A Proposal for Collaborative Ontology Editor for Animal Trait Ontology

[Draft, 2005-06-04]

Jie Bao[a], Zhiliang Hu[b], James Reecy[b], and Vasant G Honavar[a]

[a]Artificial Intelligence Research Laboratory, Department of Computer Science
[b]Department of Animal Science
Iowa State University, Ames, IA 50011-1040, USA

Abstract. In this proposal, we outline the intended feature, related work, existing resources and primary design for a collaborative ontology editor, aimed for the team-work building of Animal Trait Ontology (ATO). ATO is an ontology with DAG skeleton to describe terminology of animal traits. The proposed editor supports communication with an ontology repository and concurrent editing on partial ontologies. Ontology in the repository is manipulated with packages and formalized using modular partial-order ontology languages.

1 Introduction

The need for biological ontologies has risen in recent years in large part due to the development of large biological databases. Successful genotype ontologies, such as Gene Ontology[1] and Rice Ontology[2], as well as phenotype ontologies such as Plant Ontology[3] and Phenotype And Trait Ontology (PATO)[4], have emerged in the past few years. The goal of these ontologies is to form a standardization that is both syntactically and semantically correct. Animal Trait Ontology (ATO) is an on-going project, developed by Iowa State University, Animal Science department, to formally describe animal trait on several aspects, such as meat quality, health factors, exterior features, production and reproduction parameters. The ontology has following properties:

- It has DAG (Directed Acyclic Graph) structure;
- Each term has a unique name;
- Relations between terms include is-a, part-of and possibly more;
- Each term, in additional to hierarchical information, may have other properties, for example, scale_unit for each trait value;
- The ontology is specie-dependent, each specie may need its specific ontology; however, those different specie ontologies share some common upper level structure and terms and maybe comparable when developed.

[1] http://www.geneontology.org/
[2] http://www.ro.dna.affrc.go.jp/docs/index.html
[3] http://www.plantontology.org/
[4] http://obo.sourceforge.net/cgi-bin/detail.cgi?poav

There is a need for a tool to create and manipulate the animal trait ontology. Intended features of such tool include:

- The ontology will be stored on a relational database server; the editor can connect to the server and "check-out" part of the ontology, do editing on the checked-out part, and "check it in" when finished editing.
- The editor should ensure the consistency of submitted ontology and ensure the integrity of entire ontology.
- The editor should support multiple users to concurrently edit the ontology. Mechanism should be provided to avoid conflict and abnormalities. Modules of the ontology can be developed by different parts and be assembled later on.
- There should be user profile management. Different user, such as administrator and common user, has different privileges on reading/writing.
- The editor should provide a handy user interface to edit a DAG.
- The editor can import and export ontology in standard format, such as OBO[5] and OWL[6].
- The editor should have good scalability, that can manipulate ontology of 10^5 terms (the order of Gene Ontology)

2 Related and Background Work

2.1 Related Work

Ontology building has received considerable attention during the past few years, and there are many available ontology editors, such as Protege[7], OilEd[8], OntoEdit [9] and DAG-Edit[10] . For the state-of-art of ontology editor, please see survey by OntoWeb (2002)[Pe02], Michael Denny (2002) and[Den02] (2004)[Den04], and Irene Polikoff (2003) [Pol03].

Despite of availability of such existing tools, new tool is still needed for ATO, based on the limitation of those tools

- Efficiency: ATO ontology is a hierarchy-based ontology, which demands a succinct ontology representation. However, most of the general-propose ontology editors are based on expressive knowledge representation paradigms, such as Description Logic (or its variances of DAML, OIL and OWL), Frame Logic(such as OKBC) or First Order Logic (FOL). Although such expressive languages are capable to build hierarchy, tools based on such languages are less efficient than a devoted hierarchy tool. This is also the reason why Gene Ontology using with its own tool DAG-Edit.

[5] http://www.geneontology.org/GO.format.shtml#oboflat
[6] http://www.w3.org/TR/owlfeatures/
[7] http://protege.stanford.edu/
[8] http://oiled.man.ac.uk/
[9] http://www.ontoknowledge.org/tools/ontoedit.shtml
[10] http://www.godatabase.org/dev/java/dagedit/docs/index.html

– Collaboration: Many tools, such as Protege, OilEd and DAG-Edit, lack the support for collaborative ontology building. Such tools only support single user to be active at certain moment and hardly support the development for team work.
– Scalability: Many file-based systems face saleability problem when the ontology is large. Although some systems support database storage, they need a whole in-memory model for user interaction and reasoning, such as Protege-OWL.

No tool, for the best of our knowledge, has met all the requirements for ATO application. Therefore, a new tool is needed, might with components absorbed from existing tools.

2.2 Background Tool

In the preparatory research[11], Jie Bao, Swetha Gottimukkula, LaRon Hughes, Jialin Le, and Jia Tao has developed a set of tools to create and browse ATO. They include:

– Two database servers (one on MySQL and another on PostgreSQL) to test database storage.
– A preliminary ontology editor
– A web browser for the ontology

”

The existing ATO editor has following features (refer to user manual[12] for more details)

– Edit and browse the ontology as tree;
– Create connecting to the database and remembers the configuration;
– Load all or part of the ontology into the interface;
– Modify existing terms (including changing comments, changing details), deleting terms, deleting a subtree and insert new terms;
– Find terms on the tree;
– Take records of editing actions and enable undo and redo.

System diagram of the editor is shown in figure 2 The editor is based on the INDUS ontology editor[13] [CBP+05,RCSC+03] developed by Artificial Intelligence Research Lab, Computer Science Department, Iowa State University.

There is multiple reasons to choose INDUS ontology editor as the start point, instead of using other open source projects, such as DAG-Edit or Protege, such as:

– INDUS has a very clean and concise kernel for hierarchial ontologies;

[11] http://boole.cs.iastate.edu/indus/ato/ato.html
[12] http://boole.cs.iastate.edu/indus/ato/doc/editor%20document.doc
[13] http://www.cs.iastate.edu/ dcaragea/INDUS/documentation.htm

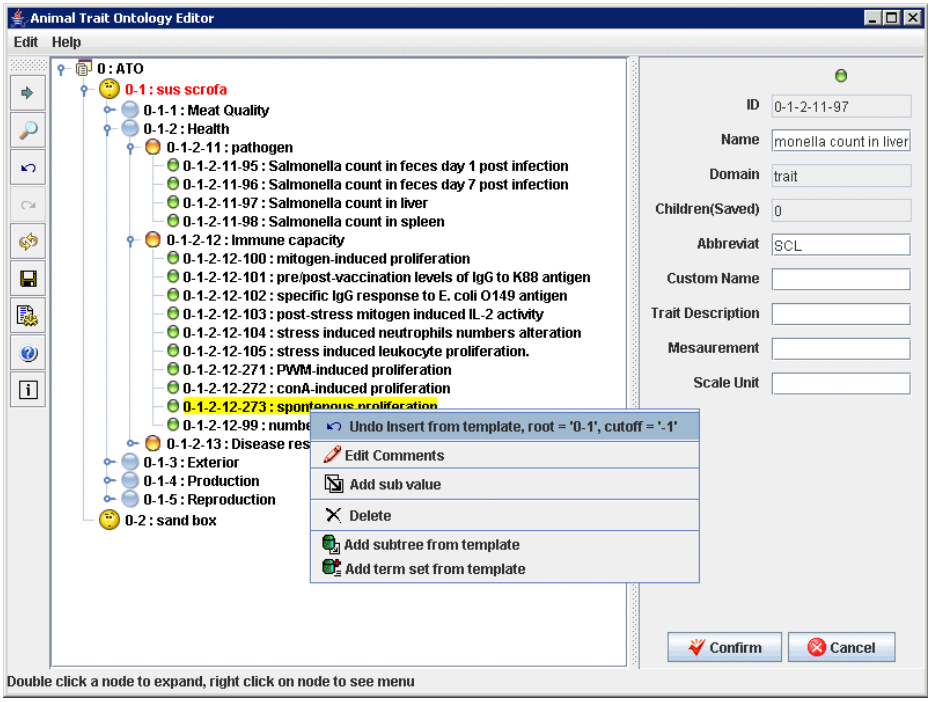

Fig. 1. The Preliminary ATO Editor

- INDUS supports database storage and partial loading of the persisted ontology, which means good scalability
- INDUS includes a reasoner for hierarchical ontologies, that could be used to check consistency.
- Another part of INDUS, such as the ontology mapping editor, data authoring editor, and query engine, could be used for further applications of ATO.

2.3 Knowledge Representation

Two knowledge representation formalism has been proposed for ontology in our previous experiences in ontology engineering and ATO is suitable for their combination.

The first one is partial-order($\leq$) based ontology language [Car04] [BCPH05]. Partial-Order ontology is the theoretic model for hierarchies, such as DAG and tree. INDUS is based on such formalism and include parser and encoder for its XML syntax.

The second formalism is Package-based Ontology[BH04b]. In such a modular ontology representation, ontology is divided into smaller components called packages, and each package contains a set of highly related terms and their relations; packages can be nested in another package, and form a package hierarchy;

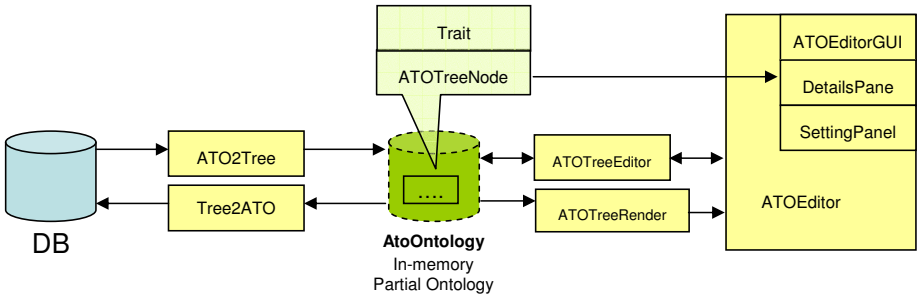

Fig. 2. Design of a infrastructure where the proposed ATO editor serves as an external tool to edit term details and relations that are stored remotely on a relational database (MySQL/Postgres)

the visibility of a term is controlled by scope limitation modifier such as `public`, `private` and `protected`.

Reasons to use package in ATO include:

- Package serves as organization unit. User can check out one or more packages for editing, while other packages are still open to other users.
- Package can be reused. For example, if pig component and horse component of ATO shares information on "production", the shared part can be organized in a package and be reused by the pig component and the horse component.
- During the editing, ontology may evolve. To reduce the risk of unwanted coupling and inconsistency diffusion, package can be used with "visible interface" (a set of terms that can be used by other packages) and "invisible" inner contents. The inner contents can be changed (for example, insert new node between exiting node, or merge two inner nodes) while the visible interface is still unchanged.
- In reasoning, package has localized semantics, thus an error in a package can be more easily controlled and traced.

Packages is the structural unit in the *design* phase; when the ontology is released, the user does not necessarily aware of packages. From the view of end user, ATO has similar structure as that of GO.

The combination of this two, packaged-based partial-order ontology, is the basic KR model for collaborative ATO editor.

3 Ontology Engineering Issues

3.1 Nomenclature

Instead of cascading naming such as '0-1-1-1-1' (abdominal fat), in the new design, each node in the DAG should have a fixed length id, like that of GO,

in the form of "ATO:1234567". It allows 10^7 terms, as the same of GO. Reason to abandon cascading naming is that terms may change their position on the ontology during the design process and fix-length naming can avoid frequent renaming. It is also compatible to naming of other OBO ontologies (such as GO).

Package name is not intended to be used by users, only by the ontology authors. It should be easy to remember, such as "pig", "pig.production", "production.common". Cascading naming is recommended for package hierarchy.

3.2 Data Storage

All information about the ontology will be stored in a relational database (also see figure 3).

- Package: to store information of packages
- Pkg_relation: to store importation and package hierarchy, such as "p1 import p2" and "p1 nested-in p2"
- Term: to store id, name, package membership and visibility
- Relation: to store relations among terms, such as "t1 isa t2","t1 part-of t2"
- Details: to store trait details, such as measurement and scale_unit.
- User: to store user profiles and privileges, such as administrator and guest.
- Authorization: to store user privilege for specific package, such as read-only and read-write.

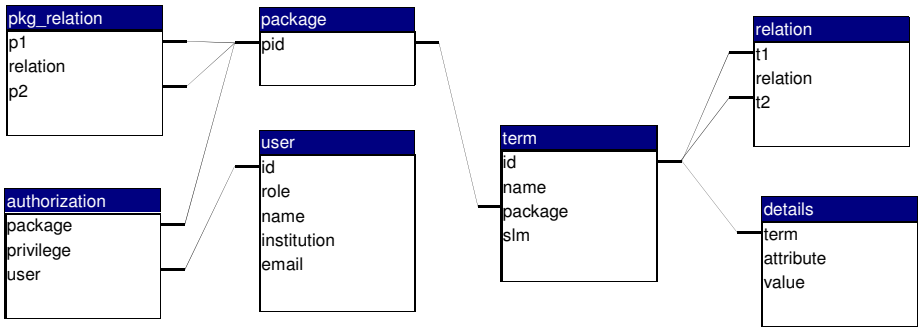

Fig. 3. Major Tables For ATO Ontology Database

More tables may be needed for concurrency management, such as page locking information. Detailed design needs further investigation.

3.3 Package Design

The whole ontology is built with many packages. Several rules are recommended in the design of packages.

– A package should have moderate size. In general, a package has a set of related terms, and the author of package should be clear about those terms. For a suggest, package size should between 10 and 10^3.
– Each species has an top level package, and aspects of traits of this species, such as meat quality and health, will be second level packages. There are may also third or even more level packages if necessary.
– It is best to keep the package hierarchy consistent with the semantic hierarchy(such as the term isa hierarchy). However, it could be different for managerial propose, for example, a large package can split into two smaller packages while the semantic structure is unchanged, or if a species has small number of terms, all aspects of its terms can be organized in one package.
– If a "pattern" is repeated in different packages, it is better to extract it as a reusable package. For example, If Production - Litter Size - Total Number Born ... is shared by more than one species, it can be extracted as a common template for all species.

3.4 Concurrent Management

A specific locking strategy should be defined in later stages of this project. For reference, please see similar systems such as OntoEdit[SEA+02] and WikiOnt[BH04a]

3.5 Ontology publication

The produced ontology can be released in OBO format and OWL format. With the OBO format, AmiGO can be used as a web browser for ATO. "AmiGO is an HTML based application which allows the user to browse, query and visualize data from the Gene Ontology, or any ontology in OBO format. AmiGO can be accessed online at http://www.godatabase.org/cgi-bin/amigo/go.cgi. AmiGO is open source software and can be downloaded and installed as part of the go-dev cvs repository from the Gene Ontology sourceforge page." (cited from AmiGO Users Guide [14])

The OBO version of ATO can be submitted to the Open Biomedical Ontologies site[15] as a member of Gene Ontology Consortium core ontology.

An OWL version can also be released. A OWL/RDFS schema of ATO ontology should be also released.

4 System Description

4.1 Workflow Description

The basic task for the editor is to edit existing ontology or add new module to the ontology. A typical workflow is as follows:

[14] http://www.godatabase.org/amigo/docs/user_guide/index.html
[15] http://obo.sourceforge.net/

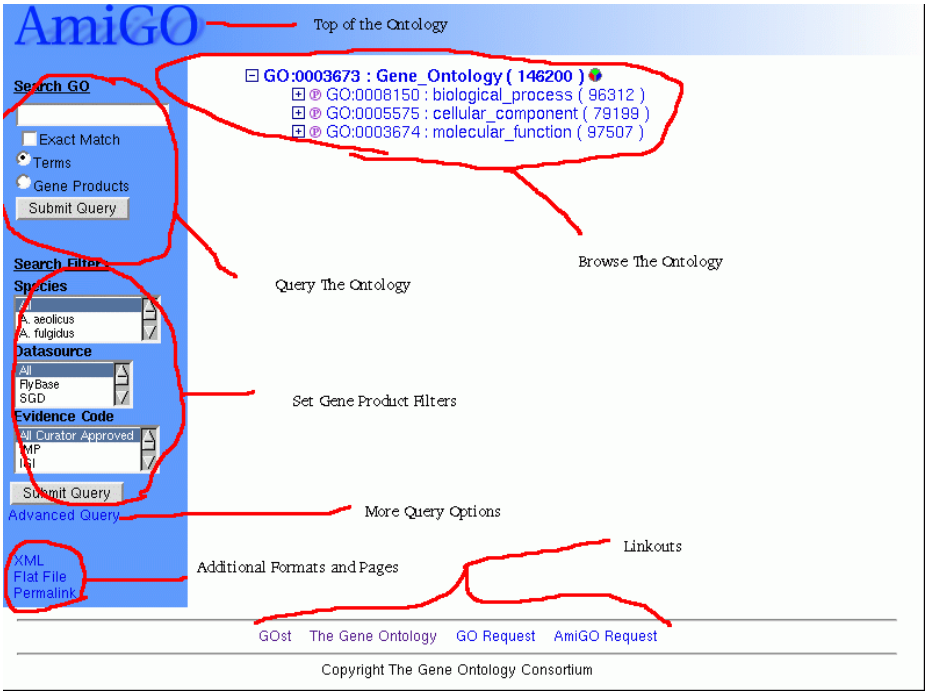

Fig. 4. AmiGO web ontology browser

1. User connects to the database, login with user name and password
2. User sees all packages, requests access to certain package (such as double-click), or creates new package
3. If the requested package is available (not locked by other users), user can access it with given privileges.
4. User may load part or all of the package into memory, do editing on terms; The edited package and other related packages, according to certain locking strategy, are locked.
5. During the editing, user can see both the organizational structure (package-wise) and the semantic structure (in isa or part-of hierarchy) of the ontology;
6. User submits the editing; the submission is checked by a reasoner, will be accepted or rejected.
7. User logouts and closes connection to the database.

4.2 Functional Description

The ontology editor should have following functions

- Ontology Editing
  - Load, save, create, split, merge package;

    - Create terms in package and define its visibility (public, private, protected);
    - Obsoleting, Deleting, and Destroying Terms;
    - Working with Term Relationships
    - Changing the Relationship Type
    - Moving a Term to a Different Parent (Drag and Drop)
    - Copying a Term to a New Parent
    - Merging Two Terms
    - Splitting a Term
    - Managing Relationship Types
    - Managing Obsolete Terms
    - Change relation for a subtree (move the sub-tree to be under another main term
    - Multiple tree structure allowed for the same set of ontology, eg. a term could be a isa tree and also a part-of tree
- IO
    - Saving packages
        * Save to OBO file
        * Save to OWL file
    - Loading packages
        * Load from OBO file
        * Load from OWL file
- User Management
    - Create user profile, include id, password,
    - Assign privilege for different packages
- Concurrent Management
    - Lock a package/term if it is under editing upon user requirement
    - Lock subpackages / superpackages when necessary
    - Lock related terms/ package on the DAG when necessary
- Reasoning: ensure consistency. For example, find cycle definition when a user submits an edited package.

4.3 Structural Description

The editor will be the extension of the preliminary editor, with following major modules

- Knowledge Representation (ato.kr): the classes for in-memory model of ATO ontology.
    - POOntology, Package, Term: for package-based partial order ontology.
    - AtoOntology, ATOTreeNode,Trait: ATO specific classes
- User Interface (ato.gui)
    - ATOEditor The editor behaviors (the menu, button and other user action handler)
    - ATOEditorGUI and a number of panels, dialogs: the editor GUI definitions

    - ATO2DB, DB2ATO : write/read the in-memory model to/from database
    - AtoVisualizer, ATOTreeRender: To build GUI trees for the in-memory model. DAG is shown with two trees, one top down and another bottom up (from selected node).
    - ATOTreeEditor The editing handler. It listens to popup menu action on the tree and handle user selection action.
- Concurrent Management: lock/unlock packages
    - LockingManager
- IO: import and export packages to/from OBO or OWL format. [it communicates with database, not in-memory model]
    - Ato2Obo, Ato2Owl
    - Obo2Ato, Owl2Ato
- Reasoner: to check consistency
    - POReasoner: to check if scope limitation is obeyed and no cyclic definition. a prototype is available at http://boole.cs.iastate.edu:9090/popeye/Wiki.jsp?page=Academic.Topic.INDUS.Reasoner

The editor may be executed from local or as applet/Jave Web Start. To be web accessible, the size of the compiled jar should be less than 5MB, the smaller the better.

5 Schedule

Stage I (June 6 - July 1): Initial scheme layout and code development

Stage II (July 5 - July 29): Test, debug, meet the demand; To properly comment the codes and document the program details for future works.

Stage III (August 1 - August 26): Modularize the tools to be packaged and tested on separate platforms; Draft an installation manual and a user's manual.

References

BCPH05. J. Bao, D. Caragea, J. Pathak, and V. Honavar. A framework to extend the semantics of data and schema of data source with ontology. In *Submitted to ODBASE 2005*. 2005.

BH04a. Jie Bao and Vasant Honavar. Collaborative ontology building with wiki@nt - a multi-agent based ontology building environment. In *Proc. of 3rd International Workshop on Evaluation of Ontology-based Tools, located at the 3rd International Semantic Web Conference ISWC 2004, 8th November 2004, Hiroshima, Japan*, 2004.

BH04b. Jie Bao and Vasant Honavar. Ontology language extensions to support localized semantics, modular reasoning, collaborative ontology design and reuse. In *3rd International Semantic Web Conference (ISWC2004), Poster Track , 7-11 November 2004, Hiroshima, Japan*, 2004.

Car04. D. Caragea. *Learning from Distributed, Heterogeneous and Autonomous Data Sources*. PhD thesis, Department of Computer Sciene, Iowa State University, USA, 2004.

CBP+05. D. Caragea, J. Bao, J. Pathak, A. Silvescu, C. Andorf., D. Dobbs, and V. Honavar. Information integration from semantically heterogeneous biological data sources. In *Proceedings of the 3rd International Workshop on Biological Data Management (BIDM'05), Copenhagen, Denmark.* 2005.

Den02. Michael Denny. Ontology building: A survey of editing tools. XML.com, http://www.xml.com/pub/a/2002/11/06/ontologies.html, 2002.

Den04. Michael Denny. Ontology tools survey, revisited. XML.com, http://www.xml.com/pub/a/2004/07/14/onto.html, 2004.

Pe02. Asuncion Gomez Perez and et.al. Deliverable 1.3: A survey on ontology tools. OntoWeb, http://citeseer.ist.psu.edu/525623.html, 2002.

Pol03. Irene Polikoff. Ontology tool support: Ontology development lifecycle and tools. TopQuadrant, http://www.topquadrant.com/documents/TQ1202_Ontology%20Tool%20Survey.pdf, 2003.

RCSC+03. J. Reinoso-Castillo, A. Silvescu, D. Caragea, J. Pathak, and V. Honavar. Information extraction and integration from heterogeneous, distributed, autonomous information sources: A federated, query-centric approach. In *IEEE International Conference on Information Integration and Reuse*, 2003.

SEA+02. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In *Proceedings of the first International Semantic Web Conference 2002 (ISWC 2002), June 9-12 2002, Sardinia, Italia.* Springer, LNCS 2342, 2002.