of amino acids and is grouped with a number of other proteins that belong to the same family. Its isoforms, mutants and the species in which the mutants are found are also recorded. Every protein subfamily and mutant form is also given a unique identifier. For every protein the researchers also record the interaction of the protein with elements and ions, the evidence of signal transduction pathways it participates in and the disease processes it contributes to. The database organizes the signal transduction and disease information by the species where the evidence has been found. Also grouped by species, the researchers record the tissue and cell-level localization they have found in their experiments. However this group does not conduct any experiments at a subcellular level. Although the system does not store the genetic code of the proteins they study, they maintain the reference identifier for the protein form in the GENBANK[8].

Group C ("Neurotransmission"). Finally, group C stores information about neurotransmission including neurotransmitter substances, neurotransmitter receptors and voltage-gated conductances in a database NTRANS[9]. In this system every neuron is modeled to be composed of a canonical set of non-overlapping compartments. For every compartment of each neuron studied, the experimenters record the input receptors and their description, the intrinsic ionic currents along with their description, and the output transmitters. The description contains a textual account of the function of the receptor or transmitter, the brain region where they are active. Each type of current is characterized by the ions that generate them, their electrical properties, and their firing characteristics. Receptors and transmitters are also organized into families, representable by a tree structure.

When these systems are integrated, a biologist would like to make queries such as:

- *Find the cerebellar distribution of rat proteins with more than 90% amino acid homology with the human NCS-1 protein. Compare the distribution of this protein or its homologs in other rodents.*
- *Are any calcium-binding proteins found only in the thin dendritic spines of the rat neostriatum and not in the stubby spines? Do these proteins always co-localize?*
- *Is there any experiment performed on other mammals on the proteins involved in signal transduction in the visual systems of primates? How similar are these proteins?*

A major challenge in today's bioinformatics is to find ways to correlate, combine and unify information from mul-

[5] `http://structbio.vanderbilt.edu/cabp_database/`
[6] Protein Data Bank, `http://www.rcsb.org/pdb/`
[7] `http://www.expasy.ch/`
[8] `http://www2.ncbi.nlm.nih.gov/genbank/query-form.html`
[9] e.g. `http://ycmi.med.yale.edu/senselab/neurondb/`

tiple data sources such as described above. But even with many online data sources and information retrieval tools, biologists have little or incomplete technological framework to make this unification possible across disciplines, scales of observation and diversity of viewpoints. As a result, they perform the "integration" task manually, by physically assembling data from multiple sources and putting them together by individual effort. Hence they are seldom allowed the luxury to make "inter-database" queries although the capability to perform such queries is fundamental to the task of the broad-based knowledge unification that they seek. An ostensible source of difficulty arises from the semantic incompatibilities both within and between the data sources. For example, consider that group A has a number of experiments on the protein distribution of basket cells and their neighboring Purkinje cells without stating that they both belong to the rat cerebellum. We need to model and use this additional piece of knowledge in order to answer the first query. Similarly, the fact that proteins are related because they share amino acid homologies is never recorded because it is "common knowledge" to the domain. However, unless this information is explicitly available from a supplementary source, the query cannot be answered.

In addition to the need for having additional knowledge integration of biological information also have the following issues:

- The information representation at the mediator should be flexible enough to accommodate a wide degree of heterogeneity at the data sources, and at the same time, represent the class-structure evident from the taxonomic character of the data. To accommodate this, we use an object-oriented formalism, but unlike the collection-based model used by Bio-Kleisi [DOTW97], we use F-logic that is well-equipped to represent object-orientation, flexible enough to represent semistructured data, and has the machinery to perform inferences and recursive computation such as path expressions and transitive closure.

- The computation of numeric aggregates and numeric features describing the content of 2D and 3D images and reconstructed volumes is an essential component of the data to be integrated. Equally important is the need to represent complex semantic rules to model the associations between numeric features computed from multiple image and volume instances.

- Queries involving graph operations such as graph intersection and computation of the spanning tree are important in discovering the associations between data coming from different sources that are initially unconnected. Meta-level reasoning with schema and attributes are an important component in creating these associations.

4 The Integration Framework

Most current approaches to integration of information from heterogeneous sources are based on the well-known *mediator architecture* [Wie92]: The problem of heterogeneous data models of sources is solved by translating the data into a common format using *wrappers*. The *semistructured data model* (essentially *labeled directed graphs*, [Abi97]) in general and XML [XML98b] in particular have been shown to be suitable target data models.[10] Once the data can be accessed in a uniform way, a *mediator* is used to integrate between the different local views and schema elements, based on the specification of an integrated view. The definition of such an integrated view can often be a highly complex task and requires dealing with all of the well-known integration problems from information integration in databases like *structural*, *semantic*, and *descriptive* conflicts[11] (e.g., flat vs. nested relational vs. object-oriented modeling, homonyms, synonyms, ... [SL90, PS98]). Thus, for complex integration tasks, a powerful declarative specification language is required, e.g., for querying and restructuring local schemas, mapping data between models and schemas, integrity checking, and knowledge inference.

4.1 The KIND Architecture

For our neuroscience application domain, we have developed an elaboration of the mediator architecture called KIND (Figure 1): The main data sources are scientific STUDIES of various types which themselves can refer to further heterogeneous data sources like PROLAB (image databases of protein labelings) and DENDREC (volumetric reconstructions of dendrites). Other KIND sources are CAPROT (calcium-binding protein databases) and NTRANS (neuro-transmission database). Apart from these sources of observational data, there are also sources with general domain data and knowledge like ANATOM (anatomical knowledge base) and TAXON (animal taxonomy database).

Unlike other mediator approaches that solely use the semistructured model throughout the integration, we additionally incorporate a rich object-oriented knowledge representation formalism, i.e., F-logic [KLW95] into the architecture. This enables a better modularization and more adequate modeling of complex application domains like biology and neuroscience.[12] In our biological integration do-

[10] Raw data is often given in an unstructured or semistructured format (HTML, spreadsheets, formatted text, etc.). Also, structured data like relational, hierarchical, and object-oriented data can be easily mapped into the semistructured model.

[11] Since wrappers for legacy sources are often "thin" and just hide *syntactic* differences and different access methods etc. from the mediator, the latter has to deal with all of these integration problems.

[12] Indeed, as will be shown below, simple database view definition mechanisms (e.g., joins) are not sufficient to capture such rich domain semantics.

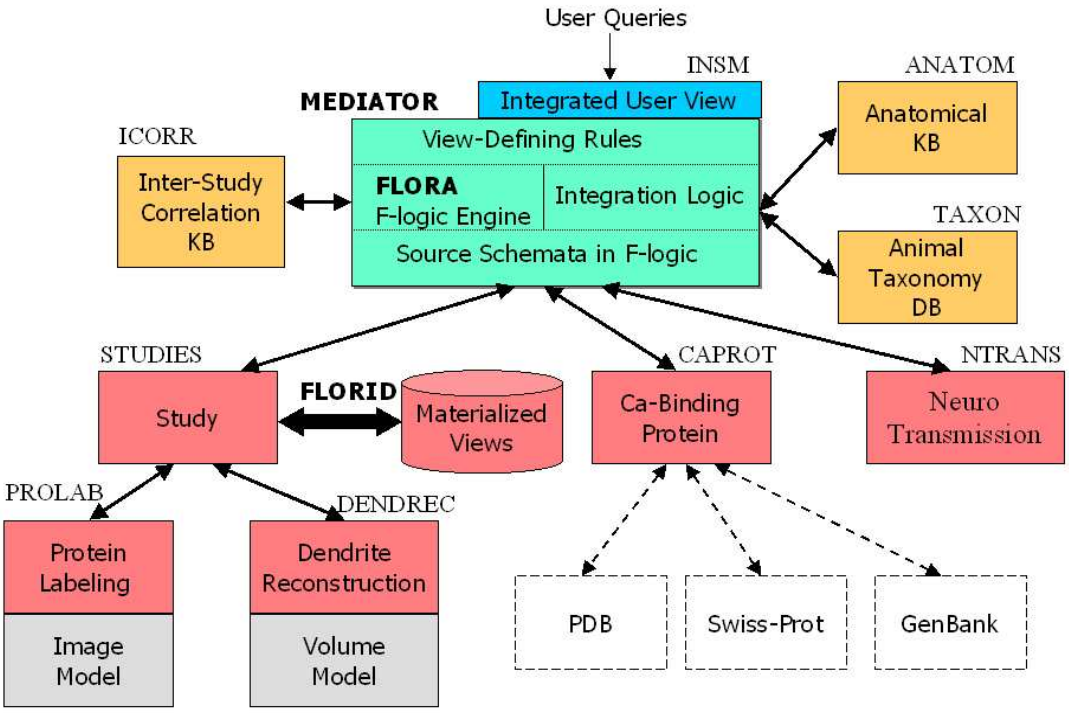

Figure 1. The KIND Mediator Architecture

main, for example, links between otherwise unrelated data are established using expert knowledge (like anatomical, taxonomic, or partonomic relationships [HF96]) which is represented using F-logic rules. In some cases, ontologies exist for modeling specific aspects or parts of a domain, thereby providing a unique semantics for that part. While an ontology captures the semantics of some domain, the problem remains to mediate across different ontologies for providing the user with an integrated view [WJ98]. Again a powerful integration language like F-logic is needed for mediating between the ontologies.

4.2 KIND Modules

The *source modules* of the KIND architecture, i.e., data and knowledge sources, have an associated XML DTD (*Document Type Definition*) describing the structure of the exported data after wrapping. Here, we speak of a *data source* (or data module), when the modeled information has mainly observational character like data collected during an experiment, and of a *knowledge source* when we model information about the application domain ("general" or "expert knowledge"), usually in rule form. Note that for some sources wrapping may be done once and off-line. However, often this translation has to be done online (or *on-demand*), i.e., the wrapper has to translate incoming XML queries to native queries against the actual source data. In general, the *query capabilities* of the underlying source are limited in which case the wrapper can support only specific XML queries.[13]

Syntactic integration with some minimal consistency at the source level is achieved by enforcing that a source module M exports *valid* XML, i.e., which conforms to the associated $\mathrm{DTD}(M)$. Clearly, additional integrity constraints dealing with both structural aspects and application domain constraints should be modeled in order to guarantee consistency at a higher, conceptual level [Tür99, CGL+98].

Exported Object and Class Structure

At the level of a source module M, we incorporate F-logic by providing a *class signature* $\Sigma(M)$ of exported

[13] Note that the wrapping step has been omitted from Figure 1.

classes and their objects' structure, which constitutes a semantically much richer conceptual-level specification of the source than just $\mathrm{DTD}(M)$ and thereby facilitates the integration of M at a conceptual level at the mediator. In particular, $\Sigma(M)$ specifies

- the source's *class hierarchy*,
- whether attributes (i.e., F-logic methods) are *single-valued* or *multi-valued*,
- whether and how they are *parameterized*, and
- whether they are *inheritable* or not.

Formally, the object-oriented class structure of M is given by a mapping Φ_M: $\mathrm{DTD}(M) \rightarrow \Sigma(M)$. Technically, Φ_M is straightforward to implement as it amounts to a simple syntactic transformation from XML elements to F-logic expressions (e.g., using an XML parser whose output is "pretty-printed" to F-logic, or using the XML stylesheet/transformation language XSL(T)). The difficulty consists in choosing the most appropriate "semantically adequate" representation in F-logic of the underlying, XML-encoded, object model. Consider, for example, a *generic* mapping Φ_{gen} which maps *arbitrary* XML documents (i.e., irrespective of an object model of the encoded information) to F-logic representations: Since an XML document is a semistructured database (more precisely, a labeled ordered tree) it can be represented in F-logic, for example, over the signature

```
xml_node[
        element_type⇒string;
        attribute@(string)⇒string;
        child@(integer)⇒xml_node ]  .
```

While Φ_{gen} faithfully represents any given XML document, the application domain structure is *not* visible at the schema level and has to be extracted from the data. Thus, whenever possible, it is preferable to model a source by first specifying its application domain structure in F-logic, i.e., designing $\Sigma(M)$. Then a syntactic representation of $\Sigma(M)$ using an XML DTD is straightforward, and we can trivially go back from that DTD to $\Sigma(M)$. Hence we get Φ_M essentially "for free".

In case a source module does not have an F-logic signature $\Sigma(M)$, for example, because M is a new source module being added to the system and $\mathrm{DTD}(M)$ is unknown, or Φ_M has not yet been established, then Φ_{gen} can still be useful as a first means to bring the new data into the system. Indeed, F-logic is also suitable as a language for managing semistructured data, i.e., extracting data using generalized path expressions, discovering schema etc. [LHL+98, MHLL99].

Exported Integrity Constraints

In addition to $\Sigma(M)$, a set of application specific *integrity constraints* $\mathrm{IC}(M)$ can be provided. These are F-logic rules that create "alerter objects", i.e., instances of class alert whenever an inconsistency (at the class or object level) is derived. An alerter object indicates the type of inconsistency encountered and some hints on which objects and classes were involved in the inconsistency (see Section 5) which greatly simplifies debugging the data. In particular, this allows to differentiate between *local inconsistencies* (i.e., within a module M) and *global inconsistencies* [Tür99].

Derived Knowledge

Finally, some modules M also export *intensional knowledge* in the form of a set of F-logic rules $\mathrm{IDB}(M)$.

Example 2 (ANATOM Fragment Cont'd) The anatomical knowledge base ANATOM (cf. Example 1, Figure 1) includes the following rules

```
purkinje_cell[located_in→↠{purkinje_cell_layer}].
basket_cell[located_in→↠{cerebellar_cortex}].
X[located_in →↠C] IF
    X:nucleus[located_in →↠{N:neuron}],
    N[compartments →↠{C:cell_body}].
X[located_in→↠{X}] IF X:neuro_anatomic_entity.
Y[located_in→↠{X}] IF X[has@(P)→↠{Y}].
X[located_in→↠{Y}] IF X..located_in..located_in = Y.
```

defining the located_in relation from base facts, specific anatomical knowledge rules, and generic rules for defining reflexive and transitive closure. □

4.3 The KIND Mediator

As explained above, the structure and semantics of a source module M is specified using an XML $\mathrm{DTD}(M)$ (mainly for inputing the wrapped raw data), a class signature $\Sigma(M)$, the correspondence mapping Φ_M between them, integrity constraints $\mathrm{IC}(M)$ and, in the case of derived knowledge, $\mathrm{IDB}(M)$. The KIND mediator module itself exports an integrated F-logic view INSM (***I**ntegrated **N**euro**S**cience **M**odel*) to the user, which is defined based on the imported source modules (STUDIES, TAXON, ...), the facts and rules from the imported knowledge bases (ANATOM, ICORR), and the actual view-defining integration rules (Figure 1). The mediator imports signatures of a module M using declarations like

```
:- import study[
        id⇒string; project⇒string;
        experiments⇒⇒experiment; ...]
from 'STUDIES'.
```

In this way, a subgoal of the form S:study[id→I; project→P; ...] induces a query against the STUDIES source. Note that oids of objects from different source modules M_1 and M_2 are guaranteed to be distinct. The declarative way to achieve this is by qualifying each oid with the URI of the module from which it was imported (in the implementation we can just use disjoint sets of integers as oids). The only oids which can be shared across modules are those of string objects and constants occurring in the import declaration. For example, consider two sources SIMPLE and DETAILED of animal data. We can simultaneously import from both modules as follows:

```
:- import animal[name⇒string] from 'SIMPLE'.
:- import animal[common_name⇒string; species⇒string;
        genus⇒string; ...] from 'DETAILED'.
```

Constants appearing in the import declarations (animal, species, . . .) and string-valued objects like "Rodentia" are distinguished and thus shared external object names. Therefore a subgoal of the form X:animal[Attr→Val] will yield both, instance from SIMPLE and from DETAILED, together with their attribute/value pairs. As part of the integration process, we may have to distinguish between instances of animal from SIMPLE and those from DETAILED (note that the way internal oids are differentiated may not be visible to the rule programmer). This is accomplished by qualifying names with the module they were important from: e.g., X:("DETAILED".animal)[M→R] will only range over objects from DETAILED. Logically, this corresponds to defining for each module M, the methods M.N for all distinguished names imported from M:

```
M[N→M.N] IF N:distinguished_name[imported_from→M] .
```

When importing data from M, a distinguished (exported) class name C is prefixed with M and all instances of C in M are made instances of M.C.

F-Logic Query Evaluation

The current prototypical implementation of the KIND system [KIN00] uses a central mediator component with FLORA [LYK99, FLOa], an F-logic to XSB-Prolog compiler, as the evaluation engine. Due to its built-in top-down strategy[14], FLORA derives facts in a *demand-driven* way somewhat similar to the VXD architecture of MIX*m* [LPV00]. At the current implementation stage, sources have no independent query evaluation mechanism but simply export all data and rules to the central mediator. However, the design of the architecture allows for source modules to have their own evaluation engine in which case source data is imported only as needed for answering queries.

[14] ... with *tabling* to ensure termination in the function-free case – this is not guaranteed with standard Prolog.

In contrast, the FLORID system [FLOb] is an implementation of F-logic which employs a bottom-up and thus a model *materialization* strategy. It has been shown that FLORID is well-suited for management of semistructured data [LHL+98] and as a unified framework for wrapping and mediating Web data [MHLL99]. Therefore, we plan to incorporate the FLORID engine into the KIND architecture for modules and views where materialization is advantageous.

5 Elements of the Integrated Neuroscience Model

In this section we illustrate of the concepts described in the previous section by examples.

XML-DTD and F-logic Representations

Each source module M has an associated XML DTD. The XML data may result from wrapping of the raw data, or the source may natively support XML. The mediator can either import the XML DTD as is (using the generic mapping Φ_M to F-logic; cf. Section 4.1), in which case any application specific structure not visible from the DTD has to be "recovered" at the mediator, or the mediator can import the semantically richer F-logic signature.

The following XML DTD is used by the STUDIES database:[15]

```
<!ELEMENT Studies       (Study)*>
<!ELEMENT Study         (study_id, project_name,
                         project_description, animal,
                         experiments, experimenters)>
<!ELEMENT animal        (subject_id, scientific_name,
                         strain, age)>
<!ELEMENT experiments   (experiment)*>
<!ELEMENT experiment    (description, instrument,
                         parameters)>
<!ELEMENT instrument    (type, name)>
<!ELEMENT parameters    (parameter)*>
<!ELEMENT parameter     (name, value)>
<!ELEMENT experimenters (experimenter)*>
<!ELEMENT experimenter  (name, affiliation)>
```

One of many ways to model this in F-logic is as follows:

```
studyDB[studies⇒⇒study].
study[id⇒string; project_name⇒string; description⇒string;
      animal⇒animal; experiments⇒⇒experiment;
      experimenters⇒⇒string].
animal[subject_id⇒string; scientific_name⇒string;
      strain⇒string; age⇒string].
experiment[description⇒string; instrument⇒instrument;
      parameters⇒⇒exp_parameter].
instrument[type⇒string; name⇒string].
exp_parameter[name⇒string; value⇒string].
```

In general, the F-logic signature $\Sigma(M)$ can carry much more semantics from the application domain (due to

[15] We omit the PCDATA elements from the schema.

class hierarchies, parameterized methods, single- vs. multi-valued, etc.), in particular when accompanied by integrity constraints $\mathrm{IC}(M)$ and derived knowledge $\mathrm{IDB}(M)$.

Creation of Mediated Classes

At the mediator level, the F-logic schema of the source is modified to relate it to the knowledge sources. For example, the type of scientific_name, which consists of genus and species names and may optionally include a subspecies name, is modified from a string to a taxon reference, where a taxon is an element from the taxonomic database. The modification is made by first creating a new class called animal at the mediator as the union of the classes called animal at the sources. Thus, at the mediator

```
animal[M⇒R] IF S:source, S.animal[M⇒R].
```

Then a new method is added to this union class to link it to the taxonomic database:

```
animal[taxon⇒'TAXON'.taxon].
```

Finally, the association between the scientic name in the PROLAB database and the taxonomic database is created:[16]

```
X[taxon→T] IF
        _:'PROLAB'.animal[scientific_name→N],
        words(N,[W1,W2|_]),
        T:'TAXON'.taxon[genus→W1; species →W2].
```

The built-in predicate words, when given a string as first argument, returns the list of words of that string. As will be used later, this predicate can also create a string of words which are separated by a whitespace. Such somewhat "procedural" predicates like word can easily be defined at the mediator, since the whole XSB-Prolog machinery is accessible from the FLORA F-logic engine.

Geometric Modeling

In modeling reconstructed volumes of dendritic spines[17], we first create a number of solid-geometric primitives called shape3D like cylinders, spheres and hyperboloids, used by solid modeling software.

```
shape3D[volume⇒scalar; area⇒scalar]
cylinder::shape3D[radius⇒scalar; length⇒scalar].
```

The model of dendritic spines is composed of these primitives:

```
spine::shape3D[view_files⇒⇒url].
mushroom_spine::spine[
        head⇒head; taper⇒taper; neck⇒neck].
head[shape⇒sphere]. taper[shape⇒hyperboloid].
neck[shape⇒cylinder].
```

[16] Each occurrence of the *anonymous variable* "_" corresponds to a fresh variable.

[17] Dendritic spines are specialized protrusions on neurons that receive the bulk of synaptic input.

The view_files attribute yields a list of urls that represent the images of different projections of the 3D volume. A parametric attribute of F-logic is used to model spines protruding out from the shaft of a dendrite at a coordinate (x,y,z):

```
shaft[connected_spine@(x,y,z)⇒spine; num_spines⇒integer].
```

Rules for Classification and Integrity

The dendritic spines are classified into thin, stubby and mushroom classes using F-logic rules:

```
S:mushroom_spine IF S:spine[head→_; neck→_; taper→_].
S:stubby_spine IF S:spine[head→_; undef↠{neck, taper}].
S:thin_spine IF S:spine[neck→_; undef↠{head, taper}].
```

The method undef applied to an object O yields those methods M that are declared for class C but which are not defined for O. This is a simple example for reasoning about schema and is specified in F-logic as follows:

```
O[undef↠{M} IF O:C[M⇒⇒_], not O[M↠_].
```

The dendritic reconstruction data source also has integrity constraint rules, defined as a special class called **alert**. For example, the constraint that a dendritic spine cannot have only a taper (but no head or neck) is modeled as:

```
ic1(S):alert[type→"singleton taper"; object→S] IF
        S:spine[taper→_; undef↠{head,neck}].
```

Complex Relationships

As mentioned earlier, experimental biological information often have complex semantic relationships. For example, two experiments in a single study in source A may be related in the following way. In the first experiment, the experimenters perform a protein labeling on the entire brain and record the result as a segmentable image. In the second experiment, they would like to investigate the protein labeling pattern of the heavily stained portions of the brain region called *cerebellum*. So they extract that part of the cerebellum (from an identical specimen) which showed heavy staining in the previous experiment, and produce a finer resolution image to identify the actual cells that took the heavy stain (see Figure 2). Although these two images are related, the relationship cannot be modeled just by linking the second image to a segment of the first. In reality, the second image is related to *any segment* in the first specimen that satisfies the condition of being "heavily stained" and in the cerebellum. We model this by using *named predicate*s. Consider a fragment on the schema of the class image.

```
image[anatomical_structures⇒⇒anatomical_structure].
anatomical_structure[name⇒string; segments⇒⇒segment].
segment[description⇒string; features⇒⇒feature].
```

We consider the simple case where a feature is a single floating point number. Let us assume that the class

protein_label_image :: image has only one feature called protein_amount. In order to express that a segment is *heavily stained* we can specify a user-defined predicate has_prop as follows:

```
has_prop(I, heavily_stained, S) IF
        S..features.protein_amount > 100 .
```

meaning that segment S of image I has the property "heavily_stained" if the staining intensity is greater than 100.[18] We assume a relation derived_with(P, I_1,I_2) that is instantiated every time a researcher creates a finer resolution image I_2 based upon some property P on a coarser resolution image I_1. Then the rule

```
highlight_parent_segments(I2,S) IF
        derived_with(P, I1,I2), I1 : protein_label_image,
        I1[anatomical_structures..segments↠{S}],
        has_prop(I2,P,S).
```

can be used to encode the relationship between the two images. Used this way, the rule will produce all possible segments in I_1 that could have produced the image I_2. We could also use a rule with the same body to return all derived images from a given image.

Meta-Reasoning with Schema

The mediator performs meta-reasoning of the schema of TAXON to create a class hierarchy of animals. Consider the schema of TAXON:

```
taxon[subspecies⇒string; species⇒string; genus⇒string;
    family⇒string; order⇒string; infraclass⇒string; ...
    ... phylum⇒string; kingdom⇒string; superkingdom⇒string].
```

At the mediator, a hierarchy is defined for the taxonomic ranks:

```
subspecies :: species :: genus :: ... :: kingdom :: superkingdom.
```

Now the *data* in the TAXON database is used to infer the taxonomic *class hierarchy*:

```
T : TR, TR :: TR1 IF
    T : 'TAXON'.taxon[Taxon_Rank→TR; Taxon_Rank1→TR1],
    Taxon_Rank :: Taxon_Rank1.
```

The rule states that given two taxon ranks, e.g., order and kingdom with data values *rodentia* and *metazoa* respectively, and given that kingdom is a subclass of class, then *rodentia* is a subclass of *metazoa*. In other words, from the *data* of TAXON we infer new *schema information*, i.e., that all rodents belong to the metazoe kingdom. As we will show in the next section, this rule will be used in a query to determine the appropriate taxonomic ranks for computing joins and closures.

[18] Instead of the equivalent body S[features↠_[protein_amount→A]], $A > 100$ path expressions with ".." and "." are used here.

Rule Export from Knowledge Bases

Knowledge bases export rules to the mediator. The anatomical knowledge base, for example, contains both an is-a and a has hierarchy. Thus a Purkinje cell is a neuron and cerebellum has a Purkinje cell layer. We also use the predicate located_in as an inverse of the has relation. Thus the fact that Purkinje cell is located_in Purkinje cell layer implies it is also located in the cerebellum. This rule is used in the mediator to create a transitive closure over the locations of neuro_anatomic_entity(ies) during a query.

6 Semantic Integration from the Mediator's and User's Perspective

To illustrate how an integrated query is evaluated in the Integrated Neuroscience Model, we trace through the phases of evaluating the first example query:

(1) *Find the cerebellar distribution of all rat proteins with more than 90% amino acid homology with the human NCS-1 protein.*

The broad steps for evaluating this query in the INSM module are: (i) retrieve facts about *shared homologies* where homology>90% (uses CAPROT), (ii) determine the *protein distribution* using data from PROLAB and ANATOM, and (iii) compute the *aggregate*, grouped by anatomical structure.

More precisely, let us assume that the mediator defines and exports the following two views called homologous_proteins and aggregated_protein_distribution.[19]

The first can be treated as a relation

```
homologous_proteins(Protein1,Animal1, Protein2, Animal2,
        Name_type, Value)
```

Here the two protein-animal pairs refer to the variety of the specified protein as found in the given animal. This relation depicts that given two such pairs, the database stores how similar they are in terms of their amino acid sequence as a percentage value. The attribute name_type specifies whether the common name or the scientific name of the animals have been specified in the query[20]. Similarly, the second view can be treated as the relation

```
aggregated_protein_distribution( Protein, Organism_name,
    Name_type, Brain_region, Feature_name,
    Anatom_struct, Result)
```

The relation records the distribution of a feature (such as protein_amount) of proteins occurring in the brain region of an organism, grouped by the anatomical structures in that brain region. The user's query is expressed in terms of these views in the following manner.

[19] Note that such information can be obtained as a service from several Web sites [PRO00].

[20] We make the simplifying assumption that for both animals the name type is the same in the query.

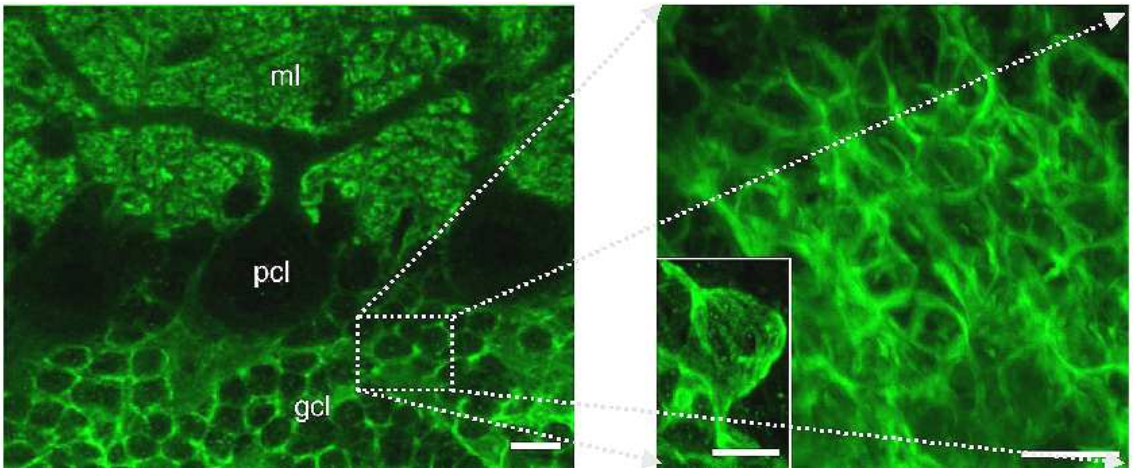

Figure 2. Immunofluorescent localization of potassium channel subunit Kv3.1b in the cerebellum [SMW+97]. **Left**: Distribution of Kv3.1b in cerebellar cortex labeled with fluorescein and imaged on a Bio Rad 1024 MRC confocal microscope. **Right**: A 3D projection through the granule cell layer (at higher magnification). A 3D network of ring-like structures are apparent. Scale bar, 10 μm. The inset shows a single granule cell in the white matter, displaying three strongly labeled emerging processes and thick as well as fine labeled rings around the soma. Inset scale bar, 5 μm.

```
query1(Anatom_struct, Result ) IF
    homologous_proteins("NCS-1", "human", Rat_Protein,
        "house rat", common, Value),
    Value > 90,
    aggregated_protein_distribution(Rat_Protein, "house rat",
        common, "cerebellum", "protein_amount",
        Anatom_struct, Result).
```

In the mediator the first view homologous_proteins is constructed by importing from module CAPROT the class amino_acid_homology. The view definition of homologous_proteins based on amino_acid_homology is:

```
homologous_proteins(Protein1,Animal1, Protein2, Animal2,
    Name_type, Value) IF
        _ : amino_acid_homology[
            shared@(
                _ : protein_in_animal[
                    name→Protein1;
                    found_in→ _ : animal[
                        name@(Name_type)→Animal1]],
                _ : protein_in_animal[name→Protein2;
                    found_in→ _ : animal[
                        name@(Name_type)→Animal2]])
            → Value].
```

Note that since this entire view is in the scope of one source module we do not need to qualify names by the module name.

The definition of the second view illustrates the use of aggregation (here: summation of Values, grouped by Anatom_struct):

```
aggregated_protein_distribution(
  Protein, Organism_name, Name_type, Brain_region,
  Feature_name, Anatom_struct, Result) IF
      Result = sum{ Value [Anatom_struct] ;
        protein_distribution(Protein, Organism_name
          Name_type, Brain_region, Feature_name,
          Anatom_struct, Value)}.
```

Here the view protein_distribution is defined by importing the protein_label_image class of module PROLAB and the class neuro_anatomic_entity class of module ANATOM.

Finally, a *semantic join* based on the ANATOM knowledge base is illustrated by the following rule:

```
protein_distribution(Protein, Organism_name, Name_type
            Brain_region, Feature_name,
            Anatom_struct, Value) IF
    I:'PROLAB'.protein_label_image[
        proteins→→Protein;
        organism@(Name_type)→Organism_name;
        anatomical_structures→→
            {A : 'PROLAB'.anatomical_structure[
                name→Anatom_name]}],
    NAE : 'ANATOM'.neuro_anatomic_entity[
        name→Anatom_name;
        located_in→→{Brain_region}],
    A..segments..features[name→Feature_name; value→Value].
```

In this view-definition rule the last two arguments of protein_distribution are used as output variables while the rest are used as input variables. The anatomical structure from the PROLAB and ANATOM modules are explicitly joined using the variable Anatom_name. As explained before, the recursive definition of located_in in the ANATOM module, causes the rule to transitively traverse every substructure of the cerebellum down to the cellular level in order to find the

"leaf level" anatomic structures where the protein is localized. This constitutes the semantic join between Brain_region and the anatomical structure A whose features are being extracted.

(2) *Compare this with the distribution of this protein or its homologs in other rodents.*

The primary difference between this query and the previous one is that it is executed over the set of *all rodents except rat*, and that it uses information from the module TAXON. With this modification, the second query is stated as:

```
query2(Anatom_struct, Result ) IF
  homologous_proteins("NCS-1", "Homo sapiens",
      Rodent_Protein, Rodent_name, scientific, Value),
  _: 'TAXON'.taxon[order→"Rodentia"; genus→G; species→S ],
  words(Rodent_name, [G,S]),
  Scientific_name =\= "Rattus rattus",
  Value > 90,
  aggregated_protein_distribution(Rat_Protein, "house rat",
      common, "cerebellum", "protein_amount",
      Anatom_struct, Result).
```

In this query we use the scientific rather than the common names of organisms, and we explicitly use the information that "Rodentia" is a value of the *order* attribute of the class taxon. This directly collects all known rodents in a set over which the rest of the query is evaluated. A less straightforward (but perhaps easier for the less knowledgeable user) way of evaluating the query could be to walk the transitive relationship of taxonomic classes to discover that we need all species under the order "Rodentia". Also note that the species "Rattus rattus" has been explicitly eliminated from the set to compute the rest of the aggregated protein distributions.

7 Discussion and Outlook

The KIND mediator system [KIN00] is being developed in the context of the MIX[21] project [MIX99a] at SDSC and UCSD, as an enhancement to the MIX mediator system MIX*m* [MIX99b]. MIX*m* uses XML as a common semistructured data model and XMAS[22], an XML query language resembling XML-QL [XML98a], as the view definition and integration language. Thus, like many other mediator approaches (e.g., [TSI98, LOR98, CDSS98]), MIX*m* is based *solely on a semistructured model* for information integration. While the semistructured data model and its most prominent representative XML certainly allow the flexible handling of source data, they do not by themselves support a rich semantic model as is required for complex application domains like the one we have described. Indeed, the weaknesses of XML DTDs as a schema mechanism are well-known and the XML Schema effort [XML99] highlights the necessity for richer modeling constructs.

However, in order to capture the semantics of complex "multiple worlds" integration scenarios like biological studies (Section 3), much more powerful and flexible formalisms like F-logic are needed to adequately handle the semantic integration of sources. Indeed, similar (but more restricted) logical formalisms have been used [DK97, CL93] for information integration between heterogeneous databases. In our architecture, we have introduced the concept of *source modules* (i.e., databases or knowledge bases which the mediator can query) that not only have an XML DTD for describing the syntactic structure of exported data, but which also have a mapping to an associated object-oriented F-logic schema $\Sigma(M)$. Moreover, when modeling a source database, arbitrary complex integrity constraints $\mathrm{IC}(M)$ can be specified for M. Finally, sources may also express expert knowledge for "semantically gluing" together the otherwise unrelated sources. In this case modules export a rule base $\mathrm{IDB}(M)$.

While we are convinced that a formalism like F-logic can and in fact should be used for knowledge-based information integration, many technical issues remain to be addressed: The current prototypical implementation of the KIND mediator is based on FLORA, an F-logic to XSB-Prolog compiler [LYK99] that evaluates rules in a top-down manner. This is clearly desirable when integrated views are computed *on-demand*, i.e., as the user queries the view. Conversely, for certain integration tasks, a bottom-up F-logic engine like FLORID [FLOb, LHL+98] can be advantageous, for example when the results of complex integration steps are to be materialized as in the case of the STUDIES database (see Figure 1). Apart from the adequate modeling of the involved source, the technically challenging problem remains how to efficiently evaluate rules, in particular in the distributed environment as it usually exists when integrating information.

An obvious shortcoming of the current prototypical implementation is that it consists of one central mediator, and that sources do not support autonomous query evaluation. Instead, in the current system, the mediator has the burden of retrieving all potentially relevant objects from the source and cannot push more specific rewritten queries to the sources. In future work we need to investigate how the mediator can make good use of different query capabilities of sources (e.g., some data is available from relational databases) in order to optimize overall performance and how to extend the system by user-defined functions and data types. Finally, another challenging problem that has not been addressed yet is the design and implementation of an end-user friendly user interface which would allow the domain expert to issue and refine ad-hoc queries and visualize results in an intuitive way.

[21] ***M**ediation of **I**nformation using **XML***

[22] ***XML M**atching **A**nd **S**tructuring Language*

References

[Abi97] S. Abiteboul. Querying Semi-Structured Data. In *Intl. Conference on Database Theory (ICDT)*, LNCS 1186, pp. 1–18. Springer, 1997.

[BE96] O. Bukhres and A. K. Elmagarmid, editors. *Object-Oriented Multidatabase Systems: A Solution for Advanced Applications*. Prentice Hall, 1996.

[CDSS98] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your Mediators Need Data Conversion! In *ACM Intl. Conference on Management of Data (SIGMOD)*, pp. 177–188, 1998.

[CGL+98] D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information Integration: Conceptual Modeling and Reasoning Support. In *6th Int. Conf. on Cooperative Information Systems (CoopIS)*, 1998.

[CL93] T. Catarci and M. Lenzerini. Representing and Using Interschema Knowledge in Cooperative Information Systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[CRD94] Y. Chang, L. Raschid, and B. Dorr. Transforming Queries from a Relational Schema to an Equivalent Object Schema: A Prototype Based on F-Logic. In *Intl. Symposium on Methodologies in Information Systems, (ISMIS)*, LNCS 869, pp. 154–163. Springer, 1994.

[DEFS99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In *DS-8: Semantic Issues in Multimedia Systems*. Kluwer, 1999.

[DFKR99] H. Davulcu, J. Freire, M. Kifer, and I. Ramakrishnan. A Layered Architecture for Querying Dynamic Web Content. In *ACM Intl. Conference on Management of Data (SIGMOD)*, pp. 491–502, Philadelphia, 1999.

[DK97] S. Davidson and A. Kosky. WOL: A Language for Database Transformations and Constraints. In *Intl. Conference on Data Engineering (ICDE)*. IEEE, 1997.

[DOTW97] S. B. Davidson, G. C. Overton, V. Tannen, and L. Wong. BioKleisli: A Digital Library for Biomedical Researchers. *Intl. Journal on Digital Libraries*, 1(1):36–53, 1997.

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

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

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

[HF96] C. Hafner and N. Fridman. Ontological Foundations for Biology Knowledge Models. In *4th Intl. Conf. on Intelligent Systems for Molecular Biology (ISMB-96)*, pp. 78–87. AAAI Press, 1996.

[Kas96] A. Kashyap, V.; Sheth. Semantic and Schematic Similarities between Database Objects: A Context-based Approach. *VLDB Journal*, 5(4):276–304, 1996.

[KH96] S. H. Koslow and M. F. Huerta, editors. *Neuroinformatics: An Overview of the Human Brain Project*. Lawrence Erlbaum Associates, 1996.

[KIN00] KIND (Knowledge-Based Integration of Neuroscience Data). `www.npaci.edu/DICE/Neuro/KIND/`, May 2000.

[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.

[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.

[LOR98] The LORE page at Stanford. `www-db.stanford.edu/lore/`, 1998.

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

[LYK99] B. Ludäscher, G. Yang, and M. Kifer. FLORA: The Secret of Object-Oriented Logic Programming. Technical report, State University of New York, Stony Brook, June 1999. see also `www.cs.sunysb.edu/~sbprolog/flora/`.

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

[MIX99a] MIX (Mediation of Information using XML). `www.npaci.edu/DICE/MIX/` and `www.db.ucsd.edu/Projects/MIX/`, 1999.

[MIX99b] MIXm (MIX Mediator System). `www.db.ucsd.edu/Projects/MIX/MIXm`, 1999.

[PRO00] Protein/Amino Acid Databases. `www.bioscience.org/urllists/protdb.htm`, May 2000.

[PS98] C. Parent and S. Spaccapietra. Issues and Approaches of Database Integration. *Communications of the ACM*, 41(5):166–178, 1998.

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[SMW+97] C. Sekirnjak, M. E. Martone, M. Weiser, T. Deerinck, E. Bueno, B. Rudy, and M. Ellisman. Subcellular Localization of the K+ Channel Subunit Kv3.1b in Selected Rat CNS Neurons. *Brain Research*, 766(1–2):173–187, 1997.

[TSI98] TSIMMIS Homepage. `www-db.stanford.edu/tsimmis/tsimmis.html`, 1998.

[Tür99] C. Türker. *Semantic Integrity Constraints in Federated Database Schemata*. DISDBIS 63, infix-Verlag, 1999. Ph.D. thesis, Fakultät für Informatik, Universität Magdeburg.

[Wie92] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

[WJ98] G. Wiederhold and J. Jannink. Composing Diverse Ontologies. Technical report, Stanford, 1998. `www-db.stanford.edu/SKC/publications/ifip99.html`.

[XML98a] XML-QL: A Query Language for XML. W3C note, `www.w3.org/TR/NOTE-xml-ql`, 1998.

[XML98b] Extensible Markup Language (XML). `www.w3.org/XML/`, 1998.

[XML99] XML Schema, Working Draft. `www.w3.org/TR/xmlschema-1/`, December 1999.

[XPa99] XML Path Language (XPath), Version 1.0. W3C Recommendation, `www.w3.org/TR/xpath`, 1999.