AVAILABLE AT
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT•

Neural Networks 16 (2003) 1277–1292

Neural Networks

www.elsevier.com/locate/neunet

2003 Special issue

Towards a formalization of disease-specific ontologies for neuroinformatics

Amarnath Gupta[a], Bertram Ludäscher[a,*], Jeffrey S. Grethe[b], Maryann E. Martone[b]

[a]San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0505, USA
[b]Department of Neuroscience, University of California San Diego, USA

Received 31 January 2003; revised 11 July 2003; accepted 11 July 2003

Abstract

We present issues arising when trying to formalize *disease maps*, i.e. ontologies to represent the terminological relationships among concepts necessary to construct a knowledge-base of neurological disorders. These disease maps are being created in the context of a large-scale data mediation system being created for the Biomedical Informatics Research Network (BIRN). The BIRN is a multi-university consortium collaborating to establish a large-scale data and computational grid around neuroimaging data, collected across multiple scales. Test bed projects within BIRN involve both animal and human studies of Alzheimer's disease, Parkinson's disease and schizophrenia.

Incorporating both the static 'terminological' relationships and dynamic processes, disease maps are being created to encapsulate a comprehensive theory of a disease. Terms within the disease map can also be connected to the relevant terms within other ontologies (e.g. the Unified Medical Language System), in order to allow the disease map management system to derive relationships between a larger set of terms than what is contained within the disease map itself. In this paper, we use the basic structure of a disease map we are developing for Parkinson's disease to illustrate our initial formalization for disease maps.
© 2003 Elsevier Ltd. All rights reserved.

Keywords: Neuruoinformatics; Semantic; Parkinson's disease; Logic Model; Ontology

1. Introduction

Recently there has been a significant increase in the development and publication of terminological systems for biology. In addition to general-purpose controlled vocabularies such as the Unified Medical Language System (UMLS) (National Library of Medicine, 2003) and Gene Ontology (Gene Ontology Consortium, 2002), a large number of more specialized vocabularies are being created. For example, TaO (TAMBIS Ontology) (Stevens et al., 1999) is an ontology for protein properties, motifs and similarities; The Cyc family of ontologies (EcoCyc (Karp, Riley, Paley, Pellegrini-Toole, & Krummenacker, 1999), MetaCyc (Karp, Riley, Saier, Paulsen, & Pellegrini-Toole, 2000), HinCyc (Karp, Ouzounis, & Paley, 1996)) describe the genes, gene product function, metabolism and regulation within specific species such as *E. coli* and the *H. influenza*; whereas the MGED Ontology provides standard terms for the annotation of microarray experiments. In the domain of neuroscience, BrainML (Gardner, Xiao, Abato, Knuth, & Gardner, 2002) is a controlled vocabulary for describing the standard vocabulary of neurophysiological experiments. NeuroML represents a standardized vocabulary to express information about neural simulations.

Despite this growth, it was observed by (Karp, 2000; Williams & Andersen, 2003) and others that many of the publicly available ontologies remain just controlled vocabularies and do not satisfy the primary requirements of being a formal ontology that can used for purposes like automated logical interpretation. Hence they cannot be easily integrated into larger information management systems. Gruber (1993) defined an ontology as a "formal explicit specification of a shared conceptualization",[1] where conceptualization refers to "an abstract model of how people think of things in the world, usually restricted to a particular subject area" (Guninger & Lee, 2002).

* Corresponding author. Tel.: +1-858-822-0864; fax: +1-858-822-0861.
E-mail address: ludaesch@sdsc.edu (B. Ludäscher).

[1] See also Guarino's detailed discussion on the notion of 'formal ontology' (Guarino & Giaretta, 1995).

0893-6080/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2003.07.008

For the purposes of this paper, and following Gruber, by an ontology O we mean a *representational vocabulary* V and a set of *axioms* A, that constrain the interpretation and the well-formed use of this vocabulary. We first illustrate the notions of *constrained interpretation* and *well-formed use* through an example

Example 1. Consider a vocabulary $V = (T, R)$ where T is a set of terms denoting concepts, and R is a set of relationship names. As a simple example, let $T = \{$`cell,nerve_cell,neuron,axon,purkinje_cell`$\}$ that represents the user's world of neurons, and $R = \{$**`isa, part_of`**$\}$, that represent characteristic the usual subclass and part-of relationships for this domain. Assume that the user has represented the relationships among them through the following binary relations, called *facts*:

```
isa(nerve_cell, cell).
isa(neuron, nerve_cell).
part_of(axon, neuron).
isa(purkinje_cell, neuron).
```

Further, let us assume the ontology $O = (V, A)$ contains an axiom A_1 stating that **`isa`** is a transitive relationship, i.e.

$A_1 : \mathbf{isa}(X, Y) \leftarrow \mathbf{isa}(X, Z), \mathbf{isa}(Z, Y)$

With this ontology, we can use standard logic programming semantics[2] to conclude the evident fact: **`isa`**(neuron, cell), i.e. the neuron is a cell. We can also add the axiom:

$A_2 : \mathbf{part\_of}(X, Y) \leftarrow \mathbf{isa}(Y, Z), \mathbf{part\_of}(X, Z).$

to conclude **`part_of`** (axon, purkinje_cell). This is because if X is a part of Z and every Y is a Z, then X is a part of Y as well, as specified by this rule.

Let us assume that as part of a disease map 'exploration', we want to add the following axiom and inspect its consequence:

$A_3 : \mathbf{part\_of}(Y, Z) \leftarrow \mathbf{isa}(X, Z), \mathbf{part\_of}(Y, X).$

Will this rule still yield valid conclusions? For instance, with A_1 and A_3, we can conclude **`part_of`** (axon, nerve_cell) and **`part_of`** (axon, cell). These statements are of questionable validity—while it is true that an axon is a part of *some* nerve cells, it is certainly not a part of *all* nerve cells. Similarly, an axon is not a part of all cells. Hence, unless some other, more constraining axioms (rules) are in place, the errant inferences cannot be avoided for this newly added exploratory rule. On the other hand, the disease map explorer could have added some other similar rule which leads to *plausible* (albeit unverified) conclusions. In that case, she may decide to keep the hypothetical rule and qualify the conclusions using a 'may(be)' operator (cf Section 3.9).

The example above illustrates that the design of an ontology is not just the collection or standardization of a large vocabulary that represents the 'terminology base' of a domain, it is the characterization of useful and interpretable direct and inferred relationships among the terms in the vocabulary, geared toward a specific set of tasks.

[2] See Section 3.

The goal of this paper is to present issues arising when attempting to devise a formal model for *disease maps*, i.e. ontologies to represent the terminological relationships among concepts necessary to construct a knowledge-base of neurological disorders. We use an initial formalism as the basic foundation to represent the content of disease maps, to develop computational procedures to search and analyze the ontology, and to integrate multiple information sources that contain data about the neurological disorders modeled with disease maps.

2. Disease maps: scope and desiderata

We introduce the motivation behind building our specific disease map, the Parkinson's Disease Map (PDM) (BIRN-PDM, 2003) through a brief description of the Biomedical Informatics Research Network (BIRN), a multi-institution project. The project studies a set of neurological disorders including Parkinson's Disease (PD), Alzheimer's disease, Schizophrenia, cases of clinical depression that progress into dementia, and corresponding animal models, particularly in the mouse. Different institutions specialize both in the patient and animal populations they observe and the categories of experimental data they collect. The data across the participants range from structural and functional MRI of human patients and animals to diffusion tensor images to high-resolution microscopy for immunolabeling and gene-expression, to electron tomography of relevant ultrastructures. A central goal of the BIRN project (http://www.nbirn.net) is to assimilate this wide variety of data in an effort to understand the basic mechanism of the target diseases. The PDM is incrementally created by a team of domain scientists and computer scientists as a consolidated, extensible ontology by combining information from existing public ontologies and building upon them new information from relevant research literature. Thus the roles of disease maps such as PDM are to serve as:

- a central body of knowledge that a researcher in the field can query, navigate through and analyze. This does not imply that a disease map will be an all-encompassing 'world repository' of such knowledge, but rather will be part of a group's own 'local repository' that researchers in a specific project like BIRN would construct and utilize.
- a 'glue ontology' that helps one to perform semantic (i.e. ontology-based) information integration across the experimental data collected by the partner institutions

Scope. Ontologies can be considered as closely related to, or variants of, knowledge bases (Guarino & Giaretta, 1995). Since a disease map itself can be seen as a special ontology, the question arises how disease maps differ from ontologies and knowledge bases in general, and expert systems in particular. A famous example for the latter is MYCIN (Buchanan & Shortliffe, 1984), which was performed simple diagnosis and treatment recommendation for certain infections, based on partial information from a 'dynamic questionnaire' on symptoms and test results. Our disease map approach is fundamentally different from MYCIN-like expert systems. It differs, e.g. from MYCIN in its purpose (diagnosis and treatment suggestions in the case of MYCIN, and 'knowledge exploration' in the case of disease maps), and in its underlying technology (Dempster-Shafer reasoning with uncertainties in MYCIN vs logic semantics of disease maps). Moreover, MYCIN-style expert systems and 'conventional' ontologies and knowledge bases are not designed in a way which would make the encoded knowledge the focus of a 'knowledge exploration'. In contrast, this is precisely the focus of disease maps: capturing various aspects of some knowledge about certain diseases, and then exploring—via graph—queries or other deductive reasoning—what the given knowledge entails and how certain modeled aspects relate to one another.

Also note that disease maps are aimed at capturing certain aspects of *processes* (e.g. biochemical processes in the human body). However, this does not mean that the actual *dynamic/concurrent behavior* is meant to be simulated or fully captured by disease maps. Rather, disease maps model only a high-level abstraction of certain dynamic aspects of processes. If a detailed process model is required, then, in principle, the specific simulation or process model at hand may be 'plugged into' a disease map, albeit not as an integral part of the reasoning process, but as an illustration of a specific behavior of a modeled process.

2.1. Content of a disease map

In the following we present some design criteria we have identified while developing the Parkison's Disease Map PDM.

Multiple perspectives. A disease and its animal models are complex phenomena and different scientists view them from different perspectives. The clinician's set of terms and relations identifying the character of the disease is distinct from a physiologist's viewpoint. In the following example, we illustrate this distinction.

Example 2. Consider the term *resting tremor*. From a clinical viewpoint one can make the statements:

isa(tremor, movement_disorder).
isa(resting_tremor, tremor).
occurs_in(resting_tremor, posture(resting)).
clinically_associated(resting_tremor, parkinsons_disease).

From a physiological viewpoint, one can say:

isa(oscillation(oscType), process).
causes(oscillation, neural_firing(neuron)).
isa(tremor, oscillation(abnormal)).
causes(neuron_of_ventral_intermediate_nucleus)(neural_firing(abnormal), resting_tremor).

Here, we use slightly different versions for the causes relation, one which just pairs cause and effect, and one which additionally describes the context in which the causality holds: **causes**(Cause, Effect) and **causes**(Context)(Cause, Effect). Notice in our notation that also a concept like posture can be parameterized by the term resting, and that a term like oscillation can be parameterized by an oscillation type (here any subtype of oscType).

In general, a statement of the form $r(\bar{x})(p(\bar{y}), q(\bar{z}))$ with $\bar{x}$, $\bar{y}$ and $\bar{z}$ (possibly empty) parameter vectors can be visualized as

$$p(\bar{y}) \xrightarrow{r(\bar{x})} q(\bar{z})$$

which can be read as '$p(\bar{y})$ is $r(\bar{x})$-related to $q(\bar{z})$'. The relation parameters $\bar{x}$ can be used to describe the context in which the statement holds, or to denote a refinement of the relation r.

As an ontology, the disease map needs to preserve the separate threads along which a term can be perceived, and yet provide a means to correlate them. To satisfy the first requirement we need to distinguish the 'clinical block' from the 'physiological block' by a syntactic procedure, effectively 'tagging' all facts and rules in a block with the unique name of the block within which they occur.

To satisfy the second, we need to provide the mechanism to derive statements like *the (abnormal) firing of neurons in the ventral intermediate nucleus* ***maybe*** *(but not necessarily is) associated with PD*.

The example illustrates that unlike many ontologies that model only inter-object relationships, a disease map needs to model both interprocess relationships, as well as object-process relationships that may be relevant for a disease. Specifically, it needs to handle terms like *neuron_firing*, a nominalization of the process *fires(neuron)*, where both forms will appear as valid terms in the ontology.

An important consequence of the multiple viewpoints is that semantically, the same lexical term can belong to different *roles* (e.g. resting_tremor is a symptom and it is also a process), very similar to the notion of 'word senses' used in dictionaries. The disease map thus needs to have *role specifiers* to syntactically determine which role is meant when a term is used in a fact, rule, or query.

Very often, the same relationship name used in the ontology may have distinct properties depending on the types of the terms they relate. For example, if *a* and *b* are *events*, then **part_of**(a, b) implies that the *temporal duration* of *a* is fully contained within the temporal interval of *b*. On the other hand, if they are anatomical structures then the *spatial extent* of *a* is fully contained within the spatial extent of *b*. The formalism of disease maps models this by making certain predicates *polymorphic*, i.e. different versions of a predicate are implied by the types of its arguments.

Note that the results of applying inference rules of a disease map cannot be considered valid under all circumstances, but instead should be interpreted as 'default logic conclusions' (Reiter, 1980), which can be invalidated or refined by the addition of new facts (see (Brewka & Dix, 2003) for a comprehsenive treatment of such nonmonotic behavior).

For example, let us assume we model causes a transitive, antisymmetric, irreflexive relationship. However, the transitivity of causes be interpreted cautiously. Let us say we have the facts:

causes(process1, process2).
causes(process2, process3).
...
causes(process9, process10).

Can we infer **causes**(process1, process3)? Can we also infer **causes**(process1, process10)? While the real answer always depends on the specific situation, quite often, the first answer will be affirmative but the second would be negative. This example illustrates two important problems in many ontologies including the UMLS. First, an intuitive relationship like causes has no obvious or well-specified semantics, and should possibly be broken down into a number of concrete relationships like may_cause and necessarily_causes that do have well-defined semantics. Second, the transitivity property of these relations may not hold beyond a certain length of the transitivity chain, requiring additional semantic specification of the relation.

Multiple granularities. A disease map must incorporate models of disease processes at various levels of detail so that the individual scientist can extend the map with information from his own experimental domain. This needs the formal framework to define extension mechanisms by elaborating on existing concepts, relationships, or some combination thereof, as well as an abstraction mechanism to reduce details if necessary.

Example 3. Consider the observation that the enzyme monoamine oxidase B ('MAOB') is an enzyme that catalyzes oxidation of dopamine in the substantia nigra pars compacta (Snpc).

isa('MAOB', enzyme).
p0: **isa**(oxidation(dopamine), process).
occurs in(oxidation(dopamine), 'Snpc').
p1: **catalyzes**('MAOB', oxidation(dopamine)).

Another researcher, trying to understand this process, comes up with the following detailed instantiation (Beal, 2001): *Neurotoxin 'MPTP' is converted to 'MPP$^+$' by 'MAOB'. The active form 'MPP$^+$' is picked up by the dopamine transporter, and released inside the neuron, where it accumulates in mitochondria. This leads to complex I (an antioxidant) inhibition, which leads to free radical generation*.

This can be expressed as follows:

isa('MPTP', neurotoxin).
isa('MPP$^+$', neurotoxin(active)).
isa(complex_I, antioxidant).
p2: **isa**(conversion('MPTP', 'MPP$^+$'), process).
p3: **catalyzes**('MAOB',conversion('MPTP', 'MPP$^+$')).
p4: **transports**(dopamine_transporter, 'MPP$^+$', inside(neuron)).
p5: **accumulates**('MPP$^+$', inside(mitochondria)).
p6: **contained-in**(complex_I,mitochondria).
p7: **inhibits**(p5, complex_I).

Here, we use *named statements* of the form $p_i : A$, where A is a logic atom, and p_i its unique identifier. The use of an identifier p_i as an argument in a fact (such as p5's use in p7) creates a new concept, the nominalization of the stated fact. For example, p7 can now be read as: accumulation (= the nominalized form of 'accumulates' in p5) of MPP$^+$ inside of mitochondria inhibits complex_I.

In this description, we did not explicitly represent that process p6 would lead to free radical generation because the system infers this from the fact that p6 inhibits complex_I and corresponding rules which specify inheritance through the **isa** The processes p2–p7 are one possible mechanism to realize processes p0–p1. The task of the formal model of a disease map is to make explicit this *elaboration relationship* between processes.

Animal models. An animal model is a model system where a natural or transgenic animal, exhibits some symptoms and/or pathological manifestations of a disease. Thus an animal model is a separate disease map by itself, parameterized by species characteristics and the properties that are related to the disease process. Consequently, there are terminological differences among disease maps of different species—for example, the terms *β-synuclein* and *γ-synuclein* should not appear in a Drosophila model of PD, but should in the mouse and human disease maps. The need to compare animal models impose on the disease map formalism the need to specify an extended form of what Peter Karp (Karp, 2000) calls *functional equivalence* across objects and processes of the animal model disease and the actual human disease. The anatomical structure of one animal should be mapped to the homologous structures in its counterparts (Bota & Arbib, 2001), the enzymes and the processes they participate in would need to be mapped as accurately as possible. Often this mapping is not obvious.

For example, it will be anatomically incorrect to map the retina of a drosophila to the substantia nigra pars compacta of a human. However, it might be appropriate to state:

model_of (neurotransmission)(X(drosophila),Y(human)) ← **isa**(X, dopaminergic_neuron(drosophila)), **isa**(Y, dopaminergic_neuron(human)).

That is, with respect to neurotransmission, dopaminergic neurons of the Drosophila are models of those of humans.

However, when this rule is applied to the disease maps, one should constrain its interpretation such that the system does not automatically make the deduction: **model_of**(X)(retina(drosophila), 'Snpc'(human)),
where X is anything other than neurotransmitter. Thus the formalism of the disease map needs to limit the propagation of inferences from mapping axioms.

Hypotheses and evidences. A primary utility of a disease map is to give its user the ability to place a hypothesis (derived from the literature, or conjectured by the user) in the map and explore how this hypothesis may compare with respect to other known or hypothesized facts or rules. This presents a number of requirements for the disease map formalism:

- It must be able to isolate hypotheses from known facts and rules, but at the same time, allow them to be used together to test for ramifications.
- Multiple hypotheses might contain or derive contradictory content. The formalism should harbor and detect these contradictions in a controlled manner, without becoming inconsistent itself.
- A hypothesis may have links to object instances in a database, such that the instance serves as the *evidence* of the hypothesis. Aside from comparing and testing hypotheses, querying the evidence of user-specified hypotheses is an important way to access experimental data connected to a disease map.

2.2. Accessing the disease map

In this section we explore the operational aspects of disease maps, highlighting different ways in which they can be used.

Specialization. The fundamental backbone of the ontology describing a disease map will be a general-purpose set of properties that characterize the genesis, activity, symptoms, treatment and the outcome of a disease. From this backbone, one should be able to construct the model for any disease either *incrementally* or *differentially*.

Incremental specialization. In the incremental method, a disease (e.g. PD) is described by specializing different aspects of the disease by adding facts and rules. For example, the fact that diseases under the familial form of PD have a genetic risk factor can be added by the following:

false ← **isa**(D, disease), **inheritable**(D), ¬ ∃H (etiology(heredity)(D,H), **isa**(H, hereditary_factor)).
inheritable(D2) ← **isa**(D1, disease), inheritable(D1), **isa**(D2, D1), ¬ **exception**(inheritability)(D2).
inheritable(familial_parkinsons_disease).
isa(inheritance(recessive), hereditary_factor).
isa(juvenile_parkinsons_disease, familial_parkinsons_disease).
etiology(heredity)(juvenile_parkinsons_disease, inheritance(recessive)).

The first rule is an integrity constraint and declares that for every inheritable disease there is a hereditary factor such that it is the etiology (of the hereditary kind) of the disease. If this were not the case, then a contradiction (**false**) would be derived by this rule. We treat terms like hereditary_factor as class names for which further subclasses can be derived. Of course, for a non-hereditary disease like idiopathic PD, there should be no hereditary_factor-this is ensured by the inheritable(Disease) predicate in the rule. The second rule states that if a disease is inheritable, then all subcategories of the disease are also inheritable, unless it is an exceptional disease with respect to inheritability. This illustrates the need for employing a default reasoning scheme in the formalism. This mechanism would then conclude from the next two statements that juvenile Parkinson's disease is inheritable. From the last statement, it should conclude that recessive inheritance is a hereditary factor for juvenile Parkinson's disease. We show in Section 3 that in ontologies such as the disease map, the process of specialization has interesting semantic properties.

Differential specialization. Often a disease map can be specialized by specifying the differences between a known disease and a new, to-be-defined disease. For example, Multiple Systems Atrophy differs from PD in several ways, two of which are: its onset age can be 30 (as opposed to 65 for PD) and it progresses much more rapidly. It also shows substantial loss in the density of the neurons that are postsynaptic to dopaminergic neurons, a phenomenon absent in PD. This can be stated as:

like(msa: multiple_systems_atrophy, pd: parkinsons_disease).
unlike(msa, pd, activity(onset_age)(multiple_systems_atrophy, Age), Age > 30).
associated_with(msa, loss(density(dopaminergic_neuron(postsynaptic))))).

Recall that throughout the paper we follow the convention to read the first argument of an atom as the 'subject', and the predicate symbol as the (possibly parameterized) predicate of the logical sentence. Therefore, *Multiple Systems Atrophy,* **`unlike`** *Parkinson's disease has and* `onset_age activity` *with an* `Age` *value greater than 30 (while the default onset age for PD (defined elsewhere) is greater than 60)*. Further note that here we use ':' to define a synonym.

The second statement, about the loss of density, is not given as an exception via the **`unlike`** because it does not 'replace' any fact from the facts about PD. Instead the differentiating property is given as a separate positive statement. However regardless of the manner of specialization, the internal representation of the disease properties should be identical.

Finally, a form of specialization can be used to *derive* new nodes. Suppose, the disease map has the facts:

isa(`dopaminergic_neuron, neuron`).
part_of(`axon, neuron`).

Using rules A_1 and A_2 from Section 1, we can derive the fact **part_of**(`axon, dopaminergic_neuron`). The usual interpretation of this is that every dopaminergic neuron has an axon. However, in this case, we want the system to *construct* an object, called the *axon of a dopaminergic neuron*, which is distinguished from all other axons, because we might want to define properties that pertain to only these axons. Suppose, we adopt the convention that these 'system-invented' objects (also called *derived nodes*) will be named: `of(axon,dopaminergic_neuron)`. We could write an 'object generating' rule:

isa(`of(D,C1),D`) ←
isa(`C1,C`), **part_of**(`D,C`), ¬ **exception-**(`C1`).

asserting, in our example, that `of(axon,dopaminergic_neuron)` is an axon.

Path finding. Our neuroscience users have identified *path finding* to be a very important class of queries that help them find interesting correlations among concepts, processes and their properties. A path search can be *ordered* or *unordered*. An *ordered path search* is given as a path expression over concepts and relationship names; the path expression can contain wild-cards and predicates to be satisfied by the nodes and edges of the path, but the expression constrains the order of the specified path elements. An *unordered path search* specifies the properties of the nodes and edges to be included and excluded from the path, and possibly on the number of occurrences of each type of node and edge, but not on their order. The utility of a path search can be illustrated by a researcher who wants to know of all paths that go from the term *α-synuclein* through the term *protein_aggregate(_Protein)* to the term *cell_death*, but not through the term *Lewy_body*. These paths, if present, would give the researcher a sense of all theories assembled in a disease map where cell death occurs due to protein aggregates that are not part of Lewy bodies. Note that the unspecified parameter in the term *protein_aggregate(_Protein)* refers to any protein that might form an aggregate.

Neighborhood finding. Neighborhood finding has been identified as another important operation with a disease map. If we treat the disease map as a graph over concepts (nodes) and relations (edges), neighborhood finding corresponds to a subgraph search where the properties of only some nodes, edges or paths are specified. A typical use is from a scientist who has created a tnetatis hypothesis in the form of a small set of concepts and relationships, and wants to situate it in a larger disease map. The neighborhood finding task would take a specified set of terms and relationships and locate all neighborhoods where they can fit. Once a desirable neighborhood is located, the user might want to further tune the results, for example, by adding to it the *k*-neighborhood of a node in the previous result.

Model matching. Comparison of one fragment of a disease map with another corresponds to the task of subgraph matching (homomorphism), and is the central requirement of evaluating animal models with human disease models. There are several different ways a user might want to perform a subgraph match. In many cases, producing the node and edge intersection between the two graphs, as well as their difference graph would suffice. In other cases, a numerical *matching score* is desired. Either way, the comparison needs to cover both the explicit and the derived relations (edges) of the disease map.

Semantic mediation. Semantic mediation refers to a class of information integration problems where two database schemas from different data sources are integrated by using a third, auxiliary body of knowledge (e.g. in the forms of facts and rules) provided by a domain expert. The presence of the auxiliary knowledge is essential, because without this additional glue, the schemas do not overlap enough to allow the definition of any integrated view over them. In this context, the role of the disease map is to act as a body of glue knowledge (Gupta, Ludäscher, & Martone, 2000; Ludäscher, Gupta, & Martone, 2001; Ludäscher, Gupta, & Martone, 2003) that ties together experimental data from the different participants of a federation. In the case of BIRN, the disease model helps to integrate information from researchers dealing with the human and animal forms of Parkinson's disease. A query on such an integrated view could be: *Which animal model shows a rate and distribution of neuron degeneration most similar to the patient population P*, where *P* can be specified by conditions on patient age, gender, medical history and treatment received.

In Section 3, we propose an initial formalization of disease maps using logic programs and illustrate how the resulting formalism can accomplish the various desiderata described above.

3. Towards a formal model of disease maps

It is common practice to model and visualize ontologies as *labeled directed graphs*. For example, the labeled edge $C \xrightarrow{r} D$ states that the concepts C and D are r-related. In particular, a labeled *tree* in which the only label r is '**`isa`**' describes a simple concept (or class) hierarchy.

As is the case for most ontologies, binary relationships of a disease map can be understood as labeled edges between nodes representing concepts. Such a graph representation facilitates the visualization of disease maps via graph layout programs, and allows the user to issue powerful graph queries to explore the graph structure and the relationships between concepts (e.g. one can compute *reachability under regular path expressions*, find *shortest paths*, *minimal spanning trees*, etc.). In this way, ontologies in general, and disease maps in particular not only serve as sources of 'terminological glue knowledge' in scientific data integration systems, but become study objects in their own right.

However, in order to capture scientific knowledge using disease maps, a pure graph-oriented model of disease maps is not sufficient, as it does not address the following crucial issues:

- What does an individual edge $C \xrightarrow{r} D$ really *mean*? For example, can C and D be understood as classes of objects? If so, what does this edge say about the relationship r between instances $c \in C$ and $d \in D$? Is *every* r-related object o of c in D, or does the edge just state that there is *some* such r-related o in D? A logic formalization of disease maps can resolve this ambiguity: e.g. the former meaning is specified using the description logic formula $C \sqsubseteq \forall rD$,, while the latter is specified by $C \sqsubseteq \exists rD$.
- Given a set of edges, what is their *combined meaning*? For example, does $C \xrightarrow{r} D \xrightarrow{r} E$ also imply that $C \xrightarrow{r} E$? The answer is yes for $r =$ **`isa`** and (in general) no for $r =$ `inhibits`. A logic formalization allows us to specify what facts are or are not implied by certain 'edge configurations'.

3.1. Disease maps as logic programs

We use a formalization of disease maps as logic programs for the following reasons: Logic programs are a standard, well-understood formalism for knowledge representation and reasoning (Baral, 2003; Brewka & Dix, 2003). Logic rules provide a concise, declarative specification mechanism for defining the semantics of 'edge interactions' in disease maps. The meaning of the sometimes quite intricate interactions among different logic rules and axioms is unambigously given by the declarative logic programming semantics of the rule set. All common relational integrity constraints and arbitrary application-specific semantic constraints can be expressed as logic programs. Finally, logic programs are *executable specifications* and complex queries over these specifications can be evaluated by deductive database engines such as the XSB system (Sagonas, Swift, & Warren, 1994).

The *syntax* of logic programs is defined as follows: A *logic program* is a set of *logic rules* of the form $H \leftarrow B$, where the *head* H is a logic atom, and where the *body* B is a conjunction of *literals*. If B is empty, then we say H is a *fact*. A literal is an atom A or its negation $\neg A$. An *atom* (short for: atomic formula) is an expression of the form $r(t_1, \ldots, t_n)$, where r is an n-ary relation symbol and the t_i are terms. The set of terms is constructed in the usual way, based on a set of constants, variables, and function symbols. In particular, if $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \ldots, t_k)$ is a term. When writing concrete logic rules, variable names are capitalized, e.g. $X, Y, \ldots$, while all other symbols (i.e. constants, function symbols, and relation symbols) are lower-case.

The *semantics* of a logic program is given using the notion of a *model* of the program. Due to lack of space, we only provide the basic intuition; see, e.g. (Brewka & Dix, 2003) or (Abiteboul, Hull, & Vianu, 1995) for details (using logic programming and database perspectives, respectively). A logic *interpretation* assigns a meaning to the syntactic constructs of the language, e.g. by mapping constants and terms to domain elements, function symbols to functions, and relation symbols to relations. For the purpose of answering queries, one considers only *Herbrand interpretations* which interpret the underlying domain syntactically or symbolically. In particular, the *Herbrand universe* consists of all ground terms that can be constructed from constants and function symbols. Relations are then interpreted over this universe of terms. A (Herbrand) *model* of a logic program is a (Herbrand) interpretation that *satisfies* all facts and rules of a program.

Intended, declarative semantics. The question arises: Which model(s) of a program are the 'right' and intended ones. The logic programming community has extensively studied and solved this problem (Dix, 2003). For positive logic programs P (i.e. not involving any negated subgoal) the intended semantics is given by the unique *minimal model* of P. The minimal model can be obtained as the intersection of all Herbrand models or, equivalently, as the least fixpoint when 'firing' the rules simultaneously, starting from the facts. For programs with negation, there are two main accepted semantics, the *well-founded semantics* (Van Gelder, Ross, & Schlipf, 1991), and the *stable model semantics* (Gelfond & Lifschitz, 1988). Under the first semantics every logic program has a unique *well-founded model* which assigns a third truth-value $\perp$ (undefined) to atoms which cannot be decided to be either true or false because (i) its truth-value depends negatively on itself, and

(ii) there is no well-founded reasoning process using only the facts and rules of the program that would establish a unique truth-value. In contrast, the stable model semantics 'guesses' models and considers them stable if they reproduce themselves under a certain natural transformation (Gelfond & Lifschitz, 1988).

For example, for the logic program P with the two rules `a_lies` $\leftarrow \neg$ `b_lies` and `b_lies` $\leftarrow \neg$ `a_lies`, the well-founded semantics assigns the truth-value $\perp$ (undefined) to both atoms since it cannot be established whether a or b lies. P has two stable models though: in one `a` is a liar and `b` speaking the truth, while in the other model the roles of `a` and `b` are reversed.

Therefore, for disease maps, we adopt as our canonical semantics the less controversial well-founded semantics. Well-founded models can be computed efficiently (e.g. using the XSB deductive database (Sagonas et al., 1994)) due to their PTIME data complexity on function free programs. Moreover, every stable model coincides with the well-founded model on the true and false atoms and only interprets the undefined atoms as either true or false (according to a 'stable guess', i.e. one which does not contradict itself).

3.2. Edge semantics

A large number of facts in a disease map are binary relations, i.e., sets of tuples of the form $r(C,D)$. If C and D are classes, we can distinguish two kinds of edges:

- $C \xrightarrow{r} D$ (the default edge type) which stands for the description logic axiom $C \sqsubseteq \exists r.D$ (every $c \in C$ has *some* $r(c) \in D$). We formalize this using the logic rule[3]
- **false** $\leftarrow$ **class**$(C) \wedge$ **class**$(D) \wedge$ **instance** $(C_0, C) \wedge \neg \exists D_0{:}r(C_0,D_0) \wedge$**instance**$(D_0,D)$

Such a rule with the distinguished predicate **false** the head is called a *denial*. Denials are a convenient way to express *integrity constraints* by specifying what must not happen-here: there must not exist a $c_0 \in C$ such that there is no $d_0 \in D$ to which c_0 is r-related. If **false** be inferred under the well-founded semantics (or in a stable model), then an inconsistency has been detected.

$C \xrightarrow{\forall r} D$ which stands for the description logic axiom $C \sqsubseteq \forall rD$ (*all* r-related d of any $c \in C$ are in D). This integrity constraint is captured as follows:
false $\leftarrow$ **class**$(C) \wedge$ **class**$(D) \wedge$ **instance** $(C_0, C) \wedge r(C_0, D_0) \wedge \neg$ **instance**(D_0, D).

The rules states that there can be no $c_o \in C$ such that $r(c_0, d_0)$ and $d_o \notin D$.

[3] For conciseness and readability, we allow first-order rule bodies such as $\neg \exists D_0 : r(C_0, D_0) \wedge$ ***instance***(D_0, D). It is well-known how they can be translated into several standard logic rules.

Note that both axioms for the two different edge types apply only to concepts which have been declared to be classes, since only classes have instances (also known as the *members* of the class). The distinguished predicate **instance**(X, C) holds if X is an instance of class C.

Separation of class and instance level. If we want to disallow disease maps that blur the distinction between the schema level and the instance level, we can require that no class can be an *instance* of another class:

false $\leftarrow$ **class**(C),**class**(D),**instance**(C, D).

Clearly, C can still be a *subclass* of D, i.e. **class**(C),, **class**(D), and **isa**(C, D) can be true.

3.3. Parameterized concepts and relations

Often it is convenient to parameterize concepts or relationships. For example in the above examples for specifying semantic integrity constraints as denials, it is desirable not only to signal an inconsistency, but to provide an explanation for the inconsistency by 'witness terms'. For example, we can reformulate the above class-instance separation constraint as follows:

false(`cis`(C, D)) $\leftarrow$
class(C),**class**(D),**instance**(C, D).

Now if **false**(`cis(c,d)`) is derived under the canonical semantics, we know that the **c**lass-**i**nstance **s**eparation constraint has been violated for `c` and `d`.

Apart from the distinguished relation symbol **false**, any other relation can be parameterized as well. For example, the edge

$$\mathtt{disease} \xrightarrow{\mathtt{etiology(pathology)}} \mathtt{aberration(cell)}.$$

involves a parameterized relation `etiology(pathology)` and is represented using the fact `etiology(pathology)(disease, aberration(cell))`.

In `aberration(cell)`, aberration is a conventional first-order logic *function symbol* mapping a concept name c (here: `cell`) to the concept 'the aberration of c'. In contrast, `etiology` occurs at the position of a *relation symbol* which in conventional first-order logic cannot be parameterized. However, in real applications it is desirable to be able to handle different 'flavors' of a relation either uniformly across all flavors or differently depending on the specific flavor. A standard example is the **part_of** which, in a finer modeling, can be parameterized as **part_of**(F) to include flavors F such as *member/collection*, *portion/mass*, *phase/activity*, etc. (Artale, Franconi, Guarino, & Pazzi, 1996).

Here, we may query the system which flavors `etiology` has (e.g. `pathology, agent, vector, host`, etc.) and then define either overarching rules for multiple flavors or distinct ones for individual flavors.

3.4. Closure operations

A common semantic constraint for certain relations is to perform various kinds of 'deductive closure' operations. For example, a standard requirement for the **isa** hierarchy is that it should be transitive and antisymmetric. This can be formalized as follows:

$\mathbf{isa}(C,D) \leftarrow \mathbf{b\_isa}(C,D).$

$\mathbf{isa}(C,D) \leftarrow \mathbf{isa}(C,C'), \mathbf{isa}(C',D).$

$\mathbf{false}(\mathbf{on\_isa\_cycle}(C,D)) \leftarrow \mathbf{isa}(C,D), \mathbf{isa}(D,C), C$

$\neq D.$

The system can then derive all transitive edges using the first two rules, starting from a (typically much smaller) base relations **b_isa** 'initial' **isa** facts. The last rule allows the system to detect inconsistencies in the concept hierarchy and reports all pairs of concepts (C,D) which are involved in a (disallowed) concept cycle.

Similarly, the following rule closes all flavors of **part_of** transitively *within each flavor F*, but not across flavors (a variant of this rule can be used to achieve the latter):

$\mathbf{part\_of}(F)(X,Y) \leftarrow$
$\quad \mathbf{part\_of}(F)(X,Z), \mathbf{part\_of}(F)(Z,Y).$

Relation variables and hilog features. In addition to parameterized relations symbols, we allow variables at the position of relation symbols. For example, we can specify a generic transitive closure rule for any relation R as follows:

$\mathbf{tc}(R)(X,Y) \leftarrow R(X,Y).$

$\mathbf{tc}(R)(X,Y) \leftarrow R(X,Z), \mathbf{tc}(R)(Z,Y).$

While relation variables such as R and parameterized relations such as $\mathbf{tc}(R)$ correspond to a restricted second-order syntax, they have a standard first-order semantics via a simple transformation. For example $R(X,Y)$ is mapped to a conventional first-order atom $\mathtt{apply}(R,X,Y)$. This encoding is used, e.g. in Hilog (Chen, Kifer, & Warren, 1993) and F-logic (Kifer, Lausen, & Wu, 1995), and is implemented in the XSB system.

3.5. Inheritance

An import use of the **isa** relation is property inheritance: a subclass inherits all properties of its superclass. This is specified as follows:

$L(C,X) \leftarrow \mathbf{isa}(C,D), L(D,X).$

A similar rule can be used to inherit properties across the instance level. Note that *multiple inheritance*, i.e. a situation where C is a subclass of both D and E can lead to problems: what if the properties inherited along D and E contradict each other? There are several approaches to deal with this problem: e.g. if D itself is a subclass of E, then typically the more specific information from D is inherited, while E is ignored. If D and E are incomparable, then one can inhibit inheritance altogether, or specify some ad-hoc overriding policy.

3.6. Default inheritance

It is sometimes convenient to describe a new concept, such as a new disease, relative to another, similar one. This can be achieved by providing facts of the form $\mathbf{like}(X',X)$, stating that X' should by default inherit all properties of X, and the following rule:

$L(X',Y)\text{-} \leftarrow L(X,Y), \mathbf{like}(X',X), \neg\, \mathbf{exception}(L)(X',X).$

Now every property described by an edge $X \xrightarrow{L} Y$ is inherited from X to X' unless a fact $\mathbf{exception}(L)(X',X)$ states otherwise. This rule is *nonmonotonic*, since a larger set of exceptions results in a smaller set of inherited properties.

Exceptions can be stated directly as facts or indirectly using rules. We call this way of handling inheritance *default inheritance*, since property inheritance is not automatic through the **isa** relation, but based on direct or indirect conditions expressed through **exception** facts. Additional properties of X' (versus X) do not need special handling, but are simply stated as new facts for X'.

3.7. Semantic typing of functions and relations

Traditionally, first-order logic and logic programs are untyped. This means that we can apply any function symbol to any term, and instantiate any argument of a relation with any term as well. However, in order to detect certain errors as soon as possible, it is desirable to employ a type system, e.g. similar to the ones used in the functional programming languages Haskell and ML.

We obtain a semantic type system, by declaring for each relation and for each function, the types of each argument. A powerful polymorphic type system is obtained by allowing type declarations with type variables. Finally, the type hierarchy is obtained by extending a given subtyping relation '$\prec$' from the base types to complex types. The resulting type system helps to structure the domain and documents this structure, and—most importantly—it allows detection of certain specification errors in facts and rules of disease maps at compile-time, before evaluating the rules.

In the disease map, an example of semantic types can be found in the relation `cellular_structure` $\prec$ `anatomical_structure`. The arguments of predicates often have a semantic signature—for example,

```
anatomical_part_of(thing, anatomical_
structure).
codes_for(gene, protein, species).
codes_for(gene, peptide, species).
occurs_in(stuff, anatomical_structure).
```

Here *stuff*, as opposed to *thing*, refers to non-discrete entities like 'fat deposit'. There can also be a $\prec$ relationship between two predicates. For example, `related_to(process, disease)` is used when no further information is available, whereas, `causes(process, disease)` is a stronger relationship, and is subsumed by `related_to`.

3.8. Reasoning as argumentation

Well-founded and stable semantics can be used to resolve disputes via so-called argumentation frameworks (Bondarenko, Dung, Kowalski, & Toni, 1997; Dung, 1995). A variant of this can be used, for example, to perform a sort of hypothetical reasoning: Assume for modeling the regulatory behavior of genes controlling the pathogenesis of protein aggregates, that a relation `inhibits(Inhibiter,Inhibitee)` is given, together with a set of genes that are active in a certain scenario like Lewy body formation. We may wish to state that a substance is active if it is not inhibited, and that it is inhibited, if there is an active inhibiter:

$$\mathtt{active}(X) \leftarrow \neg\,\mathtt{inhibited}(X).$$
$$\mathtt{inhibited}(X) \leftarrow \mathtt{inhibits}(Y,X), \mathtt{active}(Y).$$

While this logic program is not stratified (the predicate inhibited depends negatively on itself), it still has a unique well-founded model that can be used to identify which of the 'arguing' inhibits statements ultimately 'win'. At present, we are trying to use this formalism to model the behavior of fibril formation during the production of Lewy bodies. In some 'drawn positions', the stable models of the program (if they exist) may provide additional insight. For example, if a and b mutually inhibit themselves, then there are two stable models, one in which a is active, and one in which b is active. We expect this model to simulate the reasoning for two contrasting hypotheses—one in which the Lewy body is the primary cause for cell-death, and another in which they are the protectors against the process of cell death.

A general argumentation framework (Dung, 1995) comprises an *argument generation unit* AGU and an *argument processing unit* APU. The former can be given simply by a binary relation `attacks(X,Y)`, stating that argument `X` attacks argument `Y`. The APU is then the simple yet very powerful logic program

$$\mathtt{defeated(Y)} \leftarrow \mathtt{attacks(X,Y),\ acceptable(X).}$$
$$\mathtt{acceptable(X)} \leftarrow \neg\,\mathtt{defeated(X).}$$

The first rule states that an argument `Y` is defeated if it is attacked by an argument `X` which is acceptable. The second rule states that an argument is acceptable if it is not defeated. The truth values assigned by the well-founded semantics to such an argumentation framework correspond to a cautious reasoning process, in which an alternating sequence of underestimates and overestimates ultimately converges to three sets of definitely acceptable, definitely defeated, and non-determinable arguments.

3.9. Homomorphisms

As described in Section 3.8, animal models can be thought of as a 'parallel world', from which certain knowledge *may* be transferrable to the human. In order to formalize the correspondences, we can use rules such as the following:

$$\mathbf{may}(L(F))(X',Y') \leftarrow L(X,Y), \mathbf{model\_of}(F)(X,X'),$$
$$\mathbf{model\_of}(F)(Y,Y').$$

This rule states that if there are **`model`** pairs (X,X') and (Y,Y') under the same flavor F (such as `neurotransmission`), and $L(X,Y)$ holds in one **`model`** (typically the animal model), then there may be a homomorphic statement $L(X',Y')$ (typically in the human model). Here, we use an operator **`may`** to indicate that this is not a definite inference, but derived from a 'parallel model'.

The above rule could be modified in several ways. For example, we may want to allow this rule to apply for compatible flavors F_1 and F_2 instead of a single flavor F only. For this compatibility may be defined, e.g. as $\mathbf{isa}(F_1,F_2) \vee \mathbf{isa}(F_2,F_1)$.

Another variant is to derive potential facts of the form $\mathbf{may}(\ldots)(X',Y')$ only if the correspondences through $\mathbf{model\_of}(\ldots)(X,X')$ facts are established in a certain defined neighborhood of X and Y and not just locally at a single point $L(X,Y)$.

4. Towards a disease map for Parkinson's disease

In this section we describe the basic structure of the disease map we are developing for Parkinson's disease and some of the related ailments. The map is created as a logic program divided into a number of sections.

Meta model. This section catalogs the basic object, process and relationships used through the rest of the disease map. The entire meta model is based upon a few primitive class-level concepts—`object`, `organism`, `process`, and `event` that get specialized in the course of the model. These primitives have minimal structure and semantics—an object has a name, an organism has a name that comes from the animal kingdom taxonomy, a process has a name, and optionally a time-interval, and an event has a name, a process in which it occurs and optionally, a time-of-occurrence. An `anatomical_structure` specializes the meta-class `object` by adding a parameter for the organism class it belongs to. Thus,

$$\mathtt{anatomical\_structure}(\mathtt{organism}) \overset{isa}{\rightarrow} \mathtt{object}.$$

We instantiate an anatomical_structure, the value substantia_nigra for humans as

$\mathtt{substantia\_nigra} \overset{\mathtt{instance}}{\longrightarrow}$

$\mathtt{anatomical\_structure}(\mathtt{human})$.

This implicitly makes the substantia_nigra inherit the organism as human. Specialization becomes a little more complex in case of the basal_ganglia, a *group* of nuclei that are given a single name. This is declared in multiple steps:

$\mathtt{neuron} \overset{\mathtt{isa}}{\longrightarrow} \mathtt{nerve\_cell}$.

$\mathtt{nucleus} \overset{\mathtt{isa}}{\longrightarrow} \mathtt{anatomical\_structure}(\mathtt{organism})$.

$\mathtt{basal\_ganglia} \overset{\mathtt{isa}}{\longrightarrow}$

$\mathtt{anatomical\_structure}(\mathtt{human})$.

$\mathtt{subcortical\_nucleus} \overset{\mathtt{isa}}{\longrightarrow} \mathtt{set\_of}$

$(\mathtt{neuron} | \mathtt{self.location} = \mathtt{outside}(\mathtt{neocortex}))$.

$\mathtt{caudate\_nucleus} \overset{\mathtt{isa}}{\longrightarrow} \mathtt{subcortical\_nucleus}$.

$\mathtt{basal\_ganglia} \overset{\mathtt{instance}}{\longrightarrow} \mathtt{caudate\_nucleus}$.

Here the condition self.location = outside (neocortex) is a restriction predicate that limits the possible locations of the members of the group being declared.

Process specialization occurs similarly - in a simplistic approach, let us say aberration, injury, degeneration and malfunction all specialize process. We can use them to define pathological_process as a specialization of process through a union rule by introducing a parameter to associate the process through with an aberration, injury of anatomical structure, or a malfunction of physiological system, or the aberration of a gene:

$\mathtt{pathological\_process}(Y)(X) \leftarrow$

$\mathtt{process}(X), \mathtt{aberration}(\mathtt{anatomical\_}$

$\mathtt{structure})(Y)$.

$\mathtt{pathological\_process}(Y)(X) \leftarrow$

$\mathtt{process}(X), \mathtt{injury}(\mathtt{anatomical\_structure})(Y)$.

$\mathtt{pathological\_process}(Y)(X) \leftarrow$

$\mathtt{process}(X), \mathtt{degeneration}(\mathtt{anatomical\_}$

$\mathtt{structure})(Y)$.

$\mathtt{pathological\_process}(Y)(X) \leftarrow$

$\mathtt{process}(X), \mathtt{malfunction}(\mathtt{physiological\_}$

$\mathtt{system})(Y)$.

$\mathtt{pathological\_process}(Y)(X) \leftarrow$

$\mathtt{process}(X), \mathtt{aberration}(\mathtt{genetic})(Y)$.

Disease definition. In this section, a disease is defined as a pathological process that has the following properties: one or more flavors of *etiology*, an *epidemiology*, a *vector*, one or more *hereditary risk factors*, one or more *symptoms*, one or more *pathological hallmarks*, an *activity pattern*, one or more *treatments*. Some of these properties are complex. For example, the activity pattern consists of *onset*, *progress* and *outcome*, where onset may have a number of attributes like age, onset_cause, and progress in a sequence of *phases*, each of which has a sequence of landmark_events. Of course, a specific disease may not have a need for all of these attributes. So when we specialize disease to define parkinsons_disease, we might assert:

$\mathbf{isa}(\mathtt{parkinsons\_disease}, \mathtt{disease})$.

$\mathbf{false}(\mathtt{not\_applicable}(\mathtt{pd\_spc})) \leftarrow$

$\mathtt{etiology}(\mathtt{pathology})(D, \mathtt{aberration}(S))$,

$\mathbf{isa}(D, \mathtt{parkinsons\_disease})$,

$\mathtt{anatomical\_structure}(S), \mathtt{anatomical\_part}$

$(S, \mathtt{spinal\_cord})$.

meaning that for Parkinson's disease (or any of its subtypes *D*), no fact about the aberration of any anatomical subpart of the spinal cord is relevant. Note that the parameterization not_applicable(pd_spc) of **false** characterizes the nature of error. In addition to not_applicable, we use not_known, not_recorded, not_present as other forms of *null* values.

The disease definition section also contains the symptoms that are clinically associated with Parkinson's and related diseases, and other diseases like acute polio and syphilis that have been identified as risk factors for Parkinson's disease. Diseases with clinically similar presentations (like Pick's disease) are also recorded. Many of our relationship names are borrowed from semantic relationship names of the UMLS. However, we have imposed additional constraints to ensure consistency of their properties within the disease maps. For example, the relationship is_clinically_similar is not transitive.

Anatomical structures. This section of the disease map captures the anatomical structures relevant for Parkinson's disease. It uses and references the concept identifiers from the UMLS. This allows the disease map management system (which functions as a mediator) to connect to a local copy of the UMLS, and derive relationships between two terms based both on the relationships specified by the disease map as well as the UMLS. For example, although the term 'cerebral peduncle' does not exist in the disease map, the UMLS provides the relationship anatomical_part_of(midbrain, substantia_nigra). The subcellular structures of the disease map make similar references to the 'Cellular Component' fragment of the Gene Ontology.

Some concepts need bidirectional implications. Consider the statements in Fig. 1, which state that if a cell contains an abnormal filament, it contains a filamentous inclusion, conversely, if something is a filamentous inclusion, it must contain an abnormal filament. This is a part of the disease map that would allow us derive, for example, that neurons in the substantia nigra pars compacta of a Parkinson's disease patient have filamentous inclusions.

Managing correspondence problems. Often when employing multiple sources, say UMLS and Gene Ontology, we come across terms such as `mitochondria` that are referenced by both. We use different identifiers, corresponding to the different occurrences, e.g. `UMLS.mitochondria` and `GO.mitochondria`, respectively. In this way, name clashes can be avoided by fully qualifying names with the source which specifies them. This resolution of name clashes has to be distinguished from the more general 'correspondence problem' across concept hierarchies or ontologies coming from different sources. For example, according to UMLS, the `substantia nigra` is a `part_of` the midbrain. However, according to (Swanson, 1998), `substantia nigra` is `part_of midbrain-hindbrain` and the term `midbrain` is not a distinct concept. Typically, whenever such a correspondence mismatch occurs, some spatial relationship such as `contains`, `covers`, `overlaps`, etc. Discovering the actual spatial correspondence relations is outside of the scope of disease maps. (Bota, 2001) uses a spatial inference technique to derive such atlas correspondance relations from different parcellations. In the current development of our disease map, we have not modeled such relationships. One way in which we may use such information in the future is by stating these spatial relations as logic statements. Description logics (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003) have been used to reason about concept interrelations and correspondences, e.g. to establish whether one concept subsumes or overlaps with another. See (Rector, 2003) for a similar approach to nomenclature management in medical informatics.

```
isa(neurofibril, fiber).
located_in(neurofibril, cytoplasm(axon)).

isa(axon, compartment(neuron)).
located_in(X, neuron) ⟵
          isa(Y, compartment(neuron)),
          located_in(X, Y).

isa(neurotubule, neurofibril). isa(neurotubule, microtubule).
isa(neurofilament, neurofibril). isa(neurofilament, filament).

located_in(X, cell) ⟵
          isa(X, cellular_inclusion).

contains(species)(X, abnormal(filament)) ⟵
          isa(X, filamentous_inclusion).

isa(X, filamentous_inclusion) ⟵
          isa(F,filament),
          contains(_)(X, abnormal(F)).
```

Fig. 1. A fragment of the Parkinson's disease map describing filamentous inclusions, a parent category of Lewy body.

Protein aggregation. The protein aggregation section defines the relationships between the concepts regarding the localization and content of the protein aggregates like Lewy bodies. For example the fact that high concentrations of the ubiquitin protein is found in the bound state is represented as:

concentration(*high*)(*bound*(*protein*(*ubiquitin*)),

lewy_body(*early_onset_parkinsons_disease*)).

Note that `protein(ubiquitin)` is distinguished from `ubiquitin` which is semantically typed as a gene. In order to express the fact that for the human version of late onset Parkinson's disease, a Lewy body expressing mutant α-synuclein may have *either* the a53t *or* the a30p mutant but not both, we use the following integrity constraints:

```
false ← contains(human)
(lewy_body(late_onset_parkinsons_
disease),
protein_aggregate(protein(a53t))),
contains(human)
(lewy_body(late_onset_parkinsons_
disease),
protein_aggregate(protein(a30p))).

false ← ¬ contains(human)
(lewy_body(late_onset_parkinsons_
disease),
protein_aggregate(protein(a53t))),
¬ contains(human)
(lewy_body(late_onset_parkinsons_-
disease),
protein_aggregate(protein(a30p))).
```

The first rule eliminates the case in which both proteins are present in the lewy body, whereas the second eliminates the case that neither one is.

The protein aggregation section also contains relationships that pertain to the genes found in Lewy bodies. Consider the statements:

```
has form(aggregate(protein(alpha_synu-
clein)))
(protein(alpha_synuclein), beta_plea-
ted_sheet).
ligase(denaturation)( protein(parkin),
protein(ubiquitin('E3'))).
```

The first states that sort protein aggregates, α-synuclein are found in the form β-pleated sheets, and the second states that the protein of the parkin gene acts as a ligase for the E3 form of the protein of ubiquitin, tagging them for the process of denaturation.

Cellular environment. This section records the terms and relationships related to the neurons participating in or affected by the disease process. This contains information such as the dopaminergic neurons of the substantia nigra pars compacta express protein(dopamine_transporter). The role of the dopamine transporter protein as described earlier (Section 2.1) in the paper also appears in this section.

Cell death. Finally, the **cell death** section contains the process description related to the common, agreed-upon knowledge and hypothesized models of cell death as they relate to Parkinson's disease. Two important aspects of the formalism are used in this section. First, for cell death, the inherent temporal semantics of concept names denoting processes become important. For example, cell death in Parkinson's disease is not acute - the time-course of cell death for PD is progressive, exhibiting a gradual transition between the onset of a cell injury, the development of the degeneration while the cell still performs its function, the slow withering of the cellular compartments, finally leading to its death. Pathologically, cell death has been related to Lewy bodies (Kahle, Haass, Kretzschmar, & Neumann, 2002). At a very coarse level one can make statements like:

```
isa(protofibril(X),
protein_aggregate(X)) ←
protein(X).
p1: isa(formation(protofibril
(alpha_synuclein)), process).
p2: isa(formation(fibril
(alpha_synuclein)), process).
p3: isa(formation(lewy_body),
process).
has_phases(dopaminergic_neuron)(cell_
death, [p1 → p2 → p3]).
occurs_in(cell_function(normal))
( p1, dopaminergic_neuron).
```

Here, the second argument in the phase_of relation, denotes an ordered sequence of phases, which could be elaborated into finer processes.

Secondly, this illustrates the treatment of hypotheses, which are modeled as simple facts or rules, that are specially labeled. For example, a statement like "Jones' hypothesis is protofibrils of α-synuclein are toxic to dopaminergic neurons (Goedert, 2001), but Lewy bodies protect them" can be written as:

```
hypothesis('Jones')(is_toxic
(protofibril(alpha_synuclein),
dopaminergic_neuron)).
hypothesis('Jones')(protects
(lewy_body,dopaminergic_neuron)).
```

If Smith's hypothesis is 'Lewy bodies cause degeneration of dopaminergic neurons', it can be similarly stated:

```
hypothesis('Smith')(causes
(lewy_body,degeneration
(dopaminergic_neuron))).
```

To make these two statements 'oppose' each other, we need to tell the system that the facts protects(X,Y) and causes(X, degeneration(Y)) attack each other (see Section 3.8):

```
attacks(protects(X,Y), causes
(X, degeneration(Y))).
attacks(causes(X, degeneration
(Y)), protects(X,Y)).
```

This can be made to produce two stable models of cell death depending on whether the toxicity of the protofibril or the Lewy body fibrils is assumed.

5. Querying the disease map

In Section 2.2, we outlined a number of different tasks we need to perform with a disease map. They can be categorized into three primary groups—extensional queries that perform a path or graph search, intensional queries that require logical derivation to perform a search, and defining integrated views, which might need a combination of the above. The query language that enables us to perform these operations is currently only partially specified, and its full treatment is beyond the scope of this paper. Therefore we outline the basic structure of the query language as we envision it, and present examples of queries for analysis of and mediation with disease maps.

Elements of the query language. As mentioned in Section 3.1, a disease map can be considered to be an edge-labeled graph with parameterized nodes and edges. The logic rules are producers of derived edges, and occasionally derived nodes. The basic form of the query is to give the system a *graph pattern* and request it to find all occurrences (called *witness graphs*) of the pattern in the map. Consider the query 'Find all diseases that have dementia as a symptom and Lewy bodies as a pathological feature'. As a query, this would be expressed as:

```
select graph_union($Path1, $Path2) from parkinsons_disease_map M    (1)
where M.node = $N1 and isa($N1.value, 'disease') and                 (2)
                 M.node = $N2 and $N2.value = 'dementia' and          (3)
                 M.node = $N3 and $N3.value = 'lewybody' and          (4)
       connects($Path1, $N1,$N2) and connects($Path2, $N1,$N3) and    (5)
                        edgeInPath({symptom}.$N2,$Path1) and          (6)
               pathInPath({etiology(pathology)}..$N3,$Path2).         (7)
```

In this query, we are looking for a graph, the union of paths `$Path1` and `$Path2` such that there are three nodes `$N1`, `$N2` and `$N3` satisfying the specified conditions. In line (2), `$N1` is bound to nodes whose value is disease or a subconcept thereof. Note that although '**isa**' can be construed to be transitive, unless we use the explicit form **tc**(**isa**) for the transitive closure of **isa**, the system just uses the direct **isa**. Line (5) of the query shows the system-defined `connect(Path, Node, Node)` function, where the `Path` variable binds to the set of paths that the two nodes are connected by. Line (6) illustrates the `edgeIn-Path(Edge, Path)` predicate, where the first argument is an edge expression that refers to an edge with the label '`symptom`'—one end of it is our desired graph node '`dementia`' and the other end is unspecified. Note that {*label*} denotes an edge label. Line (7) shows a similar construct `pathInPath(Path1, Path2)` for a sub-path expression, where the fragment `{etiology(pathology)}'..$N3` refers to an edge label `etiology(pathology)` followed by any number of nodes or edges leading to the node `$N3` which is a Lewy body (4).

Next, consider the query 'Which animal models share common features between Parkinson's disease (PD), Alzheimer's disease (AD) and Lou Gehrig's disease(LGD)?' Under the simplifying assumptions that the terms in the animal models match exactly the corresponding terms in the human disease models, this query finds the graphs corresponding to PD, AD and LGD, and intersects them to create a new graph IG that represents the common features of all three diseases (lines (1–13) in Fig. 2. Then it reports all animal models that have a non-empty intersection with IG (lines (14–16) in Fig. 2).

The reachability graph is computed in lines (2), (6) and (10) because it projects out the part of the graph that might be relevant to each disease. As noted by (Jagadish et al., 2002), our queries on databases of graphs (and trees) need to return both the *witness graphs* (the parts that matched the query conditions) as well as a *covering subgraph* (e.g. a minimal spanning tree that includes the nodes and edges of the witness graphs) surrounding the witness graphs. Also noteworthy is the observation that in practice, this rigid graph intersection would be impractical because the graph structure of the animal models would very likely be different from the human case. We are currently investigating the query language properties to specify imperfect graph matches.

Deductive queries. To pose a query that involves a deductive computation, we use a *rules* module, which is evaluated by a deductive database engine. For example, to query for all descendants of neurons using the transitive closure **tc**(**isa**) of the **isa** we would pose the following query:

```
select * from parkinsons_disease_map M,
    rules R(1) where M.node = $N1 and R.tc
    (isa)($N1.value, 'disease').
```

Integrated view definition. Consider two groups of researchers—one who works on a mouse model of PD, and the other who works on protein localization with high-resolution light and electron microscopy images. Let us

```
select * from (                                  (1)
    select reachability_graph($N1)               (2)
    from parkinsons_disease_map M                (3)
    where $M.node=$N1 and $N1.value='PD'         (4)
    graph_intersect                              (5)
    select reachability_graph($N2)               (6)      select * from animal_models (14)
    from parkinsons_disease_map M                (7)      graph_intersect             (15)
    where $M.node=$N2 and $N2.value='AD'         (8)      select * from $IG           (16)
    graph_intersect                              (9)
    select reachability_graph($N3)               (10)
    from parkinsons_disease_map M                (11)
    where $M.node=$N3 and $N3.value='LGD'        (12)
) into graph($IG)                                (13)
```

Fig. 2. Animal model queries.

assume the first group has a table with the following schema in their database:

process_evidence(*transgenic_id*, *model_type*, *process_name*, *image_location*, *observed_structures*, *evidence_details*).

The second group has a table:

protein_localization(*subject_id*, *protein_name*, *model_type*, *image_location*, *observed_structures*, *relative_concentration*).

From these two relations one wants to ask the query: 'Find the localization of protein X in all brain regions where evidence of process P has been found in some animal model of PD'. This is an example of semantic mediation using a disease map. To formulate such a query, we need to construct a *join-view* over the two schemata, and pose the query against this view. However, in reality, `protein_localization.observed_structures` and `process_evidence.observed_structures` need to be *semantically* joined, i.e. if one has a record with the term '`midbrain`' and the other has a record with the term '`substantia nigra pars compacta`', then these records are joinable because '`substantia nigra pars compacta`' is an anatomical subpart of '`midbrain`' and a query looking for processes in the midbrain would be satisfied by a record containing '`substantia nigra pars compacta`'. Similarly, if the query has specified *P* as '`cell death`' but the database record has an observed process called '`reduction(diameter(of(axon,dopaminergic_neuron)))`' (see Section 2.2), they should be joinable because `reduction(diameter(of(axon,dopaminergic_neuron)))` is a phase of `neurodegeneration(of(axon,dopaminergic_neuron))`. Additionally, if the `observed_structure` is '`substantia nigra pars compacta`', then the observed process should be treated as a phase of the '`cell death`' process. However, for this domain knowledge to work, it has to become part of the integrated view definition. Since the formal disease map records or derives relationships such as anatomical substructures and process phases, the integrated view is defined over the heads of chosen rules from the disease map.

6. Discussion and conclusions

We have studied the problem of constructing disease maps in the context of the BIRN project, built around the study of human diseases. As projects like BIRN assemble data repositories that encompass multiple techniques, scales, diseases and species, we face the opportunities and problems of bringing together multiple types of information relevant to the understanding of disease, regardless of how, where and why the data were originally acquired. Using the data mediation framework under development, researchers will be able to issue queries across species, diseases and animal models to try to develop new insights into common processes and features that span conditions. It is this challenge that motivates the creation of data mediation architectures and formal approaches to disease maps as described here. However, disease maps and other ontologies are useful not only in the context of database construction and mediation, but serve as an important source of information and study object in their own right. By navigation and exploration of disease maps, the scientist can investigate relationships between concepts and develop testable hypotheses.

Ideally, disease maps are dynamic representations of evolving knowledge on a given disease and thus will be continually extended and modified. This gives rise to the important problem of how to manage change in ontologies. The problem of reconciling different evolving versions over time is similarly difficult as the problem of articulating correspondences across different contemporary ontologies. We have not addressed these issues in this paper. Initially, we are providing minimalistic version management, in which evolving statements are tagged with a version identifier to which descriptive meta-data about the version and change is linked.

We are also currently developing the necessary tools and protocols for researchers to register their experimental observations in the context of the disease maps. Registering data to the disease map is somewhat analogous to publishing a paper in a scientific journal. When researchers currently publish their results, they typically write an introduction, which provides a general overview of a field and a rationale for performing an experiment, followed by methods, results and finally a discussion of how their findings relate to the current understanding of the problem. We view the process of registering an experimental observation with the disease map to encompass the same steps. The researcher will first isolate the portion of the map that provides the context for a particular set of experiments. The methods and the experimental results will be deposited in a database. Finally, the researcher will map their results onto existing hypotheses about a disease, indicating where they support or fail to support existing knowledge. If the specific hypothesis does not exist in the disease map, then the researcher will be asked to extend the map with the appropriate concepts and relations. One important method by which researchers can weight their views of a particular theory or hypothesis is through the addition of data that either supports or refutes various portions of a disease map.

Disease maps represent one of the key challenges facing the biological community: the need to express complex biological concepts in a logically sound, machine-processable form. Although the technical difficulties are many, we believe that our disease map formalization provides a promising approach to interrogate complex findings for understanding biological systems on a grand scale.

Acknowledgements

This work is supported by NIH BIRN-CC Award No. 8P41 RR08605-08S1, NSF NPACI Neuroscience Thrust Award No. ACI 9619020, NIH Human Brain Project Award No. 5RO1DC03192-05.

References

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases*. Reading, MA: Addison-Wesley.

Artale, A., Franconi, E., Guarino, N., & Pazzi, L. (1996). Part-whole relations in object-centered systems: an overview. *Data and Knowledge Engineering*, *20*(3), 347–383.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (Eds.), (2003). *The description logic handbook—theory, implementation and applications*. Cambridge: Cambridge University Press.

Baral, C. (2003). *Knowledge representation, reasoning and declarative problem solving*. Cambridge: Cambridge University Press.

Beal, M. (2001). Experimental models of Parkinson's disease. *Nature Reviews Neuroscience*, *2*(5), 325–334.

BIRN-CC (2003). Parkinson's disease map. http://www.nbirn.net/Projects/CoordinatingCenter/PDM/

Bondarenko, A., Dung, P., Kowalski, R., & Toni, F. (1997). An abstract, argumentation—theoretic approach to default reasoning. *Artificial Intelligence*, *93*(4), 63–101.

Bota, M (2001). *Neural homologies: Principles, databases and modeling*. Unpublished doctoral dissertation, University of Southern California. (http://brancusi.usc.edu/thesis_bota.pdf)

Bota, M., & Arbib, M. A. (2001). The neurohomology database. *Neurocomputing*, *38–40*, 1627–1631.

Brewka, G., & Dix, J. (2003). *Knowledge representation with logic programs* (2nd ed) (*Vol. 6*). *Handbook of philosophical logic*, Oxford: Oxford University Press.

Buchanan, B. G., & Shortliffe, H. (1984). *Rule-based expert systems: The MYCIN experiments of the Stanford heuristic programming project*. Reading, MA: Addison-Wesley.

Chen, W., Kifer, M., & Warren, D. S. (1993). HILOG: A foundation for higher-order logic programming. *Journal of Logic Programming*, *15*(3), 187–230.

Dix, J. (2003). Semantics of logic programs: their intuitions and formal properties. An overview. In A. Fuhrmann, & H. Rott (Eds.), *Logic, action and information—Essays on logic in philosophy and artificial intelligence* (pp. 241–327). Berlin: De Gruyter.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, *77*, 321–357.

Gardner, D., Xiao, Y., Abato, M., Knuth, K., & Gardner, E. (2002). *BrainML and GENIE: neuroinformatics schemas for neuroscience data sharing* (*Vol. 28*). *Society for Neuroscience Abstracts*.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In R. A. Kowalski, & K. Bowen (Eds.), (pp. 1070–1080). *Proceedings of the Fifth International Conference on Logic Programming*, Cambridge, MA: The MIT Press.

Gene Ontology Consortium (2002). Creating the gene ontology resource: design and implementation. *Genome Research*, *11*, 1425–1433.

Goedert, M. (2001). Alpha-synuclein and neurodegenerative diseases. *Nature Reviews Neuroscience*, *2*, 492–501.

Gruber, T. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, *5*(2), 199–220.

Guarino, N., & Giaretta, P. (1995). Ontologies and knowledge bases: towards a terminological clarification. In N. Mars (Ed.), *Towards very large knowledge bases* (pp. 25–32). Amsterdam: IOS Press, http://ontology.ip.rm.cnr.it/Papers/KBKS95.pdf.

Guninger, M., & Lee, J. (2002). Ontology applications and design. *Communications of the ACM*, *45*(2), 39–41.

Gupta, A., Ludäscher, B., & Martone, M. E. (2000). *Knowledge-based integration of neuroscience data sources*. *12th International Conference on Scientific and Statistical Database Management (SSDBM)*, Berlin: IEEE Computer Society, pp. 39–52.

Jagadish, H. V., Al-Khalifa, S., Chapman, A., Lakshmanan, L. V. S., Nierman, A., Paparizos, S., Patel, J. M., Srivastava, D., Wiwatwattana, N., Wu, Y., & Yu, C. (2002). Timber: a native xml database. *The VLDB Journal*, *11*(4), 274–291.

Kahle, P. J., Haass, C., Kretzschmar, H. A., & Neumann, M. (2002). Structure/function of α-synuclein in health and disease: rational development of animal models for parkinson's and related diseases. *Journal of Neurochemistry*, *82*, 449–457.

Karp, P. D. (2000). An ontology for biological function based on molecular interactions. *Bioinformatics*, *16*, 269–285.

Karp, P., Ouzounis, P., & Paley, S. (1996). *HinCyc: a knowledge base of the complete genome and metabolic pathways of H. influenzae* (*Vol. 4*). *Fourth International Conference on Intelligent Systems for Molecular Biology*, pp. 116–124.

Karp, P., Riley, M., Paley, S., Pellegrini-Toole, A., & Krummenacker, M. (1999). EcoCyc: encyclopedia of the *Escherichia coli* genes and metabolism. *Nucleic Acids Research*, *27*, 55–58.

Karp, P., Riley, M., Saier, M., Paulsen, I., & Pellegrini-Toole, A. (2000). The ecocyc and metacyc databases. *Nucleic Acids Research*, *28*, 56–59.

Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, *42*(4), 741–843.

Ludäscher, B., Gupta, A., & Martone, M. E. (2001). *Model-based mediation with domain maps*. *17th International Conference on Data Engineering (ICDE)*, Heidelberg, Germany: IEEE Computer Society, pp. 81–90.

Ludäscher, B., Gupta, A., & Martone, M. E. (2003). A model-based mediator system for scientific data management. In T. Critchlow, & Z. Lacroix (Eds.), *Bioinformatics: Managing scientific data*. Los Altos: Morgan Kaufmann.

National Library of Medicine (2003). *UMLS documentation*. http://www.nlm.nih.gov/research/umls/UMLSDOC_2003AA.pdf

Rector, A. (2003). Description logics in medical informatics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, & P. Patel-Schneider (Eds.), *The description logic handbook—theory, implementation and applications*. Cambridge: Cambridge University Press.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, *13*(1/2), 81–132.

Sagonas, K., Swift, T., & Warren, D. (1994). *XSB as an efficient deductive database engine*. *ACM SIGMOD International Conference on Management of Data, San Jose, CA*, pp. 442–453.

Stevens, R., Goble, C., Paton, N., Bechhofer, S., Ng, G., Baker, P., & Brass, A. (1999). *Complex query formulation over diverse information sources in TAMBIS*. *In Workshop on Computation of Biochemical Pathways and Genetic Networks*, European Media Lab (EML), pp. 83–88.

Swanson, L. W. (1998). *Brain maps: Structure of the rat brain* (2nd ed). Amsterdam: Elsevier.

Van Gelder, A., Ross, K., & Schlipf, J. S. (1991). The well-founded semantics for general logic programs. *Journal of the ACM*, *38*(3), 620–650.

Williams, J., & Andersen, W. (2003). Bringing ontology to the gene ontology. *Comparative Functional Genomics*, *4*, 90–93.