Journal of Biomedical Informatics 42 (2009) 540–549

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin

Content-specific auditing of a large scale anatomy ontology

Ira J. Kalet [a,b,c,d,*], Jose L.V. Mejino [d], Vania Wang [c], Mark Whipple [e], James F. Brinkley [b,d,c]

[a] Department of Radiation Oncology, University of Washington, Box 356043, Seattle, WA 98195-6043, USA
[b] Department of Medical Education and Biomedical Informatics, University of Washington, Seattle, WA 98195-6043, USA
[c] Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-6043, USA
[d] Department of Biological Structure, University of Washington, Seattle, WA 98195-6043, USA
[e] Department of Otolaryngology, University of Washington, Seattle, WA 98195-6043, USA

ARTICLE INFO

Article history:
Received 16 June 2008
Available online 25 February 2009

Keywords:
Ontologies
Knowledge modeling
Anatomy
Constraints
Auditing

ABSTRACT

Biomedical ontologies are envisioned to be useable in a range of research and clinical applications. The requirements for such uses include formal consistency, adequacy of coverage, and possibly other domain specific constraints. In this report we describe a case study that illustrates how application specific requirements may be used to identify modeling problems as well as data entry errors in ontology building and evolution. We have begun a project to use the UW Foundational Model of Anatomy (FMA) in a clinical application in radiation therapy planning. This application focuses mainly (but not exclusively) on the representation of the lymphatic system in the FMA, in order to predict the spread of tumor cells to regional metastatic sites. This application requires that the downstream relations associated with lymphatic system components must only be to other lymphatic chains or vessels, must be at the appropriate level of granularity, and that every path through the lymphatic system must terminate at one of the two well known trunks of the lymphatic system. It is possible through a programmable query interface to the FMA to write small programs that systematically audit the FMA for compliance with these constraints. We report on the design of some of these programs, and the results we obtained by applying them to the lymphatic system. The algorithms and approach are generalizable to other network organ systems in the FMA such as arteries and veins. In addition to illustrating exact constraint checking methods, this work illustrates how the details of an application may reflect back a requirement to revise the design of the ontology itself.

1. Introduction

At the University of Washington we have been engaged in exploration of how anatomic knowledge can be formalized so that automated reasoning about anatomy can be an aid in planning radiation therapy for cancer [1–3], known as the Clinical Target Volume (CTV) project. The motivation for the CTV project is the use of the FMA to better inform the design of treatment target volumes for radiation therapy. This problem remains a difficult one despite efforts to define standard nodal regions [4,5]. A related approach to decision making for nodal treatment uses Bayesian modeling [6]. However, the data for coverage at the level of detail for radiation therapy is large. Modern radiation therapy techniques such as Intensity Modulated Radiation Therapy (IMRT) demand much higher precision in defining the target volume than in the past. The variety of tumor sites, the sheer number of potentially involved nodes and the difficulty of compiling clinical data to guide these decisions make it very reasonable to consider leveraging an anatomy ontology and the power of reasoning that it supports. Building a customised set of rules or guidelines for this application has not been successful.

We have established the feasibility of modeling the spread of tumor cells through the lymphatic system using the logical network model of the lymphatics contained in the UW Foundational Model of Anatomy (FMA). This modeling is possible because the FMA is not just a terminology system or simple taxonomy; it contains a rich set of assertions about relationships between anatomic entities. The FMA also is designed to satisfy design principles (which are sometimes referred to as "axioms"). These relationships and principles are the key to clinical application of the FMA. We believe other clinical applications will be possible as well as the specific application to radiation therapy.

Since we are depending on assertions about the connectivity of the lymphatic system in doing clinical reasoning, it is critical that the system satisfies several requirements. First and foremost, it must accurately represent the actual topology of the lymphatic system of the human body. The curation of this knowledge must be done by expert anatomists, but automated procedures can

* Corresponding author. Address: Department of Radiation Oncology, University of Washington, Box 356043, Seattle, WA 98195-6043, USA. Fax: +1 206 598 3786.
E-mail address: ikalet@u.washington.edu (I.J. Kalet).

1532-0464/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2009.02.006

materially assist this process. The topology of the lymphatic system must be traversable through the "upstream" and "downstream" relations. These relations must satisfy a closure constraint, in which they are connected only to entities of the same types. Moreover, at certain levels, the paths implied by these relations must terminate only at certain locations and no others. Fortunately, the constraints that we have identified so far do in fact lend themselves to automated auditing. The constraints we checked are specific to networks connected by "upstream" and "downstream" relations, which we use in our tumor spread model. They can be applied to other networks in the human body besides the lymphatics, such as the arteries, veins and nerves. Some algorithmic parts of our auditing process may be generalizable, but our most important conclusion is that the formal model itself may need revision based on specific application needs, even without adding any new content. Such insights will be difficult to generalize, but this case study illuminates their nature, and should serve as an alert to people undertaking real clinical applications.

We describe here the constraints we needed to check, our methods and our results for the lymphatic system. We also discuss how these methods would also be applied to other networks such as arteries and veins. Our experience with formulating and applying such domain specific auditing should provide useful insights for others to formulate domain specific constraints for other large scale ontologies. We include the actual code for the constraint checking algorithms (all this work was done using Common Lisp), to give precision and concreteness to the report. We did not build a software tool with an elaborate user interface, but just used the interactive Common Lisp environment to run queries, get the results and analyze them. If indeed these methods can be generalized or suitably parametrized it may be worth engineering such a software tool. The goal of this article is not to advertise such a tool, but to provide enough detail that anyone could reproduce our results and use the algorithms with little or no modification. The code is altogether only a few pages, so this is very practical.

We begin by describing the most relevant aspects of the FMA, and the specific constraints of interest for the lymphatic system, followed by an outline of the methods, and actual code that implements the constraint checking algorithm. We report the results for the lymphatic system, and discuss its implications for updating the FMA, as well as indicating how the same methods can be applied to other network organ systems in the FMA.

2. The UW Foundational Model of Anatomy (FMA)

The University of Washington Foundational Model of Anatomy (FMA) is a formal model of human anatomy. According to its creators [7–9], the FMA is intended to be a sound, accurate and consistent formal theory of human anatomy. The FMA had its beginnings in the Digital Anatomist project at the University of Washington, a project to create a large organized collection of anatomical images and textual information to enhance the teaching of anatomy to medical students. The idea was to link a hypertext (originally implemented with Macintosh Hypercard) and an image repository. The images included annotations, labeling, and overlays of contours or other drawings delineating objects in the images. Later, these facilities became network accessible, and many offshoot projects developed. The Digital Anatomist terminology system, a semantic network that expressed many different kinds of relationships among the concepts named by the terms, gained an identity independent of the Digital Anatomist image atlases as an effort to enhance the anatomical content of the UMLS [7,10].

Our initial ideas for using the FMA to guide radiation therapy planning [1] were focused on using the hyperlinked terminology network as an index into the Digital Anatomist image atlases. Information in a radiation treatment plan could be used to retrieve relevant annotated images to help the radiation oncologist identify lymph nodes and other structures difficult to see on typical patient image sets. However, we determined that deeper automated reasoning was possible. We began a project to create a theory of tumor dissemination [2,3] and evolved the image matching problem into a separate project [11,12]. The auditing we describe here supports the tumor dissemination theory development.

2.1. The components of the FMA

The FMA represents four distinct kinds of knowledge in relation to anatomy. As is typical of most ontological modeling efforts, the FMA includes a *taxonomy* of anatomical entities. This is implemented as a class hierarchy, the Anatomical Taxonomy, or **AT**. However, anatomy is not simply about classification of entity types, but about relationships between entities. In the case of anatomy, these are *structural* relationships, so the second component of the FMA is a large collection of structural relationships. These structural relationships express such ideas as containment, constituent parts, connectivity etc. The FMA describes *canonical* anatomy, i.e., the relationships are generally true of human anatomic structure, not only a particular set of attributes of a particular human being. This part of the FMA is called the Anatomical Structural Abstraction, or **ASA**. Together with the **AT**, these relationships provide the basis for anatomical reasoning in support of applications. The FMA also is designed to accommodate things that change with time, in order to describe embryological development. This part of the FMA, called the Anatomical Transformation Abstraction, or **ATA**, is much more a work in progress than the **AT** and the **ASA**.

A fourth component of the FMA is the metaknowledge that consists of the rules and principles that the other components are required to follow. These are anatomical *axioms*.

The FMA is realized as a collection of frames. It is built and maintained using the Protégé ontology (frame) editor [13,14]. The FMA can also be translated into a description logic language [15]. It is therefore a computable representation, i.e., we can write programs to search for frames, and trace the various relationships, such `subclass` and `part-of` relationships. Even more important are the relations that implement the notion of connectivity, i.e., the *downstream* relations for the vessels of the various networks in the human body.

2.2. Representing anatomical relations in the FMA

The FMA includes entries for every element of human anatomy from the body, progressing in levels of detail down to the cellular and subcellular levels. Most important, the FMA represents *relations* between entities, not only in terms of a `superclasses`, or subsumption, hierarchy (class–subclass relationships), but also other relationships such as composition (various `part-of` relations), spatial relations and connectivity (e.g., for the blood vessels and the lymphatic systems, upstream and downstream connectivity). In addition to the general relationships already mentioned, there are specialized relationships that apply only to certain subclasses of anatomical entities. The upstream and downstream relations hold between entities of the same type, but there are also relations between different types. For example, *arterial supply*, *venous drainage*, and *lymphatic drainage* are relationships between types of vessels and types of organs. In all, there are nearly 200 relationship types in the FMA.

Because the FMA is intended to be an artifact embodying a theory of anatomy, it is designed to conform to some key principles. One of these is that the type hierarchy as well as the other relationships must accurately represent the actual experimentally verifiable structure of the human body and its parts. Another is to

distinguish, consistent with more general ontology projects such as the Basic Formal Ontology (BFO) [16], between continuants, things that exist at particular moments in time, are bounded in space, and continue to exist indefinitely, and occurrents, which may not be localised in space but are bounded in time. The latter are the things we call "processes," "events" or "changes." The references cited at the beginning of this section provide a formal description of the organizing ideas of the FMA.

3. Specifying and auditing relations and paths

Because of the sheer size of biomedical ontologies, there is considerable motivation to find ways to do automated auditing. Several kinds of auditing are possible. First and foremost, for an ontology built on a formal logic framework, e.g., a description logic, it may be possible to test for formal contradictions [15]. Insofar as a given ontology is intended to satisfy some set of axioms, it may be possible to check for consistency with those axioms. The axioms may be relatively generic, or they may be very domain specific.

An example of a more generic axiom is the requirement that part-whole relationships are transitive but not circular. A more specific one might be the restriction of slot values to certain types. This latter requirement can often be enforced by the frame system itself, if the model supports it. Finally, some constraints are really content-specific and cannot be checked or enforced by the frame system, but can possibly be automated by writing suitable procedures. An example in our project is the requirement that all paths through the lymphatic drainage system must terminate at one of two possible locations, and nowhere else. Another is that certain anatomic entities should have values for the "lymphatic drainage" property, but which entities these are is difficult to specify in general. The FMA introduces this property at what was originally thought to be an appropriate place. However, it is not simply a matter of applicability to a single class subtree or a set of subtrees. At some level of granularity the property should not be present. This is an unsolved modeling problem.

In the FMA, we performed type checking on one of the relations, the "efferent to" relation, specifically for the lymphatic system. We also did some preliminary investigation of path termination and circularity constraints for the lymphatic system. All these methods are applicable to other networks in the FMA, notably the arteries, veins and nerves.

3.1. The lymphatic system: an example

The human body has two lymphatic trees, which together provide a drainage system to bring interstitial body fluids into the blood stream. Material contained in the lymph fluid can then be filtered by the kidneys, processed by enzymes, or broken down by the action of the immune system. The lymphatic system is distributed like the arterial and venous systems throughout the body, and consists of tubes which join together to make larger vessels, and which collects and conducts fluid to one of two trunks, the Thoracic duct or the Right lymphatic duct. These in turn connect respectively to the left and right brachiocephalic veins. Fig. 1 shows a part of one such tree, as specified by the "regional part" relation.

A lymphatic chain is a subdivision of a lymphatic tree. The tree consists of a trunk and a subtree. The subtree in turn has segments, which form branches, and is also described in terms of smaller subtrees. The segments may be lymphatic chains (with lymph nodes) or lymphatic vessels. Although Fig. 1 correctly shows the tree structure, it is not easy to discern or derive paths from it.

For many anatomical entities it makes sense to define their lymphatic drainage, i.e., the ends of the lymphatic system that are

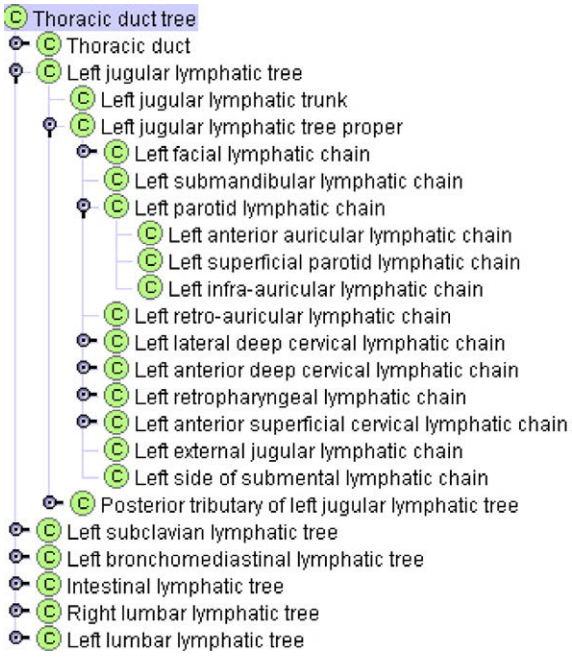

Fig. 1. A Protégé class browser display showing the regional parts of a lymphatic tree, a subdivision of the lymphatic system.

embedded in these entities and thus collect fluid from that area. Not all anatomical structures have a lymphatic drainage, so this slot is not introduced in the class, "Anatomical structure." Instead, the "lymphatic drainage" slot is introduced in each of several subclasses, including "Organ," "Organ system subdivision," "Cardinal organ part," and some others. On the other hand, cells and biological macromolecules are also anatomical structures but clearly they do not have lymphatic drainage properties.

Two relationships will be particularly central to our consistency checking in the lymphatic system. They are the *efferent to* and *afferent to* relationships. These relations only apply to components of the lymphatic system. If entity X is efferent to entity Y, that means that Y is (in the lymphatic system) *downstream* from X. Fig. 2 illustrates these relations in a schematic drawing of head and neck anatomy. The arrows show the direction of flow of lymphatic fluid. In this diagram, A is efferent to D, while B and C are efferent to A. In the FMA, where applicable, for each lymphatic chain or lymphatic vessel, the ones that are next to it going downstream, are listed in the *efferent to* slot, so A will be listed in the *efferent to* slot of B, and also in the *efferent to* slot of C. Node D will be listed in the *efferent to* slot of A. Thus, a particular chain has in its *efferent to* slot all those nodes and chains which *it* is efferent to. These are the downstream nodes and chains. The *afferent to* relationship is just the opposite, so that if A is afferent to B, that means B is upstream from A, or closer to the organ from which the lymphatics are draining. This means that B will appear in the *afferent to* slot of A.

For those entities that have a lymphatic drainage, those lymphatic chains or networks are starting points for path tracing. Starting with an organ or organ part that has some element of the lymphatic system as its drainage, asking what is each element efferent to will produce the next downstream links in the pathways through the lymphatic system. If the representation of the lymphatic system is complete and consistent, following the "efferent to" relation from any instance of a lymphatic drainage should eventually end with either the Thoracic duct or the Right lymphatic duct. An automatic path generating function should be able to identify lymphatic paths that are consistent with this requirement, and ones that are not. Several kinds of problems could occur. One is that a path reaches a dead end, because a lymphatic chain has not

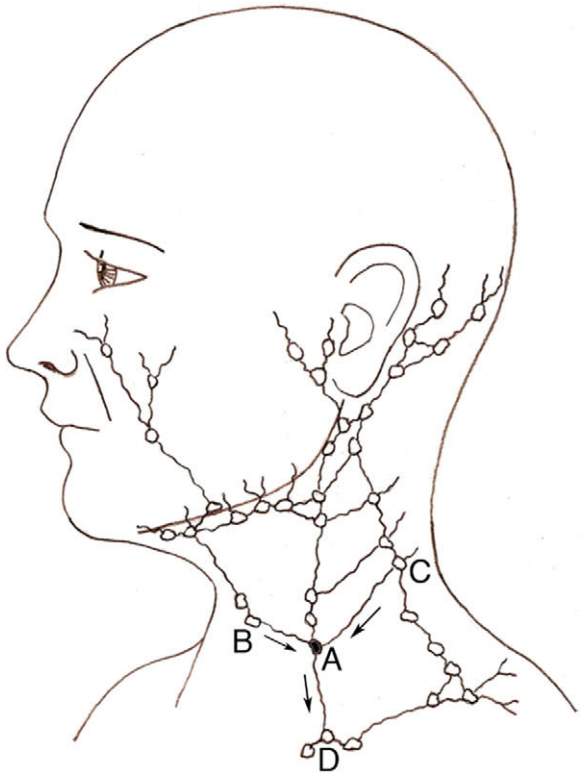

Fig. 2. A diagram showing some of the lymphatic chains and nodes in the head and neck region, illustrating the "efferent to" and "afferent to" relations. In this diagram, node A is efferent to node D, and is afferent to nodes B and C. The arrows just show the direction of flow of lymphatic fluid.

had an "efferent to" relation entered for it. Another is that a loop is found, i.e., an "efferent to" relation leads to a lymphatic chain that is upstream in the same path. A third possibility is that the type of entity that appears in the "efferent to" slot is not a lymphatic chain or lymphatic vessel. This third constraint should be enforceable by the frame system at the time of data entry, but in fact in the FMA there are several different partitionings of the lymphatic system, with some overlap. Moreover, the model (including the constraints on slots) has evolved over time. Therefore periodic auditing has value in identifying changes needed due to these modeling decisions.

We focused on the lymphatic system because it is needed for the radiation oncology application mentioned earlier. Two specific requirements for consistency and completeness in the lymphatic system are:

(1) Every lymphatic chain should have "efferent to" relations only to other lymphatic chains or to lymphatic vessels such as the Thoracic duct or the Right lymphatic duct.
(2) Starting with any lymphatic drainage from an organ or organ part, all paths using the "efferent to" relation should end up at either the Thoracic duct or the Right lymphatic duct.

In Section 4 we describe the methods, algorithms and actual code to perform these audits. The results, described in Section 5, were surprising in that we uncovered modeling problems in addition to data entry errors.

3.2. Generalization to other hierarchies

Arteries and veins form branching networks similar to the lymphatic system. In terms of flow of fluid (blood), the arteries start with a main trunk, the aorta, which subdivides into branches, with further subdivisions, until finally the arterioles (small arteries) branch into capillaries that serve particular organs or organ parts. The veins begin with the capillaries, where the blood flows into larger vessels, the venules, and these tributaries combine into yet larger vessels, the veins, much like the tributaries of a river, until finally all the flow converges into the vena cava, and then into the heart (right atrium). This describes the circulation throughout the body, but does not include the pulmonary circulation, which is separate. The blood leaves the heart via the right ventricle, travels through the pulmonary arteries and veins, returning to the heart at the left atrium, and then is pumped out again to the rest of the body through the left ventricle, into the aorta.

For veins, the flow downstream connects similarly to the lymphatics. For our purposes the only difference is the set of anatomic entities we are examining (the class "Vein" and its subclasses), and the name of the downstream relation. The relation for veins corresponding to the "efferent to" relation for lymphatics is the "tributary of" relation. If vein A is a tributary of vein B, that means that vein B is downstream from vein A. So whatever logic we apply to lymphatics should carry over to the veins.

For arteries, the constraints apply to the *upstream* relation, since the branching is outward rather than inward. So, starting from a capillary bed, tracing backward (upstream) we should only find other arteries (or subclasses), and eventually all upstream paths should terminate at the aorta. For arteries, the upstream relation we need is the "branch of" relation. If artery A is a branch of artery B, then B is upstream from A.

We have not yet applied our audit algorithms to the arteries and veins. There are many more arteries and veins than there are lymphatic chains and vessels, and we expect that there are also many more incomplete entries. Future work will include this extension of our code.

4. Methods

In this section we describe the algorithms and actual code we used to perform the audits described previously. This should serve to illustrate how one might do semantic consistency checking in an ontology like the FMA.

Most people interact with a biomedical ontology through some kind of editor or browser software. For the FMA, since it was created and is maintained as a Protégé project, it is reasonable to use Protégé itself to examine entries manually. Another option is a web interface called the Foundational Model Explorer.[1] Similarly, the Gene Ontology [17] has web based browser interface tools, as do the Kyoto Encyclopedia of Genes and Genomes (KEGG) [18], and many others.

For our purposes, an application programming interface (API) is needed, since we want to automate the process of examining frames and their contents. There have been some efforts toward creating standard APIs for knowledge bases, such as the Open Knowledge Base Connectivity (OKBC) project [19]. However, OKBC never really caught on. Other approaches include being able to export and import ontologies in an XML format, which requires considerable agreement on an actual ontology description language. At present, there are several successful versions of this, including the Web Ontology Language (OWL) [20], and the Ontology Inference Layer (OIL) language [21], which originated from the DARPA Knowledge Sharing Project. The UW Structural Informatics Group has developed several query interfaces for the FMA. One is an XML based query language, OQAFMA, tailored specifically to the FMA [22]. More recently, the FMA project has developed a SparQL[2] web service interface to an OWL version of the

[1] http://sig.biostr.washington.edu/projects/fm/FME/index.html.
[2] SparQL is a query language for RDF, which in turn is the underlying framework for OIL. It is described in documents at the W3C web site, for example, URL: http://www.w3.org/TR/rdf-sparql-query/.

FMA [23]. Of particular use to this project is a network socket based API, called the Foundational Model Server, or FMS [24].

4.1. A Simple Network Interface—the FMS

The FMS API resembles OKBC, in that it provides simple and direct Lisp-like syntax and semantics for retrieving the value of a slot in a frame, a list of available slots, and a number of other useful functions. A query is expressed as a list (in parentheses) containing a command symbol and zero or more parameters, typically strings naming entities and/or relationships. The result returned by the FMS is also a string in a Lisp-like format. This cycle repeats until the client program sends a "quit" request. This API has been functional for over 10 years, and is unsurpassed in its simplicity, so it is really easy to write a client program to query the FMS. Here we describe how it looks. Chapter 6 of [25] provides the entire code to for accessing the FMS from a Common Lisp program.

An FMS query may have parameters that are strings, numbers, `t`, or `nil`. Here is an example query, asking for the constituent parts of the heart (constitutional parts are those that make up an anatomical entity, while regional parts are spatial subdivisions),

```
(fms-get-children ‘‘Heart" ‘‘constitutional part")
```

and here is the response from the FMS:

```
("Coronary sinus" "Interatrial septum" "Interventricular septum"
"Atrioventricular septum" "Tricuspid valve" "Mitral valve"
"Aortic valve" "Pulmonary valve" "Wall of heart"
"Cavity of right ventricle" "Cavity of left atrium"
"Cavity of left ventricle" "Fibrous skeleton of heart"
"Cavity of right atrium" "Cardiac vein" "Right coronary artery"
"Left coronary artery" "Systemic capillary bed of heart"
"Lymphatic capillary bed of heart" "Neural network of heart")
```

The query above is not a function call in a Lisp program. It is a string of characters sent to the FMS through a network connection. Similarly, the return value is a long string (without newline characters) that is read back from the socket. However, it is easy to construct strings like this in a Lisp program, and even easier to parse the result string (the Lisp built-in function, `read`, does this). In order to perform this operation we need a way to make a connection to a remote server like the FMS. We use standard network socket library code to do this. The details are described in Section 6.2 of [25].

The kinds of relations in the FMA that will be needed for the audits we described are "lymphatic drainage," and "efferent to." Section 6.2 of [25] describes how to encapsulate the FMS protocol in a Lisp function, `get-children`, so that we can query about any relation from the many hierarchies in the FMA. The input to this function will be the term naming the entity of interest and the name of the relation. For example, to obtain a list of the names of the lymphatic chains that drain the Soft palate, we would perform the following query in a Lisp environment either interactively or from within another function:

```
>(get-children "Soft palate" "lymphatic drainage")
("Superior deep lateral cervical lymphatic chain"
"Right retropharyngeal lymphatic chain"
"Left retropharyngeal lymphatic chain")
```

The result is a list of strings that are the names of the entities satisfying the query. With this interface one can also write a query that obtains implementation details of the Protégé built-in relations as well. So, for example, one can get a list of all the FMA entities whose direct type is lymphatic chain, by querying as follows:

```
>(get-children "Lymphatic chain":DIRECT-INSTANCES")
("Pulmonary lymphatic chain"
"Subdivision of pulmonary lymphatic chain"
"Axillary lymphatic chain"
"Subdivision of axillary lymphatic tree"
"Posterior mediastinal lymphatic chain"
"Tracheobronchial lymphatic chain"
"Tributary of tracheobronchial lymphatic chain"
"Left cardiac tributary of tracheobronchial lymphatic chain"
...)
```

Altogether there are currently 101 direct instances of the class, "Lymphatic chain." However, this is not the entire collection of everything that is a lymphatic chain, since there are more that are classified under "Subdivision of lymphatic chain." This was done to help authors who are adding entries to the FMA to keep track of things. In this case they wanted to keep the subchains (subdivisions, or regional parts) of bigger chains in a class or category separate from the long list of individual chains, some of whose subchains are not yet identified and entered. In addition, many lymphatic chains whose type is included in the direct types above, actually appear in the body as a left variant and a right variant. These too are implemented as subclasses. Therefore in order to retrieve all the lymphatic chain entries, it is necessary to traverse the class hierarchy. To do this, we used the Protégé `:DIRECT-SUBCLASSES` relation. This is a straightforward recursive query, collecting all the subclasses below a certain point in a class hierarchy. The direct subclasses can be obtained by using `get-children`. To collect all, that list must be merged with the results of a recursive call to obtain all the subclasses of each subclass. Here is such a recursive query.

A glossary of the most important Common Lisp terms used in this paper can be found in Appendix A.

```
(defun all-subclasses (entry)
  (let ((subs (get-children entry
                            ‘‘:DIRECT-SUBCLASSES")))
    (if (null subs) nil
      (append subs (apply #’append
                  (mapcar #’all-subclasses subs))))))
```

The recursive query terminates when there are no further subclasses. The recursive call is applied to each of the subclasses (using `mapcar`), and the results are all combined into a single flat list using `append`. The `append` function is used twice, once to combine all the lists from the recursive call, and again to combine that list with the direct subclasses. Applying this to the class "Lymphatic chain" shows that there are 353 lymphatic chain entries altogether.

```
> (length (all-subclasses "Lymphatic chain"))
353
```

We also queried the FMA for lymphatic vessels, which are classified separately from lymphatic chains. The FMA currently contains 670 entries of type "Lymphatic vessel." If there are any other chains or vessels, they are misclassified. Auditing to discover such misclassifications is yet another problem, which we did not investigate.

Although the term "instance" usually refers to an individual, an entity in the real world, in the FMA extensive use is made of

metaclasses, where a class technically functions as an instance of a metaclass. Normally this relation is distinct from the "subclass" relation. It may seem that we have mixed up "instance" and "subclass" relations. In the FMA, however, every entity is indeed both a subclass and an instance of its parent class. This unconventional modeling scheme is needed in order to be able to propagate the presence of properties by inheritance, while also giving values to properties at each level [8,26].

4.2. Methods for collecting and checking relations

The first requirement we stated is a slightly more stringent requirement than the current FMA specifies in the "efferent to" relation, but it is necessary in order to get meaningful pathways. In the FMA specification of the "efferent to" relation, the following are allowed classes:

- Lymphatic chain
- Lymphatic vessel
- Anodal lymphatic tree
- Lymphatic plexus
- Lymph node

The Thoracic duct and Right lymphatic duct are lymphatic trunks, which are subclasses of Lymphatic vessel. They should be allowed as "efferent to" contents, along with other trunks. However, being the terminal entries of the lymphatic system, they do not have any "efferent to" contents. Anodal lymphatic trees are small networks of lymphatic vessels that directly drain structures. There should be no lymphatic chains efferent to these entities, as they are leaf nodes or beginning elements of lymphatic pathways. Allowing for these exceptions, the essential check is that the "efferent to" relation is closed over the set of lymphatic chains and vessels. Once we have a complete set, the check consists of iterating over the set, retrieving the "efferent to" slot contents, and classifying each chain or vessel into one of three categories. If the "efferent to" slot contents meet the constraints, it is classified as "good," if there are any invalid contents, "bad," and if no contents, "not done." This can be done in a single pass through the list, but unfortunately with lots of lookups, depending on how many entries are in the "efferent to" slot.

We checked the contents of the "efferent to" relation for lymphatic vessels as well as lymphatic chains. For each of the items in the two lists, we retrieved the values in the "efferent to" slot. This can be easily done by writing a function to go through each list, get the "efferent to" contents and pair it with its chain or vessel. Here is a parametrized implementation that makes clear we are performing the general check of closure of a set under some mapping function.

```
(defun check-closure (entity-list map-fn compare-
fn)
  (let (good bad not-done)
    (dolist (entry entity-list (list good bad not-
    done))
      (let ((slotvalues (funcall map-fn entry)))
        (cond ((null slotvalues) (push entry not-
        done))
              ((every #'(lambda (x)
                          (find x entity-list :test
                          compare-fn))
                  slotvalues)
          (push entry good))
        (t (push entry bad)))))))
```

In this function, for our purposes, `entity-list` is a list of the chains and vessels, `map-fn` is the `efferent-to` function (the application of the "efferent to" relation using `get-children`), and `compare-fn` is the `string-equal` function since the FMS returns lists of strings as results of queries. The same function should be useable for checking other networks in the FMA, such as arteries, veins and nerves, which also have similar closure requirements for upstream and downstream relations.

4.3. Checking lymphatic paths for proper termination

Another constraint we mentioned is that the result of tracing paths through the lymphatic system should always be either the Thoracic duct or the Right lymphatic duct. This is a case of the general problem of checking a directed acyclic graph (DAG) to determine if all paths terminate in a small number of specified endpoints. Since we have convenient functions for access and for the downstream relation, we just need a way to generate paths. This should be a straightforward application of a `path-search` function.

Search functions and algorithms are well described in standard Artificial Intelligence textbooks [27–30]. Path search is a simple extension to these. We used path search code described in Section 2.5.4 of [25]. The `path-search` function takes as input a starting location in the graph or network to be searched, a goal function that returns true when the goal is reached, an indicator of whether to stop at the first path, get some number of paths, or all the paths, a successors function that generates the next adjacent nodes in the network, an extender that takes a new node and puts it on the current path, and a merge function that controls the search strategy. Here is how we used `path-search`.

```
(defun find-all-paths (start)
  (labels ((successors (path)
             (get-children (first path) "efferent to")))
    (path-search start
                 #'(lambda (current)
                 ;; goal - stop when no more
                     (null (funcall successors
                                    current)))
                 nil;; get all the paths
                 #'successors
                 #'cons
                 #'append)))
```

The goal function is a function that checks if there are any more successor nodes, i.e., we keep going until the path simply ends. We specified `nil` for the next parameter to indicate that we want all the paths, not just one. The locally defined `successors` function queries the FMA to find the chains that are efferent to the current chain. The path extender is the Common Lisp `cons` function, so the path will be built up with the last node as the first item in the path list, going back to the start. Finally, the merge function will be `append`, which implements *depth-first* search. We expect that all the paths will terminate, but of course it is possible that the FMA mistakenly contains a path that is a cycle. In that case, the search will not terminate. We have not systematically checked for such cyclic paths, but so far have found none.

In order to apply this function to the lymphatic paths tracing problem, we needed to provide starting paths, which we obtained for a named anatomic entity by getting the contents of the "lymphatic drainage" slot. The following code applies the `find-all-paths` function to each lymphatic drainage entry and produces a list of paths for each. These lists all should be appended together to get the complete list of paths, so the same idiom that we used in the

`all-subclasses` function, with `append` and `mapcar`, can be used here.

```
(defun lymphatic-paths (site)
  (apply #'append
       (mapcar #'(lambda (start)
                   (find-all-paths start))
               (get-children site "lymphatic
               drainage"))))
```

We used this function to generate paths for a few interesting locations in the head and neck. The complete auditing of paths is a work in progress.

5. Results

This section describes in detail the results we obtained for the lymphatic system. We have not yet performed comprehensive checks of the arteries, veins or nerves.

5.1. Auditing the downstream relation in the lymphatics

The tests were performed in a sequence of steps, so that we could examine intermediate results. Of course for fully automated checking these steps could easily be combined. First, for the lymphatic chains and vessels, we obtained all the entities for these types and their subtypes, using the `all-subclasses` function previously defined. We obtained lists of both the "lymphatic chain" and the "lymphatic vessel" classes. Examples of lymphatic chain subclasses include:

- Pulmonary lymphatic chain
- Subdivision of pulmonary lymphatic chain
- Axillary lymphatic chain
- Subdivision of axillary lymphatic tree
- Posterior mediastinal lymphatic chain
- Tracheobronchial lymphatic chain
- Tributary of tracheobronchial lymphatic chain
- Left cardiac tributary of tracheobronchial lymphatic chain
- Brachiocephalic lymphatic chain
- Right cardiac tributary of brachiocephalic lymphatic chain

A person knowledgeable about anatomy may notice that these are all proper subclasses, but some refer to specific entities that are found in an actual person, e.g., Left cardiac tributary of tracheobronchial lymphatic chain, while others represent aggregate classes. For example, there are several subdivisions of the Pulmonary lymphatic chain, so the class, Subdivision of pulmonary lymphatic chain, does not refer to a specific entity. The Lymphatic vessel subclasses are similarly mixed, that is, specific entities are at the same level of description as aggregate classes. Here are some examples of those:

- Lymphatic capillary
- Tributary of lymphatic trunk
- Tributary of lymph node
- Superficial lymphatic vessel
- Deep lymphatic vessel
- Lymphatic trunk

Next, for each of the items in the two lists, we retrieved the values in the "efferent to" slot, giving the results shown in Table 1. Each lymphatic chain or vessel in the left column may have entries for chains or vessels that are downstream, shown in the right column. If the FMA contains nothing in the "efferent to" slot for that chain or vessel, the table shows an entry of NIL.

Table 1
Contents of "efferent to" slots of some lymphatic chains and lymphatic vessels.

Chain or vessel name	Contents of "efferent to" slot
Pulmonary lymphatic chain	Bronchopulmonary lymphatic chain
Subdivision of pulmonary lymphatic chain	NIL
Axillary lymphatic chain	Subclavian lymphatic trunk Subclavian lymphatic tree
Subdivision of axillary lymphatic tree	NIL
Posterior mediastinal lymphatic chain	Thoracic duct Tracheobronchial lymphatic chain
Tracheobronchial lymphatic chain	Bronchomediastinal lymphatic trunk Bronchomediastinal lymphatic tree
Tributary of tracheobronchial lymphatic chain	NIL
Left cardiac tributary of tracheobronchial lymphatic chain	NIL
Brachiocephalic lymphatic chain	Bronchomediastinal lymphatic trunk Bronchomediastinal lymphatic tree
Right cardiac tributary of brachiocephalic lymphatic chain	NIL
Lymphatic capillary	NIL
Tributary of lymphatic trunk	NIL
Superficial lymphatic vessel	NIL
Deep lymphatic vessel	NIL
Lymphatic trunk	NIL

We sorted these lists into three categories. The first category includes all the chains that have valid chains or vessels in the "efferent to" slot. There are 140 chains in this category. A sample of these is shown in Table 2. We found 11 chains with invalid data in the "efferent to" slot. A small sample is shown in Table 3. We found 202 chains with no data in the "efferent to" slot. Some of these are evident in Table 1, where NIL is shown in the contents column. Examining the 11 determined to be invalid, it is apparent that these entries are ones where a lymphatic tree has been entered. It is problematic that a chain could be efferent to a subtree of the lymphatic system, although this is anatomically correct in some sense. A chain that is *part of* a tree connects to the tree and the lymphatic fluid flows from the chain into that branch of the tree, but that is not the same as saying that the entire tree is downstream. For path tracing, it is misleading, because the flow from that chain does not go through the whole tree, but only through a subset of branches. So these need to be fixed in order to do sound path tracing. As there are so few, it is relatively easy to fix this modeling problem, and we are proceeding with these corrections.

We performed the same checks on the vessels list. The counts for the three categories were: 13 classified as valid, 2 as invalid and 655 with no contents in the "efferent to" slot. On inspection

Table 2
Some chains that have valid "efferent to" slot values.

Chain name	Slot contents
Left submental lymphatic chain	Left submandibular lymphatic chain Left jugulo-omohyoid lymphatic chain
Right submental lymphatic chain	Right submandibular lymphatic chain Right jugulo-omohyoid lymphatic chain

Table 3
Some chains that have invalid "efferent to" slot values.

Chain name	Slot contents
Left parasternal lymphatic chain	Left bronchomediastinal lymphatic tree
Right parasternal lymphatic chain	Right bronchomediastinal lymphatic tree

of the two that are flagged as bad, the same problem appears. A tree has been entered where we would expect a trunk, vessel, or chain.

5.2. Results of pathway checking in the lymphatics

The second check, to determine whether all paths terminate at the Thoracic duct or Right lymphatic duct, is more challenging. The fact that many "efferent to" relationships are still not done makes it highly likely that there are many incomplete paths, so this check is premature, and we have only sampled the paths for termination. Here we describe the results of these samples. There are two ways to proceed. First, one could put together a list of all the organs and organ parts whose lymphatic drainage should be traced. We would apply the `lymphatic-paths` function to each and examine the results for incomplete paths, in a manner similar to the checking of slot values for the "efferent to" relation. This is difficult because there is no simple way to identify all the relevant organs and organ parts that should be starting points.

Another possible approach would be to start with the list of chains and vessels. Using the `find-all-paths` function, one traces the paths, and examines their end points. This is difficult because there are many classes in this list for which such path tracing is actually inappropriate. By applying the first check to a few sample organs, it will become apparent what is going on.

Fig. 3 shows the results of using the `lymphatic-paths` function for the Soft palate. Three lymphatic chains drain the Soft palate, the Superior deep lateral cervical lymphatic chain, the Right retropharyngeal lymphatic chain, and the Left retropharyngeal lymphatic chain. There are two paths from each of the three lymphatic chains. So the lymphatic system is multiply connected, but at least so far, not cyclic. The paths from the Right retropharyngeal lymphatic chain end at the Right lymphatic duct, as expected. Similarly, the paths from the Left retropharyngeal lymphatic chain end at the Thoracic duct. However, the paths from the Superior deep lateral cervical lymphatic chain end at the Jugular lymphatic trunk. This is not useable for reasoning about the migration of tumor cells, although it is anatomically valid.

```
Right retropharyngeal lymphatic chain
--> Right superior deep lateral cervical lymphatic chain
--> Right inferior deep lateral cervical lymphatic chain
--> Right jugular lymphatic trunk
--> Right lymphatic duct

Right retropharyngeal lymphatic chain
--> Right superior deep lateral cervical lymphatic chain
--> Right jugular lymphatic trunk
--> Right lymphatic duct

Left retropharyngeal lymphatic chain
--> Left superior deep lateral cervical lymphatic chain
--> Left inferior deep lateral cervical lymphatic chain
--> Left jugular lymphatic trunk
--> Thoracic duct

Left retropharyngeal lymphatic chain
--> Left superior deep lateral cervical lymphatic chain
--> Left jugular lymphatic trunk
--> Thoracic duct

Superior deep lateral cervical lymphatic chain
--> Inferior deep lateral cervical lymphatic chain
--> Jugular lymphatic trunk

Superior deep lateral cervical lymphatic chain
--> Jugular lymphatic chain
--> Jugular lymphatic trunk
```

Fig. 3. Lymphatic paths from the soft palate.

There are actually two jugular lymphatic trunks, a Right jugular lymphatic trunk and a Left jugular lymphatic trunk. One goes to the Thoracic duct and one goes to the Right lymphatic duct. They are each a subclass of the general class, "Jugular lymphatic trunk." So it is correct to have no entry at the higher level of generality. In attempting to determine drainage paths for tumor cells, the reasoning system will have to somehow determine that the knowledge needed is not at this level. This is yet another complication in checking consistency and completeness, as well as in applying the knowledge to clinical problem solving.

There are many such places where anatomical structures are described as general classes, and then have more specific right and left subclasses. This is very important, since geography is important and (especially with radiation therapy) one must specify on which side the entities of interest are. The presence of the general classes and right/left subclasses makes reasoning difficult to automate. There is room for some further innovation here.

6. Discussion and conclusions

We have corrected the errors that were turned up in these audits. There were few enough that it was relatively easy to simply edit the entries with Protégé. One question that naturally arises is: did we find all the errors that we sought? A usual approach is to establish a "gold standard" test environment, in which the "true" errors are known, and then determine the true and false positive rates, from which sensitivity and specificity can be reported. Thus, one could envision comparing these search techniques to other methods. However, such an evaluation does not apply here. For the lymphatic system entities identified, the tests are pathognomonic, i.e., if an error is found, it is indeed an error, and further, the check for these kinds of errors is definitive. There is no possibility of a false positive or of overlooking an incorrect entry in the "downstream" slot. Every such slot contains a small number of entries, each with well defined type. Either the type is allowed or it is not, according to the consistency requirements. The idea of an independent "truth" simply does not apply. The main issue in regard to performance is in the coverage. Have we checked every lymphatic system object that needs to be checked? Here we depend on the class structure. There is a possibility that a lymphatic chain or vessel will not be checked because it is not specified to be a subclass or instance of the "Lymphatic chain" or "Lymphatic vessel" class. It is also possible that an entity is misclassified as such a class when it actually is not. Identifying misclassified lymphatic system entities is a check we did not undertake. That kind of checking would require an independent way of determining whether an entity was a lymphatic chain or not. One method might be to lexically analyse names, which would raise yet more issues.

Our checks have identified some complexities in the model itself. Some of these results suggest making changes to the model, e.g., removing lymphatic trees as allowed entries in the "efferent to" slot. This is an example of how the restrictions on slot values may be modified over time. Since restriction on slot values may change with time, the ontology editor's checking with data entry is not sufficient. As decisions about the model evolve, the entries should be retrospectively checked, so the methods we developed have an ongoing role. Such restrictions are themselves data that are entered, and also should be auditable, but it is not clear how that could be automated, since they represent modeling decisions that have no higher level formal representation in the system.

The problem of modeling classes like "Jugular lymphatic trunk" for which there are more specific subclasses is more vexing. The entries clearly satisfy the type constraints, but are problematic in

that a tumor cell migration application that tries to trace the possible paths will fail at this point in the FMA. The cells certainly do not stop at the Jugular lymphatic trunk, and it is not a terminus in the actual human lymphatic network.

In regard to the "efferent to" relation, one might think that the lymphatic system consists entirely of two directed acyclic graphs and that standard algorithms for analysis apply here. But the lymphatic system entities in the FMA form many disjoint graphs, not just two. In terms of a description of anatomy, this reflects multiple levels of abstraction that have been mixed together. Such mixing makes precise reasoning ineffective or inaccurate. In terms of application to a model of tumor cell migration, what really needs to be verified is that all paths from any instantiable organ, organ part or tissue connect to and terminate in one of the two endpoints, the Thoracic duct and the Right lymphatic duct. The difficulty in implementing this constraint is that this is not a requirement on every entry in the lymphatic system representation, but only on the ones that drain tissues. We have no way at present to identify this subset.

A solution to consider is to add a requirement to the FMA that relations such as "efferent to" need to be applied only to the most specific members of a class hierarchy. This recognizes that we have abstract classes and instantiable subclasses, in some cases. Solving this problem is a topic for further discussion, but at least the connectivity checks were able to identify instances of the problem. In general, the FMA models every entity as a class, because it is intended to model *canonical* anatomy, rather than the anatomy of a particular individual. However, we have demonstrated that reasoning applications require that we distinguish between classes that are directly instantiated in individuals, such as Left lung or Right lung, from classes that are abstract, never directly instantiated, such as Lung. Complicating this problem, we fully expect that some knowledge that is appropriately modeled at the more abstract level will *also* be needed for applications, so it does not appear that there is a simple solution to this problem.

In the few path checks that we have run, we have found that every path terminates, though as the results show, not always where it should. Another possibility is that the path check may reveal circularities. The path tracing code described previously would not terminate in this case, and a different kind of check is needed, where one keeps track of nodes already visited and flags returns to such nodes. Simply doing breadth first search would not be sufficient. A breadth first search will not terminate because our termination condition is that there are no more successor nodes. The function that generates successors must have access to and keep track of nodes already previously generated. In that case, even depth first search will terminate since there are a finite number of entries to consider. A systematic search is planned for the near future.

The implications for radiation therapy planning and our Clinical Target Volume project are that our model for the predictions of migration of tumor cells will be based on an incorrect or incomplete rendering of the lymphatics. If some chains or vessels are incorrectly represented as end points of paths, the model will predict no tumor cell migration beyond them. This will tend to suggest an inappropriately smaller radiation target volume, and the patient would be undertreated if such a plan were actually used.

At present the veins and arteries are modeled in an even more complex fashion than the lymphatics. For example, there is a class named "Vein," but it does not include such well known entities as the Superior vena cava or Inferior vena cava. These are subclasses (instances) of another class, "Venous trunk." So the collection of instances and checking of the contents of the downstream relation must take this into account. So, again it can be seen that complexities arise from a focus on class–subclass modeling. Again it is important to note that the assertions which these entries represent are true, but they may not facilitate correct reasoning because of the mixing of levels of abstraction. For applications that need structural information about veins and arteries, an additional complication is that variations among individuals may be significant, and will limit the accuracy of inference based on canonical structure.

The algorithm for actually performing the checks on slot contents appears to be independent of these contextual problems. Some of the code could be further elaborated and engineered into a suite of algorithms with suitable parametrization, that could be the basis for a useful tool. If enough applications exist, we will consider this for future work. However, the main conclusion from our investigations is not that the algorithms or code are generalizable. It is that applications may expose modeling issues that were not anticipated when constructing the ontology purely from a general knowledge representation viewpoint. We expected to find a few data entry errors, but we did not expect to find that there are modeling challenges as well. The modeling challenges are beyond the formal requirements of consistency and completeness that can be expressed generally. They are concerned with the content itself.

The main point here is not that the FMA needs to be adjusted to make it more suitable for a particular application, but that auditing to meet application requirements uncovered some real modeling issues. It is not sufficient to claim that an ontology expresses in a humanly understandable way what its authors know. To really apply an ontology to problem solving sets a higher standard, one to which the FMA aspires. To put yet another perspective to this work, similar modeling issues have been identified in the UMLS [31]. It is not too soon for those involved in ontology building projects to seek applications and delve deeply enough to discover what the implications may be for the ontology itself.

Acknowledgements

We gratefully acknowledge support from NIH Grants LM008788 from the National Library of Medicine, and HL087706 from the National Heart, Lung and Blood Institute. We also greatly appreciate the thoughtful comments of the reviewers, which helped considerably in focusing and clarifying the description of this work.

Appendix A. Common lisp

For those readers not familiar with the Lisp programming language, we include here a brief glossary. Several readily available books provide an introduction to Lisp programming [32–34]. Norvig's book on artificial intelligence programming [30] also includes in the first few chapters a very readable introduction.

All expressions in Lisp are of the form `(operator argl arg2 ...)`. The various operators, as in any programming language, each have their own details about what the required and other arguments should be. Here are short descriptions of some of the ones used in this work.

defun defines a named function that can then be used elsewhere in a program; its first argument is the name to be used, the second is a list of formal parameters, and the remainder is the code body.
let gives variables (symbols) a local context, similar to other languages in which variables can be given local scope inside a function or procedure.
append combines several lists into one list.
labels creates local function definitions, usable within a function or procedure without affecting the surrounding code.
dolist iterates over a list, executing a body of code for each element of the list.

cond is a conditional operator like `if` but it allows for many clauses instead of just two, each with its own test.
mapcar applies a function to each element of a list, producing a new list. Each element of the new list is the result of applying the function to the corresponding element of the original list.

The symbol sequence `(lambda ...)` is an expression that defines an anonymous (unnamed) function, that is usually used right where it is defined.

The sequence `#'` is a syntactic abbreviation for a call to the `function` operator, which returns the function associated with the following name or description. Thus, `#'append` means, "the function named `append`," and `#'(lambda ...)` means, "the anonymous function defined by the given expression."

Semicolons (;) introduce comments.

References

[1] Kalet IJ, Wu J, Lease M, Austin-Seymour MM, Brinkley JF, Rosse C. Anatomical information in radiation treatment planning. In: Lorenzi NM, editor. Proceedings of the american medical informatics association fall symposium. Hanley & Belfus Inc.; 1999. p. 291–5.

[2] Kalet IJ, Whipple M, Pessah S, Barker J, Austin-Seymour MM, Shapiro L. A rule-based model for local and regional tumor spread. In: Kohane IS, editor. Proceedings of the american medical informatics association fall symposium. Hanley & Belfus, Inc.; 2002. p. 360–4.

[3] Benson N, Whipple M, Kalet I. A Markov model approach to predicting regional tumor spread in the lymphatic system of the head and neck. In: Proceedings of the American Medical Informatics Association (AMIA) fall symposium. 2006; p. 31–5.

[4] Robbins KT, Medina JE, Wolfe GT, et al. Standardizing neck dissection terminology. Arch Otolaryngol—Head and Neck Surgery 1991;117:601–5 [Official report of the Academy's committee for head and neck surgery and oncology].

[5] Som PM, Curtin HD, Mancuso A. An imaging-based classification for the cervical nodes designed as an adjunt to recent clinical based nodal classification. Arch Otolaryngol—Head and Neck Surgery 1999;125:388–96.

[6] Galán SF, Aguado F, Díez FJ, Mira J. Nasonet, modeling the spread of nasopharyngeal cancer with networks of probabilistic events in discrete time. Artificial Intell Med 2002;25:247–64.

[7] Rosse C, Shapiro LG, Brinkley JF. The Digital Anatomist Foundational Model: principles for defining and structuring its concept domain. In: Proceedings of the Americal Medical Informatics Association (AMIA) fall symposium. 1998; p. 820–4.

[8] Rosse C, Mejino JL. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. Journal of Biomedical Informatics 2003;36:478–500.

[9] Rosse C, Mejino JL. The Foundational Model of Anatomy ontology. In: Burger A, Davidson D, Baldock R, editors. Anatomy ontologies for bioinformatics: principles and practice. New York: Springer-Verlag; 2008. p. 59–117.

[10] Rosse C, Mejino JL, Modayur BR, Jakobovits R, Hinshaw KP, Brinkley JF. Motivation and organizational principles for anatomical knowledge representation: the Digital Anatomist symbolic knowledge base. J Am Med Inform Assoc 1998;5:17–40.

[11] Teng CC, Austin-Seymour MM, Barker J, Kalet IJ, Shapiro L, Whipple M. Head and neck lymph node region delineation with 3D CT image registration. In: Kohane IS, editor. Proceedings of the American Medical Informatics Association fall symposium. Hanley & Belfus, Inc.; 2002. p. 767–71.

[12] Teng CC, Shapiro LG, Kalet IJ, Rutter C, Nurani R. Head and neck cancer patient similarity base on anatomical structural geometry. In: Proceedings of the 2007 IEEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, Piscataway, New Jersey, IEEE Engineering in Medicine and Biology (EMB) Society. 2007; p. 1140–43.

[13] Musen MA, Gennari JH, Eriksson H, Tu SW, Puerta A. PROTÉGÉ-II: Computer support for development of intelligent systems from libraries of components. In: Proceedings of the World Congress on Medical Informatics, MEDINFO'95. 1995; p. 766–70.

[14] Noy NF, Crubézy M, Fergerson RW, Knublauch H, Tu SW, Vendetti J, et al. Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In: Proceedings of the American Medical Informatics Association (AMIA) fall symposium. 2003; p. 953. Available from: http://protege.stanford.edu/.

[15] Beck R, Schulz S. Logic-based remodeling of the Digital Anatomist Foundational Model. In: Proceedings of the American Medical Informatics Association fall symposium. 2003; p. 71–5.

[16] Grenon P, Smith B, Goldberg L. Biodynamic ontology: applying BFO in the biomedical domain. In: Pisanelli DM, editor. Ontologies in medicine. Amsterdam: IOS Press; 2004. p. 20–38.

[17] The Gene Ontology Consortium. Gene Ontology project web site, 2008. Available from: http://www.geneontology.org/.

[18] Kanehisa Laboratories. Kegg project web site, 2008. Available from: http://www.genome.jp/kegg/.

[19] Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP. OKBC: A programmatic foundation for knowledge base interoperability. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98). 1998; p. 600–7.

[20] Antoniou G, van Harmelen F. Web ontology language: OWL. In: Staab S, Studer R, editors. Handbook on ontologies. Berlin, Heidelberg, New York: Springer-Verlag; 2004. p. 67–92.

[21] OIL project. Oil project web site, 2008. Available from: http://www.ontoknowledge.org/oil/.

[22] Mork P, Brinkley J, Rosse C. OQAFMA querying agent for the foundational model of anatomy: a prototype for providing flexible and efficient access to large semantic networks. J Biomed Inform 2003;36:501–17.

[23] Detwiler LT, Suciu D, Brinkley JF. Regular paths in SparQL: Querying the NCI thesaurus. In: Proceedings of the American Medical Informatics Association FALL Symposium. 2008; p. 161–5.

[24] Stalder D, Brinkley J. The Digital Anatomist Foundational Model Server. In: Perl Conference 3.0 Refereed Papers, Monterey, California, O'Reilly & Associates Inc.; 1999.

[25] Kalet IJ. Principles of biomedical informatics. San Diego, CA: Academic Press (Elsevier); 2008.

[26] Noy NF, Musen MA, Mejino JLV, Rosse C. Pushing the envelope: challenges in a frame-based representation of human anatomy. Data Knowledge Eng 2004;48:335–59.

[27] Russell SJ, Norvig P. Artificial intelligence: a modern approach. 2nd ed. Prentice Hall; 2003.

[28] Nilsson NJ. Artificial intelligence: a new synthesis. Morgan Kaufmann Publishers, Inc.; 1998.

[29] Tanimoto S. The elements of artificial intelligence using common lisp. 2nd ed. New York: W.H. Freeman; 1995.

[30] Norvig P. Paradigms of artificial intelligence programming: case studies in common lisp. San Mateo, California: Morgan Kaufman; 1992.

[31] Ceusters W, Smith B, Kumar A, Dhaen C. Mistakes in medical ontologies: Where do they come from and how can they be detected? In: Pisanelli DM, editor. Ontologies in medicine. Amsterdam: IOS Press; 2004. p. 145–63.

[32] Graham P. ANSI common lisp. Englewood Cliffs, New Jersey 07632: Prentice-Hall; 1996.

[33] Lamkins DB. Successful Lisp: How to Understand and Use Common Lisp.bookfix.com, 2004. This book is also available on line.

[34] Seibel P. Practical common lisp. Berkeley, California: Apress; 2005.