Swoop: A ‘Web’ Ontology Editing Browser

⋆

Aditya Kalyanpur[1], Bijan Parsia[2], Evren Sirin[1],
Bernardo Cuenca Grau[3], James Hendler[1]

*University of Maryland, MIND Lab, 8400 Baltimore Ave,
College Park MD 20742, USA*

{aditya, evren, hendler}[1]@cs.umd.edu, bparsia@isr.umd.edu[2],
bernardo@mindlab.umd.edu[3]

Abstract

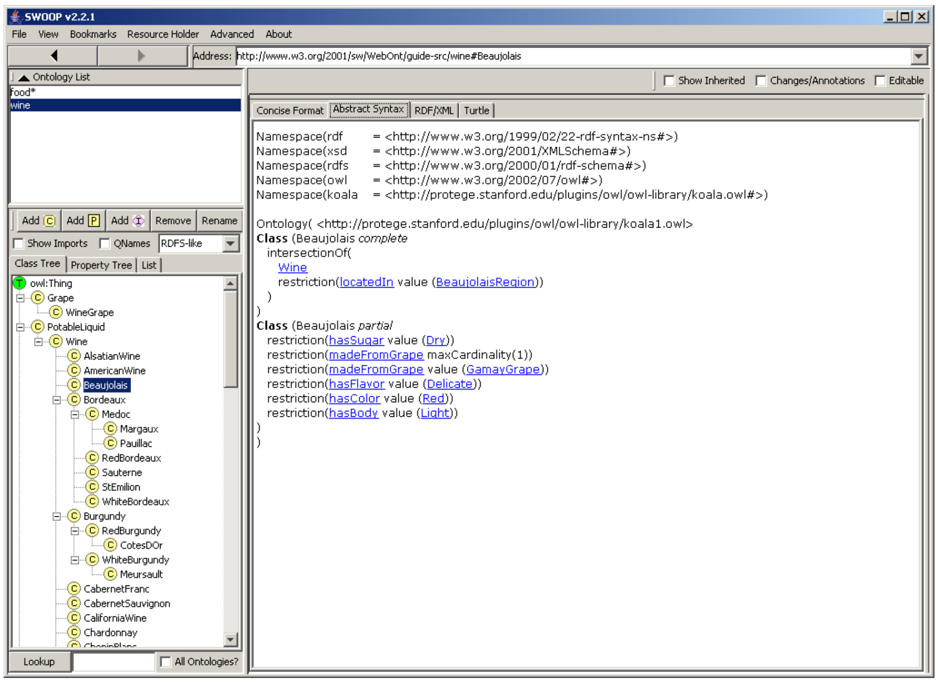
In this paper, we describe Swoop, a hypermedia inspired Ontology Browser and Editor based on OWL, the recently standardized Web-oriented ontology language. After discussing the design rationale and architecture of Swoop, we focus mainly on its features, using illustrative examples to highlight its use. We demonstrate that with its web-metaphor, adherence to OWL recommendations and key unique features such as Collaborative Annotation using Annotea, Swoop acts as a useful and efficient web ontology development tool. We conclude with a list of future plans for Swoop, that should further increase its overall appeal and accessibility.

Key words: OWL, Ontology Engineering, Systems

1 Introduction: Design Rationale and Goals

Swoop is built primarily as a *Web Ontology* Browser and Editor, i.e., it is tailored specifically for OWL [1] ontologies. Thus, it takes the standard Web browser as the UI paradigm, believing that URIs are central to the understanding and construction of OWL Ontologies. The familiar look and feel of a browser emphasized by the address bar and history buttons, navigation side bar, bookmarks, hypertextual navigation etc are all supported for web ontologies, corresponding with the mental model people have of URI-based web tools based on their current Web browsers (see **Fig. 1**).

⋆ Please note that this paper is intended to be a systems and not a research paper



```
Namespace(rdf     = <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Namespace(xsd     = <http://www.w3.org/2001/XMLSchema#>)
Namespace(rdfs    = <http://www.w3.org/2000/01/rdf-schema#>)
Namespace(owl     = <http://www.w3.org/2002/07/owl#>)
Namespace(koala   = <http://protege.stanford.edu/plugins/owl/owl-library/koala.owl#>)

Ontology( <http://protege.stanford.edu/plugins/owl/owl-library/koala1.owl>
Class (Beaujolais complete
  intersectionOf(
    Wine
    restriction(locatedIn value (BeaujolaisRegion))
  )
)
Class (Beaujolais partial
  restriction(hasSugar value (Dry))
  restriction(madeFromGrape maxCardinality(1))
  restriction(madeFromGrape value (GamayGrape))
  restriction(hasFlavor value (Delicate))
  restriction(hasColor value (Red))
  restriction(hasBody value (Light))
)
)
```

Fig. 1. **Swoop: A Web Ontology Browser**. The layout resembles a familiar frame-based website viewed through a Web browser. A navigation sidebar on the left contains the multiple ontology list and class/property hierarchies for each ontology, while the center pane contains the various ontology/entity renderers for displaying the core content. The history buttons (back, next) and address bar take their familiar position at the top of the frame.

All design decisions are in keeping with the OWL nature and specifications. Thus, multiple ontologies are supported easily, various OWL presentation syntax are used to render ontologies, OWL reasoners can be integrated for consistency checking, and open-world semantics are assumed while editing. To give an example of the latter aspect, some ontology editors make the assumption that an `owl:FunctionalProperty` can only allow one value in all cases thus providing a single input component. However, this is not true in the case of `OWL ObjectProperties` where multiple values (individuals) can exist for the functional property, allowing the reasoner to infer equivalence between the values.

A key point in our work is that the hypermedia basis of the UI is exposed in virtually every aspect of ontology engineering — easy navigation of OWL entities, comparing and editing related entities, search and cross referencing, multimedia support for annotation, etc. — thus allowing ontology developers to think of OWL as just another Web format, and thereby take advantage of its Web-based features.

A diverse array of ontology related tasks can be performed in Swoop ranging from collaborative annotation and data markup to ontology refactoring and debugging. This makes Swoop accessible to both, novice users interested in *casual* ontology building and use [2], and expert users interested in robust ontology modeling and analysis.

2 Related Work

There exist numerous ontology development toolkits (many of which have OWL support), that provide an integrated environment to build and edit ontologies, check for errors and inconsistencies (using a reasoner), browse multiple ontologies, and share and reuse existing data by establishing mappings among different ontological entities.

However, we differentiate Swoop on several aspects. First, to our knowledge, Swoop is the only ontology editor currently available that is catered wholly towards OWL from the ground up and its design rationale reflects this. Second, it contrasts with other ontology editors such as Protégé [3], OilED [4], and OntoEdit [5] which either are, or were influenced by traditional KR development tools and applications, and do not reflect "Webiness" in their UI design. In particular, they do not fully support the use of hypertext to drive the exploration and editing of ontologies.

Also, as mentioned in [6], there are several examples of current website-based ontology tools(e.g. Ontosaurus [7], WebODE [8]), pOWL [1]). However, we have found that using a standard web-based server-client architecture for ontology engineering suffers from being slow (esp. for large ontologies, and depending on network traffic), and cumbersome for maintaining consistency while editing (eg. trapping input errors, changing/deleting objects but reloading from browser cache etc), an argument also supported in [9]. In addition, such tools can be difficult to extend to new functionalities via plug-in architectures (such as the one used in Swoop). For these reasons, Swoop is developed as a separate Java application that attempts to provide the look and feel of a browser-based application, but with its specialized architecture designed to optimize OWL browsing and to be extensible via a plug-in architecture.

3 Swoop Architecture

Swoop is based on the Model-View-Controller (MVC [10]) paradigm. The *SwoopModel* component stores all ontology-centric information pertaining to the Swoop Workspace (currently loaded ontologies, change-logs, checkpoints) and defines key parameters used by the Swoop UI objects (such as selected

[1] pOWL - http://powl.sourceforge.net

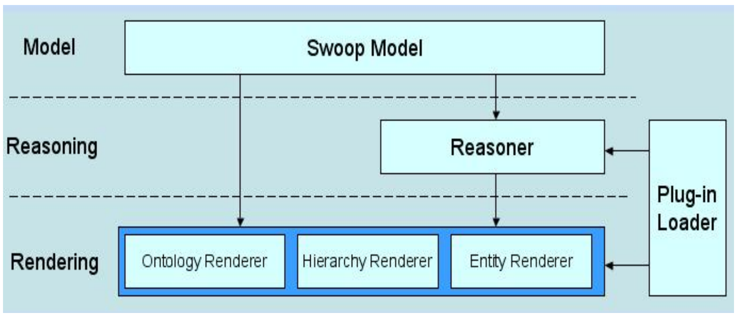


Fig. 2. **Plugin-based Swoop Architecture**

OWL entity, view settings for imports, QNames etc). Additionally, a *Swoop-ModelListener* class is used to reflect changes in the UI based on changes in the SwoopModel (using a suitably defined event-notification scheme). Control is handled through a **plugin based system**, which loads new *Renderers* and *Reasoners* dynamically. The obvious advantage of a plugin framework is to ensure modularity of the code, and encourage external developers to contribute to the Swoop project easily. Finally, we note that the entire Swoop code is written in Java, maintained in a subversion repository and makes use of numerous third party libraries, the most prominent being the WonderWeb OWL API [11] which is used as the underlying OWL representation model.

4 Swoop Features

In this section, we describe the features of Swoop that are in keeping with its design rationale and goals mentioned earlier. In particular, we focus on the unique aspects of 'Web' Ontologies and the corresponding tool support needed to develop each aspect effectively. Note that all features mentioned in this paper are with reference to the v2.2 release of Swoop, which can be obtained at the Swoop website: http://www.mindswap.org/2004/SWOOP.

4.1 Ontologies based on the Web Architecture

The idea behind Web ontology development is different from traditional and more controlled ontology engineering approaches which rely on high investment, relatively large, heavily engineered, mostly monolithic ontologies. For OWL ontologies, which are based on the Web architecture (characterized as being open, distributed and scalable), the emphasis is more on utilizing this *freeform* nature of the Web to develop and share (preferably smaller) ontology models in a relatively ad hoc manner, allowing ontological data to be reused easily, either by linking models (using the numerous mapping properties available in OWL) or merging them (using the `owl:imports` command). Thus, it becomes essential for a Web ontology development tool to *scale* to multiple ontologies easily, and to allow tasks such as creation, browsing, editing, search, reuse, linking, merge/split of OWL ontology models in the context of multiple large distributed ontologies.

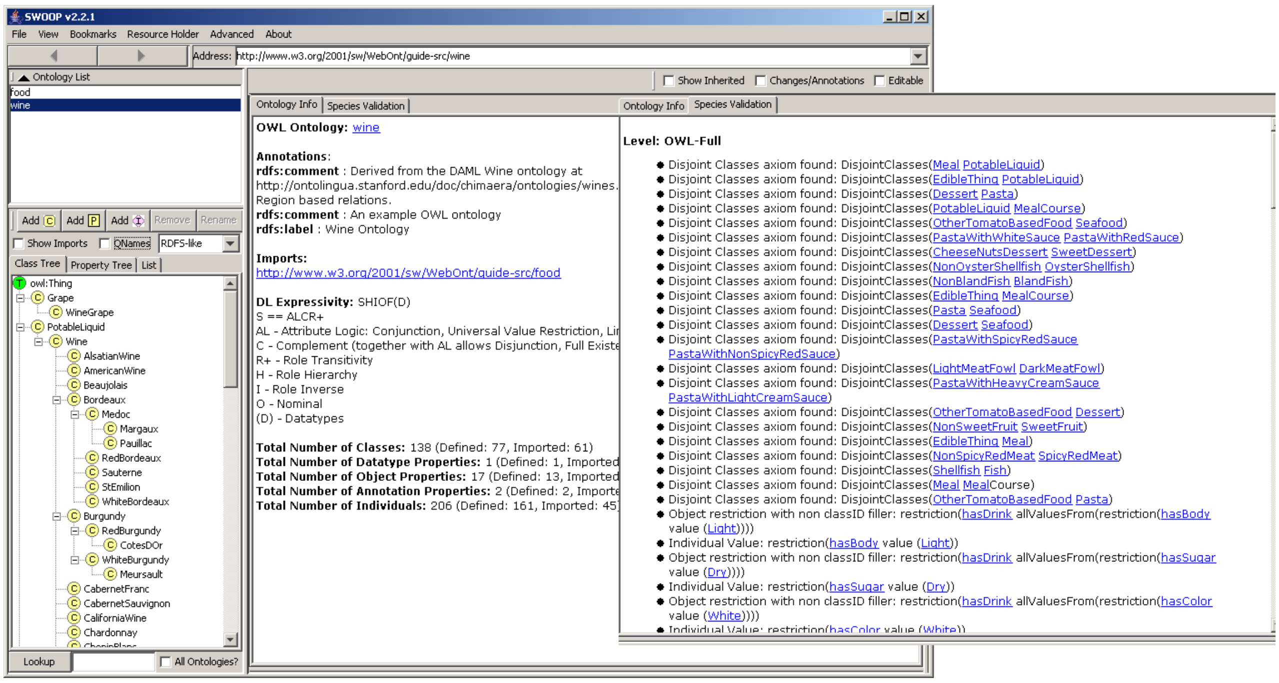
In order to attain this key requirement, Swoop ensures that users are free to load multiple OWL ontologies in any manner they prefer. The easiest and most direct way to load an OWL Ontology is by entering its physical URL (Web or local file address) in the address bar. This action not only pulls in the requested ontology, but also loads any imported ontologies (defined using `owl:imports`) into Swoop automatically. The **bookmarks** feature can be used to store, categorize and reload ontologies directly (as is the case in standard web browsers). Finally, depending on user preference, an ontology can also be brought into Swoop rather seamlessly during browsing/editing, e.g., attempting to view or refer to an externally referenced entity while in a particular ontology can load the external ontology automatically.

There are certain characteristics of OWL ontologies which are presented to the user when a new ontology is brought into Swoop. The Ontology Renderer plugins in Swoop accomplish this, and display statistics such as (see **Fig. 3**):

- the logical constructs used in the ontology model which determine the OWL species level the ontology belongs to, i.e., OWL Lite, DL or Full
- the Description Logic (DL) expressivity of the ontology - a key factor in determining complexity of reasoning
- number of classes, properties, individuals etc. (we intend to extend the granularity to axioms, e.g., no. of disjoint axioms, no. of nominals used etc.)
- annotations on the ontology object itself (including `owl:imports`)

4.1.1 Editing Web Ontologies

Consider a scenario in which a user is building an ontology for the University of Maryland (UMD) for describing its administrative hierarchy (with concepts such as `Department, Faculty, Staff, Student` etc). This user can make use of existing concepts in well-known upper-level ontologies such as FOAF or Cyc (for generic concepts such as `Person`), or in similar ontologies created for other universities (such as Stanford or CMU). Another user interested in building a finer-grained ontology than the one above, say for describing his/her research group at UMD, can now use the UMD ontology to refer to or define certain concepts. In this manner, the open development cycle of *create-link-share* web ontologies ensures that a large amount of interrelated semantic content is available in ontologies. Moreover, it satisfies one of the fundamental necessities of effective ontology development for the Semantic Web, that of semantic interoperability – instance data written in one ontology


SWOOP v2.2.1
File View Bookmarks Resource Holder Advanced About
Address: http://www.w3.org/2001/sw/WebOnt/guide-src/wine
Ontology List
food
wine
Add C Add P Add I Remove Rename
Show Imports QNames RDFS-like
Class Tree Property Tree List
owl:Thing
Grape
WineGrape
PotableLiquid
Wine
AlsatianWine
AmericanWine
Beaujolais
Bordeaux
Medoc
Margaux
Pauillac
RedBordeaux
Sauterne
StEmilion
WhiteBordeaux
Burgundy
RedBurgundy
CotesDOr
WhiteBurgundy
Meursault
CabernetFranc
CabernetSauvignon
CaliforniaWine
Chardonnay
Lookup All Ontologies?
Show Inherited Changes/Annotations Editable
Ontology Info Species Validation
Ontology Info Species Validation

OWL Ontology: wine

Annotations:
rdfs:comment : Derived from the DAML Wine ontology at
http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.
Region based relations.
rdfs:comment : An example OWL ontology
rdfs:label : Wine Ontology

Imports:
http://www.w3.org/2001/sw/WebOnt/guide-src/food

DL Expressivity: SHIOF(D)
S == ALCR+
AL - Attribute Logic: Conjunction, Universal Value Restriction, Lir
C - Complement (together with AL allows Disjunction, Full Existe
R+ - Role Transitivity
H - Role Hierarchy
I - Role Inverse
O - Nominal
(D) - Datatypes

Total Number of Classes: 138 (Defined: 77, Imported: 61)
Total Number of Datatype Properties: 1 (Defined: 1, Imported
Total Number of Object Properties: 17 (Defined: 13, Imported
Total Number of Annotation Properties: 2 (Defined: 2, Importe
Total Number of Individuals: 206 (Defined: 161, Imported: 45)

Level: OWL-Full

- Disjoint Classes axiom found: DisjointClasses(Meal PotableLiquid)
- Disjoint Classes axiom found: DisjointClasses(EdibleThing PotableLiquid)
- Disjoint Classes axiom found: DisjointClasses(Dessert Pasta)
- Disjoint Classes axiom found: DisjointClasses(PotableLiquid MealCourse)
- Disjoint Classes axiom found: DisjointClasses(OtherTomatoBasedFood Seafood)
- Disjoint Classes axiom found: DisjointClasses(PastaWithWhiteSauce PastaWithRedSauce)
- Disjoint Classes axiom found: DisjointClasses(CheeseNutsDessert SweetDessert)
- Disjoint Classes axiom found: DisjointClasses(NonOysterShellfish OysterShellfish)
- Disjoint Classes axiom found: DisjointClasses(NonBlandFish BlandFish)
- Disjoint Classes axiom found: DisjointClasses(EdibleThing MealCourse)
- Disjoint Classes axiom found: DisjointClasses(Pasta Seafood)
- Disjoint Classes axiom found: DisjointClasses(Dessert Seafood)
- Disjoint Classes axiom found: DisjointClasses(PastaWithSpicyRedSauce PastaWithNonSpicyRedSauce)
- Disjoint Classes axiom found: DisjointClasses(LightMeatFowl DarkMeatFowl)
- Disjoint Classes axiom found: DisjointClasses(PastaWithHeavyCreamSauce PastaWithLightCreamSauce)
- Disjoint Classes axiom found: DisjointClasses(OtherTomatoBasedFood Dessert)
- Disjoint Classes axiom found: DisjointClasses(NonSweetFruit SweetFruit)
- Disjoint Classes axiom found: DisjointClasses(EdibleThing Meal)
- Disjoint Classes axiom found: DisjointClasses(NonSpicyRedMeat SpicyRedMeat)
- Disjoint Classes axiom found: DisjointClasses(Shellfish Fish)
- Disjoint Classes axiom found: DisjointClasses(Meal MealCourse)
- Disjoint Classes axiom found: DisjointClasses(OtherTomatoBasedFood Pasta)
- Object restriction with non classID filler: restriction(hasDrink allValuesFrom(restriction(hasBody value (Light))))
- Individual Value: restriction(hasBody value (Light))
- Object restriction with non classID filler: restriction(hasDrink allValuesFrom(restriction(hasSugar value (Dry))))
- Individual Value: restriction(hasSugar value (Dry))
- Object restriction with non classID filler: restriction(hasDrink allValuesFrom(restriction(hasColor value (White))))
- Individual Value: restriction(hasColor value (White))

Fig. 3. **Swoop Ontology Renderers**: display statistical information about the ontology, annotations, DL expressivity and OWL species level.

can be transformed into data for another ontology provided the necessary links exist.

In keeping with the above scenario, Swoop allows users to freely link (map between) entities in different ontologies using a single common interface, which lists each ontology loaded in Swoop along with its corresponding entity list (see **Fig: 4**).

This seemingly simple feature underlines the true significance and power of Web ontologies as described above, i.e., the ease in which ontological data can be reused creating a *semantic mesh of interlinked ontologies* such that, as the mesh grows larger and more interconnected, it becomes easier and more useful for other web ontology developers/users to build and utilize specific ontology models on the fly.

It is important to consider the scenario in which a user edits an external ontology present at a URL under the control of a third party. In this case, though the ontology could be considered as *read-only*, Swoop allows the ontology to be modified and a local version of the ontology to be maintained separately (since it cannot be exported to the URL). However, now, its new physical location is used for reference in an `owl:import` axiom that specifies importing the external ontology. In general, Swoop tracks import-links across the ontologies in its environment, and when the physical location of an ontology is changed, the corresponding import links are updated as well.

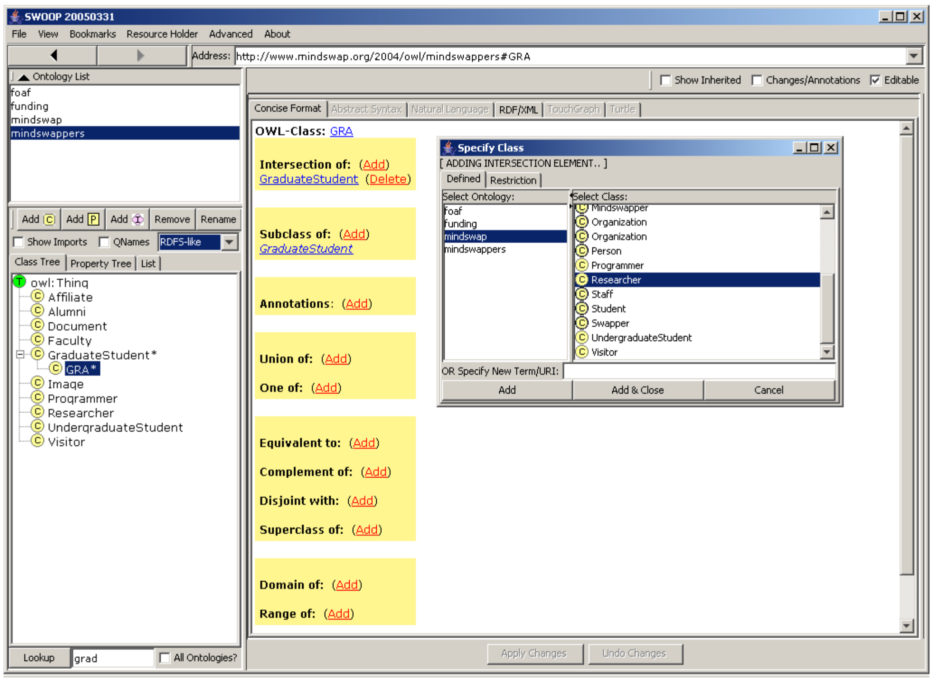
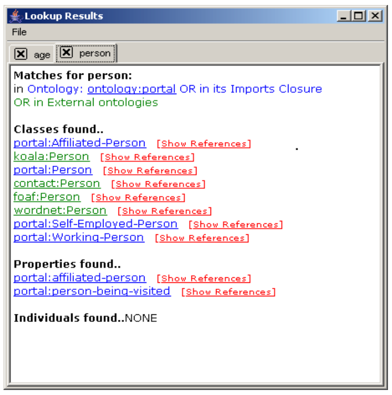


Fig. 4. **Editing in Swoop**: Clicking the 'Add' hyperlink next to an assertion heading (e.g., **Intersection of**) pops up a window to specify corresponding new information (e.g., the new intersection class). In this case, the user is specifying a class `Researcher` from an external ontology `mindswap` as an intersection element

However, there are additional caveats to be considered while editing in a multiple ontology setting as described above. For starters, it is essential to provide a search feature to help users find related ontological information. Having found such information, it then becomes critical to compare and analyze this information in order to determine which parts, if any, are useful (verifying relevance, accuracy etc). Finally, the user needs a flexible reuse scheme that supports either borrowing the entire external ontology model if desired, or a subset of it which is relevant, allowing suitable modifications if any. We deal with each of these three caveats in detail as reflected in Swoop.

Search in Swoop essentially performs a lookup for entities (classes/ properties/ individuals) across single or multiple ontologies, among those that have been loaded. The results are obtained as a set of hyperlinks (in keeping with the hypermedia-based UI) allowing the user to browse the search results easily (see **Fig. 5**).

During an extensive search/browsing process, the user may need to set aside and revisit interesting search results. In Swoop we have a provision to store and compare OWL entities via a **Resource Holder** panel (see **Fig 6**). Items can be added to this panel at any time and they remain static there until the user decides to remove or replace them at a later stage. This common placeholder acts as an excellent platform for performing interesting engineering tasks such



Matches for person:
in Ontology: ontology:portal OR in its Imports Closure
OR in External ontologies

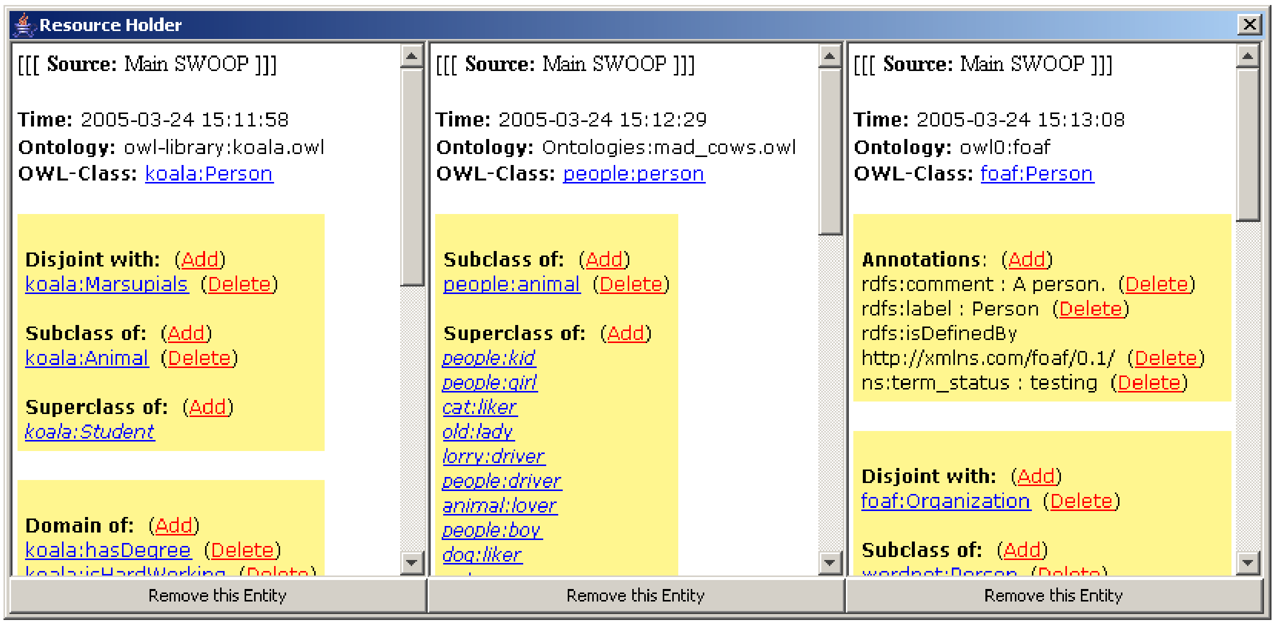
Classes found..
portal:Affiliated-Person [Show References]
koala:Person [Show References]
portal:Person [Show References]
contact:Person [Show References]
foaf:Person [Show References]
wordnet:Person [Show References]
portal:Self-Employed-Person [Show References]
portal:Working-Person [Show References]

Properties found..
portal:affiliated-person [Show References]
portal:person-being-visited [Show References]

Individuals found..NONE

Fig. 5. Ontology Search in Swoop for the keyword 'Person'

as comparing differences in definitions of a set of entities; determining semantic mappings between a specific pair of entities or simply storing entities for reusing in another ontology.



[[[**Source:** Main SWOOP]]]

Time: 2005-03-24 15:11:58
Ontology: owl-library:koala.owl
OWL-Class: koala:Person

Disjoint with: (Add)
koala:Marsupials (Delete)

Subclass of: (Add)
koala:Animal (Delete)

Superclass of: (Add)
koala:Student

Domain of: (Add)
koala:hasDegree (Delete)

[[[**Source:** Main SWOOP]]]

Time: 2005-03-24 15:12:29
Ontology: Ontologies:mad_cows.owl
OWL-Class: people:person

Subclass of: (Add)
people:animal (Delete)

Superclass of: (Add)
people:kid
people:girl
cat:liker
old:lady
lorry:driver
people:driver
animal:lover
people:boy
dog:liker

[[[**Source:** Main SWOOP]]]

Time: 2005-03-24 15:13:08
Ontology: owl0:foaf
OWL-Class: foaf:Person

Annotations: (Add)
rdfs:comment : A person. (Delete)
rdfs:label : Person (Delete)
rdfs:isDefinedBy
http://xmlns.com/foaf/0.1/ (Delete)
ns:term_status : testing (Delete)

Disjoint with: (Add)
foaf:Organization (Delete)

Subclass of: (Add)

Fig. 6. **Resource Holder:** used to compare 3 different definitions of the OWL Class `Person` as specified in 3 different ontologies

4.1.2 Reusing OWL Data: The Partial Imports Problem

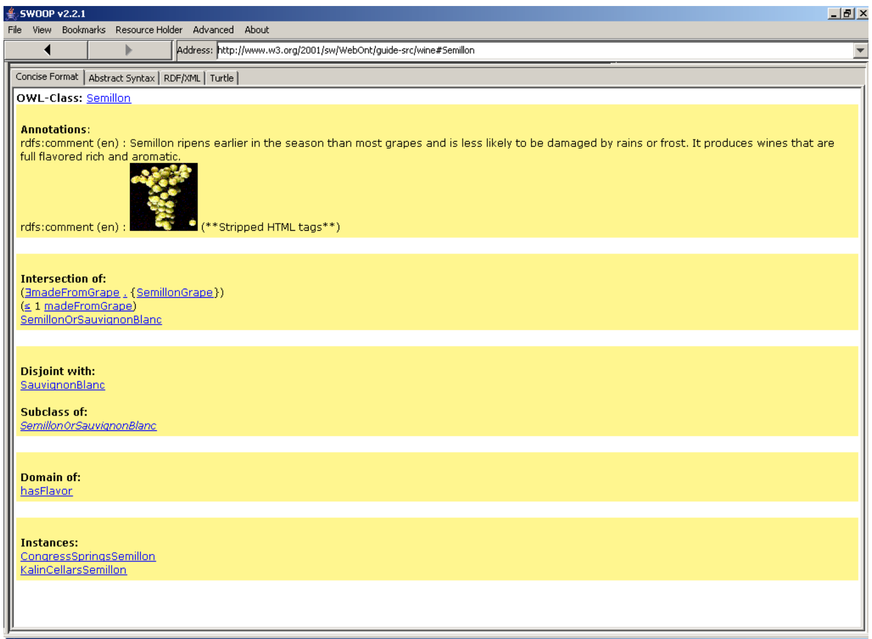
Having dealt with the important aspects of search and comparison of related data in a multiple ontology setting, we are now left with the critical task of reusing data efficiently. Note that all the constructs in RDFS/OWL (e.g. `owl:equivalentTo, rdfs:subClassOf`) can be used to relate entities from different ontologies. However, the actual semantics of the linked entities depends on whether the corresponding ontologies *import* one another, e.g., if class C_A (in ontology A) is defined as `equivalentTo` class C_B (in ontology

B), the semantics of C_A differ depending on whether A imports B or not. Thus, effectively, there are two ways in which users can reuse external ontological data, i.e., either by simply linking to the external entity, or by linking to the entity and importing the entire external ontology (using `owl:imports`). Already, one of the key drawbacks of the above reuse scheme has become clear – either users are forced to underspecify the semantics of their model by linking to but not importing the external ontology (using the former approach), or they are forced to overspecify their model by importing the entire semantics, i.e. all axioms, of the imported ontology (using the latter approach). Thus, there is no easy, direct solution for **partial imports** in OWL. Various workarounds exist such as a brute-force syntactic scheme to copy/paste relevant parts (axioms) of the external ontology, or a more elegant solution that involves partitioning the external ontology while preserving its semantics and then reusing (importing) only the specific partition as desired.

We have explored the latter solution for partitioning OWL Ontologies by transforming them into an *E-connection*. Describing the theory and significance of *E-connections*, their use in the context of OWL Ontologies and details of the partitioning algorithm are beyond the scope of this paper. For more on that visit http://www.mindswap.org/2004/multipleOnt. Here we briefly describe the process: an ontology dealing with numerous inter-related domains is transformed into an E-connection which consists of sub-ontologies (partitions) that each deal with a distinct domain. An obvious advantage of the above modularity is that a user interested in reusing a concept from a particular domain, can import only the corresponding partition instead of the entire original ontology.

4.2 Adhering to OWL Specifications: Presentation and Reasoning

Currently, various presentation syntax exist for rendering OWL ontologies such as RDF/XML [12], OWL Abstract Syntax [13], and Turtle [14]. It is important to support these different syntax while designing an open, Semantic Web ontology engineering environment. One reason for this is that people tend to have strong biases toward different notations and simply prefer to work in one or another. A second is that some other tool might only consume one particular syntax (with the RDF/XML syntax being the most typical), but that syntax might not be an easy or natural one for a particular user. A third is that it is important to support the "view source" effect, allowing cut and paste reuse into different tools including text editors, markup tools, or other semantic web tools. For these reasons, Swoop has default plugins for all three presentation syntax mentioned above. Users are free to browse and edit[2] ontological data, either at the level of a single entity (inline) or at the level of the entire ontology as a whole, in any syntax as desired, switching between syntax on a single click.


SWOOP v2.2.1
File View Bookmarks Resource Holder Advanced About
Address: http://www.w3.org/2001/sw/WebOnt/guide-src/wine#Semillon
Concise Format | Abstract Syntax | RDF/XML | Turtle

OWL-Class: Semillon

Annotations:
rdfs:comment (en) : Semillon ripens earlier in the season than most grapes and is less likely to be damaged by rains or frost. It produces wines that are full flavored rich and aromatic.
rdfs:comment (en) : (**Stripped HTML tags**)

Intersection of:
(∃madeFromGrape . {SemillonGrape})
(≤ 1 madeFromGrape)
SemillonOrSauvignonBlanc

Disjoint with:
SauvignonBlanc

Subclass of:
SemillonOrSauvignonBlanc

Domain of:
hasFlavor

Instances:
CongressSpringsSemillon
KalinCellarsSemillon

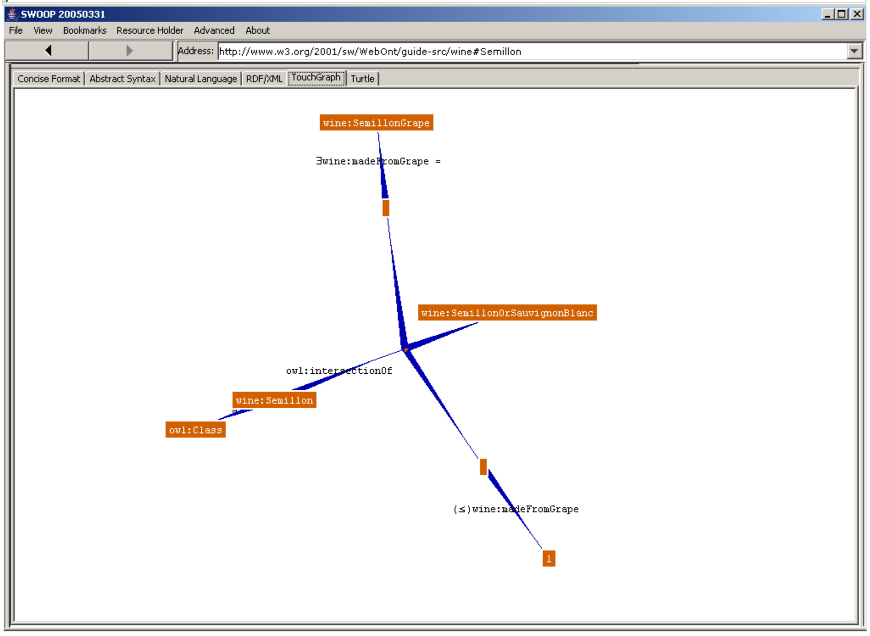

SWOOP 20050331
File View Bookmarks Resource Holder Advanced About
Address: http://www.w3.org/2001/sw/WebOnt/guide-src/wine#Semillon
Concise Format | Abstract Syntax | Natural Language | RDF/XML | TouchGraph | Turtle
wine:SemillonGrape
∃wine:madeFromGrape =
wine:SemillonOrSauvignonBlanc
owl:intersectionOf
wine:Semillon
owl:Class
(≤)wine:madeFromGrape
1

Fig. 7. **Top: Concise Format Entity Renderer**: displays all ontological information about the OWL Class `Semillon` as a single *Web document* **Bottom: Graph Visualization plugin**: displays a conceptual graph with nodes as classes and edges as properties (restrictions and collections are concisely represented)

In addition to the default OWL presentation syntax, we are working on three

[2] Note: Swoop v2.2 has support for inline editing in RDF/XML, the subsequent release will allow for inline editing in the other two syntax as well

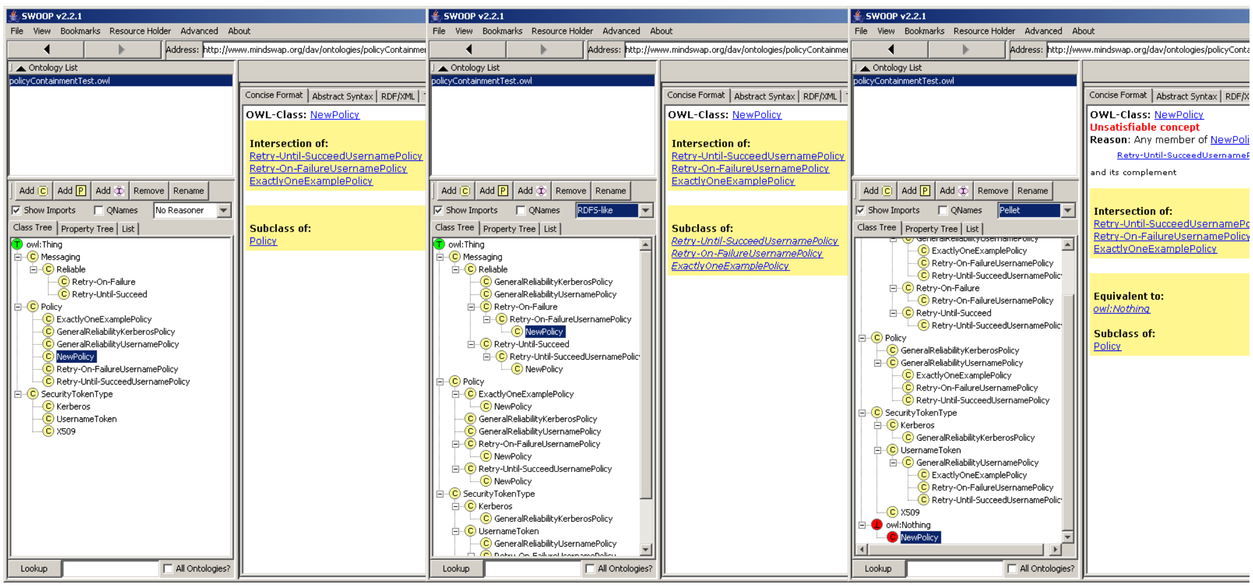
additional renderers to help users visualize and understand OWL ontologies better (two of which are shown in **Fig. 7**). These include a **Concise Format** entity renderer, where the idea is to generate a "Web document" that displays all information related to a particular OWL entity concisely in a single pane; a **Natural Language** entity renderer that provides concise, accurate NL paraphrases for OWL Concepts based on a variety of NLP techniques [15]; and an **OWL Graph visualization** renderer based on TouchGraph that displays concise conceptual graphs of the ontology model. Each of these renderers provide a different view of the model, allowing users to understand logical definitions and relationships better.

4.2.1 Reasoning in OWL

Having covered the presentation aspects of OWL ontologies, we now focus on the *reasoning* support in Swoop. Note that OWL-DL is primarily based on description logic, with open-world semantics and a non unique name assumption (UNA). Swoop strictly maintains the latter two aspects during editing, e.g., it does not try to 'interfere' with creating the KB (i.e., prevent the creation of inconsistencies) by making any additional alterations or assumptions, and accurately reflects the users' actions based on open world semantics. As for the DL reasoning, Swoop allows for special-purpose reasoner plugins that provide standard reasoner services such as satisfiability (of a single class as well as consistency of the ontology), subsumption (between classes and between properties), and realization (types of an instance). Additionally, reasoners can support the optional *explanations* feature, which is used for sophisticated ontology debugging as explained later.

Swoop contains two additional reasoners (besides the basic *Reasoner* that simply uses the asserted structure of the ontology): *RDFS-like* and *Pellet* [16]. While the former is a lightweight reasoner based on RDFS semantics, the latter, Pellet, is a powerful description logic tableaux reasoner. Pellet has a number of advantages: It natively supports OWL, including a repairable subset of OWL Full; it has extensive support for XML Schema datatypes; it has ABox (a.k.a., instance) support; it covers the broadest range of OWL DL of any reasoner that we know, including both SHIN(D), SHON(D), SHIO(D), and various subsets of their union, SHOIN(D) (a.k.a., OWL DL); it is open source and in active, public development. The last is very important for certain debugging strategies which require access to the internals of the reasoner as noted later.

The above reasoners provide a tradeoff between speed and quality of inference results, e.g., the *RDFS-like* reasoner, while much faster than Pellet in execution, is unsound (results maybe inaccurate if the ontology is inconsistent) and incomplete (does not list all possible inferences). Yet, in most cases, it provides interesting and useful results for ontology authors, and moreover, the reasoners can be used in conjunction to analyze the ontology quickly while editing it. **Fig. 8** illustrates the use and functionality of the various reasoners in Swoop, by highlighting the difference in the results provided for the same class definition in a given ontology.



```
No Reasoner

owl:Thing
  Messaging
    Reliable
      Retry-On-Failure
      Retry-Until-Succeed
  Policy
    ExactlyOneExamplePolicy
    GeneralReliabilityKerberosPolicy
    GeneralReliabilityUsernamePolicy
    NewPolicy
    Retry-On-FailureUsernamePolicy
    Retry-Until-SucceedUsernamePolicy
  SecurityTokenType
    Kerberos
    UsernameToken
    X509

OWL-Class: NewPolicy
Intersection of:
Retry-Until-SucceedUsernamePolicy
Retry-On-FailureUsernamePolicy
ExactlyOneExamplePolicy
Subclass of:
Policy
```

```
RDFS-like

owl:Thing
  Messaging
    Reliable
      GeneralReliabilityKerberosPolicy
      GeneralReliabilityUsernamePolicy
      Retry-On-Failure
        Retry-On-FailureUsernamePolicy
          NewPolicy
      Retry-Until-Succeed
        Retry-Until-SucceedUsernamePolic
          NewPolicy
  Policy
    ExactlyOneExamplePolicy
      NewPolicy
    GeneralReliabilityKerberosPolicy
    GeneralReliabilityUsernamePolicy
    Retry-On-FailureUsernamePolicy
      NewPolicy
    Retry-Until-SucceedUsernamePolicy
      NewPolicy
  SecurityTokenType
    Kerberos
      GeneralReliabilityKerberosPolicy
    UsernameToken
      GeneralReliabilityUsernamePolicy

OWL-Class: NewPolicy
Intersection of:
Retry-Until-SucceedUsernamePolicy
Retry-On-FailureUsernamePolicy
ExactlyOneExamplePolicy
Subclass of:
Retry-Until-SucceedUsernamePolicy
Retry-On-FailureUsernamePolicy
ExactlyOneExamplePolicy
```

```
Pellet

        ExactlyOneExamplePolicy
        Retry-On-FailureUsernamePolicy
        Retry-Until-SucceedUsernamePolic
      Retry-On-Failure
        Retry-On-FailureUsernamePolicy
      Retry-Until-Succeed
        Retry-Until-SucceedUsernamePolic
  Policy
    GeneralReliabilityKerberosPolicy
    GeneralReliabilityUsernamePolicy
      ExactlyOneExamplePolicy
      Retry-On-FailureUsernamePolicy
      Retry-Until-SucceedUsernamePolicy
  SecurityTokenType
    Kerberos
      GeneralReliabilityKerberosPolicy
    UsernameToken
      GeneralReliabilityUsernamePolicy
        ExactlyOneExamplePolicy
        Retry-On-FailureUsernamePolicy
        Retry-Until-SucceedUsernamePolic
    X509
  owl:Nothing
    NewPolicy

OWL-Class: NewPolicy
Unsatisfiable concept
Reason: Any member of NewPoli
    Retry-Until-SucceedUsernameP
and its complement
Intersection of:
Retry-Until-SucceedUsernamePo
Retry-On-FailureUsernamePolicy
ExactlyOneExamplePolicy
Equivalent to:
owl:Nothing
Subclass of:
Policy
```

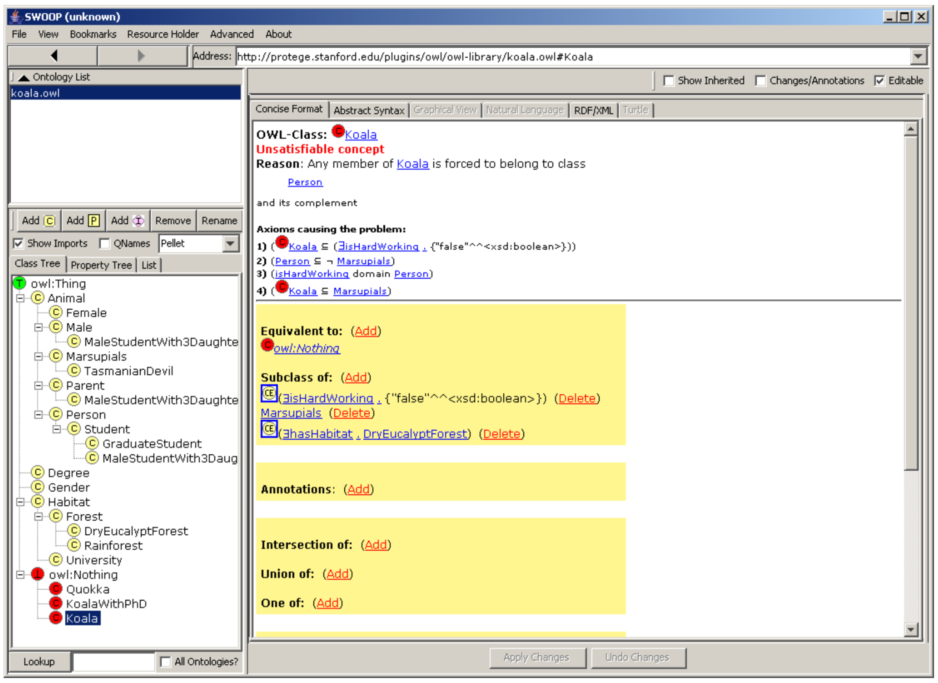
Fig. 8. **Using the Swoop Reasoners:** Note the change in position of the class `NewPolicy` in the subsumption graph when the reasoner selection changes. Initially, it is known to be a subclass of `Policy` in the asserted (No Reasoner) mode; then inferred to be a subclass of three distinct classes in the RDFS mode, and finally found to be unsatisfiable in Pellet mode.

4.2.2 Ontology Debugging and Repair

DL reasoners such as Pellet, RACER [17], FACT [18] etc can be used to detect inconsistencies in definitions of concepts (a.k.a. unsatisfiable concepts). However, typically reasoners only report that a class is unsatisfiable, not *why*. Moreover, they do not report on inter-dependencies (if any) of the unsatisfiable classes, i.e., if a class directly depends on another for its unsatisfiability (e.g., by an existential property restriction on an unsatisfiable class). We argue that both forms of explanation are essential for the purpose of debugging ontologies; while the former can be used to understand and rectify problematic axioms / class expressions, the latter can help prune out dependency bugs and let the modeler focus on the root (source) of the problem alone.

We distinguish two families of reasoner-based techniques for supporting diagnosis of the form described above: glass box and black box techniques. In glass box techniques, information from the *internals* of the reasoner is extracted and presented to the user (typically used to pinpoint the type of clash/contradiction and axioms leading to the clash). In black box techniques, the reasoner is used as an oracle for a certain set of questions e.g., the standard description logic inferences (subsumption, satisfiability, etc.) and the asserted structure of the ontology is used to help isolate the source of the problems (can be used to find dependencies between unsatisfiable classes). Presenting the details of both approaches is beyond the scope of this paper, for more see [19].

For now, we note that both forms of ontology debugging in Swoop is exposed through reasoners which support the optional *explanations* service (Pellet by default does this), and describe a simple example to illustrate the use of debugging: Consider the case of the unsatisfiable class `Koala` depicted in Figure 9. Swoop displays a one line quasi-Natural Language description explaining the cause of the clash in the class definition, followed by a list of the relevant set of axioms from the ontology responsible for the clash. This provides a direct pointer to the problematic part of the ontology as making this class satisfiable involves removing or fixing any one of the axioms displayed.



OWL-Class: Koala
Unsatisfiable concept
Reason: Any member of Koala is forced to belong to class
Person
and its complement

Axioms causing the problem:
1) (Koala ⊑ (∃isHardWorking . {"false"^^<xsd:boolean>}))
2) (Person ⊑ ¬ Marsupials)
3) (isHardWorking domain Person)
4) (Koala ⊑ Marsupials)

Equivalent to: (Add)
owl:Nothing

Subclass of: (Add)
(∃isHardWorking . {"false"^^<xsd:boolean>}) (Delete)
Marsupials (Delete)
(∃hasHabitat . DryEucalyptForest) (Delete)

Annotations: (Add)

Intersection of: (Add)

Union of: (Add)

One of: (Add)

Fig. 9. The set of axioms that support the inconsistency of Koala concept is displayed in debug mode.

4.3 Extending the Swoop Framework for Collaborative, Distributed Web Ontology Evolution

Till now we have seen how Swoop makes use of the Web-based nature of OWL Ontologies and remains true to the OWL specifications while providing for an open and scalable hypermedia-inspired ontology development environment.

Now, we extend the distributed nature of Web-based ontologies to support collaborative ontology evolution in Swoop. Our goal is to allow a group of ontology authors, possibly working in a remote environment (i.e., on separate machines), to develop and evolve a set of OWL ontologies together. The set of tasks we would like to support include:

- **Easy Exchange of Ontological Data** at different granularity levels such as a set of ontologies/entities, a single ontology/entity or a *change set.*
- **Collaborative Annotation** for exchanging and discussing ideas during all stages of the ontology development cycle
- **Version Control** for efficiently maintaining the ontologies over time by logging all changes and regulating its progress
- **Unit Testing** to ensure that the ontology satisfies certain predetermined criteria and goals

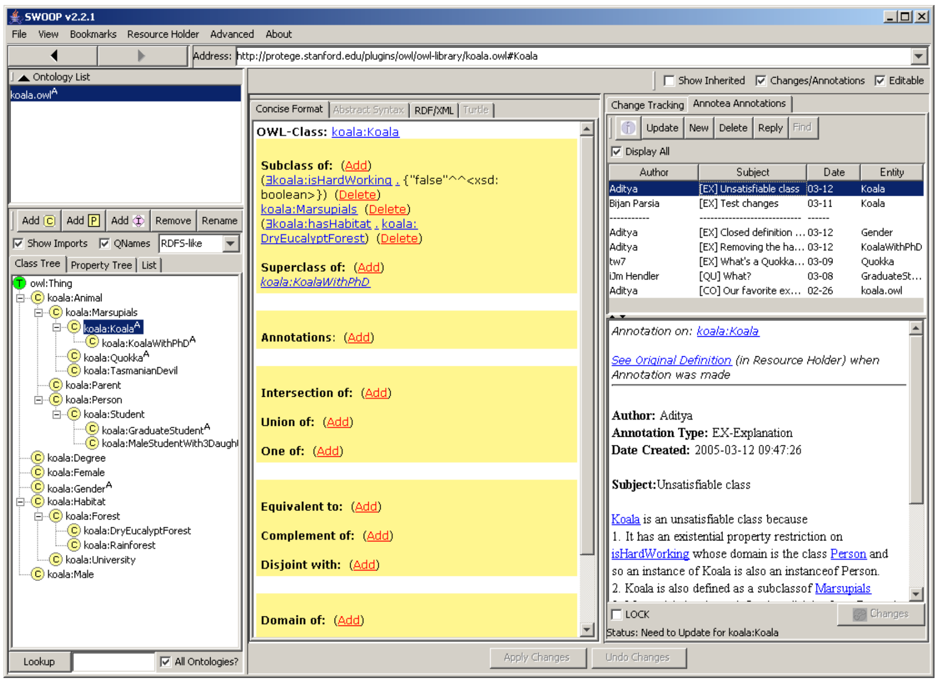
An underlying objective of the above tasks is to make full use of existing standards for representing, annotating and versioning ontological data, e.g., using versioning constructs of OWL such as `owl:versionInfo` `owl:priorVersion`, `owl:backwardCompatibility` and `owl:deprecated..` where necessary.

We note that work towards this feature is **still in progress**. For now, we describe the middle two components (listed above) that are fairly well developed and usable in their own right.

For **collaborative annotation** in Swoop, we use the Annotea framework [20], which takes the idea of separating annotations about ontologies from the core ontologies themselves and provides both a specific RDF based, extensible annotation vocabulary, and a protocol for publishing and finding out-of-band annotations (annotations that do not live inside the document being annotated).

Annotea support in Swoop is provided via a simple plug in whose implementation is based on the standard W3C Annotea protocols [21] and uses the default Annotea RDF schema to specify annotations (see **Fig. 10**). Any public Annotea Server can then be used to publish and distribute the annotations created in Swoop. The default annotation types (comment, advice, example, etc) seem an adequate base for human oriented ontology annotations. One extension we have begun experimenting with is "PrototypicalIllustration", that is, a photo or drawing that represents a typical or canonical instance of the class.

We have extended the Annotea Schema with the addition of an OWL ontology for a new class of annotations — ontology changes (similar to [22]). The "Change" annotation defined by the Annotea projected was designed to indicate a proposed change to the annotated document, with the proposal described in HTML-marked-up natural language. In our extended ontology,


SWOOP v2.2.1
File View Bookmarks Resource Holder Advanced About
Address: http://protege.stanford.edu/plugins/owl/owl-library/koala.owl#Koala
Ontology List
koala.owl
Add C Add P Add I Remove Rename
Show Imports QNames RDFS-like
Class Tree Property Tree List
owl:Thing
koala:Animal
koala:Marsupials
koala:Koala
koala:KoalaWithPhD
koala:Quokka
koala:TasmanianDevil
koala:Parent
koala:Person
koala:Student
koala:GraduateStudent
koala:MaleStudentWith3Daugh
koala:Degree
koala:Female
koala:Gender
koala:Habitat
koala:Forest
koala:DryEucalyptForest
koala:Rainforest
koala:University
koala:Male
Lookup All Ontologies?
Show Inherited Changes/Annotations Editable
Concise Format Abstract Syntax RDF/XML Turtle
Change Tracking Annotea Annotations
Update New Delete Reply Find
Display All
LOCK Changes
Apply Changes Undo Changes

OWL-Class: koala:Koala

Subclass of: (Add)
(∃koala:isHardWorking . {"false"^^<xsd:boolean>}) (Delete)
koala:Marsupials (Delete)
(∃koala:hasHabitat . koala:DryEucalyptForest) (Delete)

Superclass of: (Add)
koala:KoalaWithPhD

Annotations: (Add)

Intersection of: (Add)

Union of: (Add)

One of: (Add)

Equivalent to: (Add)

Complement of: (Add)

Disjoint with: (Add)

Domain of: (Add)

Author	Subject	Date	Entity
Aditya	[EX] Unsatisfiable class	03-12	Koala
Bijan Parsia	[EX] Test changes	03-11	Koala
-----------	---------------------------	------	
Aditya	[EX] Closed definition ...	03-12	Gender
Aditya	[EX] Removing the ha...	03-12	KoalaWithPhD
tw7	[EX] What's a Quokka...	03-09	Quokka
iJm Hendler	[QU] What?	03-08	GraduateSt...
Aditya	[CO] Our favorite ex...	02-26	koala.owl

Annotation on: koala:Koala

See Original Definition (in Resource Holder) when Annotation was made

Author: Aditya
Annotation Type: EX-Explanation
Date Created: 2005-03-12 09:47:26

Subject: Unsatisfiable class

Koala is an unsatisfiable class because
1. It has an existential property restriction on isHardWorking whose domain is the class Person and so an instance of Koala is also an instanceof Person.
2. Koala is also defined as a subclassof Marsupials

Status: Need to Update for koala:Koala

Fig. 10. **Annotea in Swoop**: Collaborative Annotation and Discussion of OWL Entities using the Annotea Client.

change individuals correspond to specific changes made in Swoop during editing. Note that Swoop uses the OWL API to model ontologies and their associated entities, benefiting from its extensive and clean support for changes. The OWL API separates the representation of changes from the application of changes. Each possible change type has a corresponding Java class in the API, which is subsequently applied to the ontology (essentially, the Command design pattern). These classes allow for the rich representation of changes, including metadata about the changes.

The Swoop change annotations can be published and retrieved by Annotea servers, or any other annotation distribution mechanism. The retrieved annotations can then be browsed, filtered, endorsed, recommended, and selectively accepted. A similar collaborative framework based on an interactive dialogue was implemented in a more local (tool-specific) context in the WebOnto system [9]. However, we decided to exchange annotations using the Annotea protocol to make the collaboration less tool-specific (any Annotea client can be used to discuss ontology annotations), and to allow users to arbitrarily extend the Annotea schema the way we have for ontology-change sets. These change sets also make it possible to define "virtual versions" of an ontology, by specifying a base ontology and a set of changes to apply to it (a feature present since the Swoop v2.2 release).

Note that in certain cases, changes may not be applicable to the ontology, if the change operation refers to an entity that is not present (defined) in the ontology. In such cases, a warning message is reported to the user describing the reason for the change conflict.

Finally, regarding **version control**, Swoop supports ad hoc undo/redo of changes (with logging) coupled with the ability to checkpoint and archive different ontology versions. Each change set or checkpoint can be saved at three different granularity levels - entity, ontology, workspace, which basically specify its *scope*. While the change logs can be used to explicitly track the evolution of an ontology[3], checkpoints allows the user to switch between versions directly exploring different modeling alternatives. Swoop also has the option to save checkpoints automatically, i.e., during specific tasks such as loading a new ontology, applying changes, removing or renaming an entity etc. Note that each time a checkpoint is saved, a *snapshot* of the entity definition is cached as well, and can be used to preview a checkpoint before reverting back to it.

The ontology evolution framework in Swoop is modeled on Smalltalks notion of change records and sets and is not as advanced as that in KAON [23], which has a special-purpose API for systematically controlling composite changes, applying change strategies and ensuring consistency during modification. In the future, we plan to learn from and improve the flexibility of the evolution scheme based on systems such as KAON.

5 Immediate Future Plans

Currently, search is implemented as a straightforward (sub)string matching algorithm that looks in entity name declarations, and works very fast since all indexing is native to Swoop. We are working on extending this naive search algorithm (and the associated indexing scheme) to handle more complex regular expressions and to look inside entity annotations (such as `rdfs:comment`). We are also investigating *class expression* search to support anonymous classes (b-nodes) in queries such as *'Find all classes which contain* $\exists p.C$*'*, and more advanced *structural* search to support queries such as *'Find all classes C, such that property* p_1, p_2, p_3 *have C as its domain'*. Using these various search routines, users can customize their queries to find highly specific or generic matches based on their interests.

We also plan on developing an **Advanced Resource Holder** that would feature automatic dynamic tracking for selected entities, color coding *diffs* between different entity definitions, and providing support for the editing of mapping terms, such as "owl:equivalentTo" between terms in different re-

[3] Change logs can also be serialized in RDF/XML and exchanged among users

source panes. Additionally, the Swoop v2.2 release has preliminary support for *sound and complete* conjunctive ABox queries written in RDQL [24]. We plan to extend this feature to include hybrid TBox/ABox queries as well as non-standard structural queries. Finally, since the OWL-API has support for SWRL [25], we plan to develop an intuitive UI and integrate a Rule Engine into Swoop for writing and processing rules.

6 Conclusion

In this paper, we describe Swoop, a *hypermedia inspired* Ontology Browser and Editor based on OWL, the first standardized Web-oriented ontology language. In designing Swoop, we take the familiar Web browser as our User Interface (UI) paradigm, focusing on a simple and intuitive ontology development interface. The primary goal has been to design a tool based on the successful architecture of the Web itself, i.e., its **open** (ad hoc local and remote multiple ontology support, various OWL Presentation syntax, extensible plugin architecture), **scalable** (e.g., ontologies as large as NCI [26] with over 27000 classes can be loaded in Swoop is under 2 minutes), and **distributed** (e.g., collaborative annotation support via Annotea). By doing the above, and remaining true to the OWL specifications, we allow the Swoop user to take advantage of the Web-based features of OWL.

Our secondary goal has been on overcoming the drawbacks, if any, of traditional ontology development tasks as applied to OWL. For instance, we have presented a partitioning approach for OWL ontologies to solve the *partial owl:imports* problem while facilitating ontology reuse. We have also presented work on guiding the user through the process of ontology debugging and repair (using explanations), where standard DL Reasoners do not provide additional help.

Finally, we are working on extending Swoop to further aid various forms of Web ontology development such as distributed, collaborative ontology evolution, the pieces of which already exist as separate Swoop components.

In this manner, with the strong platform Swoop provides for Web ontology development and its easy extensibility, Swoop is both, accessible and useful, for OWL developers and users.

7 Acknowledgments

The authors would wish to thank Ron Alford, Taowei David Wang, Vladimir Kolovski, Daniel Hewlett, and Michael Grove for their contribution in developing Swoop and the associated API's/plugins.

This work was completed with funding from Fujitsu Laboratories of America – College Park, Lockheed Martin Advanced Technology Laboratory, NTT Corp., Kevric Corp., SAIC, National Science Foundation, National Geospatial-Intelligence Agency, DARPA, US Army Research Laboratory, NIST, and other DoD sources.

References

[1] M. Dean, G. Schreiber, OWL Web Ontology Language Reference W3C Recommendation. http://www.w3.org/tr/owl-ref/.

[2] A. Kalyanpur, N. Hashmi, J. Golbeck, B. Parsia, Lifecycle of a casual web ontology development process, in: Proceedings of the WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, May 18, 2004, 2004.
URL `http://www.mindswap.org/ aditkal/WWW04`$_C$*OD.pdf*

[3] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, M. Musen, Creating semantic web contents with Protégé-2000, IEEE Intelligent Systems.

[4] S. Bechhofer, I. Horrocks, C. Goble, R. Stevens, OilEd: a reason-able ontology editor for the Semantic Web, Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence.

[5] Y. Sure, M. Erdmann, J. Angele, S. Staff, R. Studer, D. Wenke, OntoEdit: Collaborative ontology development for the Semantic Web, Proceedings of the International Semantic Web Conference (ISWC).

[6] A. Kalyanpur, B. Parsia, J. Hendler, A tool for working with web ontologies, in: In Proceedings of the International Journal on Semantic Web and Information Systems, Vol. 1, No. 1, Jan - March, 2005.
URL `http://www.mindswap.org/papers/Swoop-Journal.pdf`

[7] A. Farquhar, R. Fickas, J. Rice, The Ontolingua server: A tool for collaborative ontology construction, Proceedings of the 10th Banff Knowledge Acquisition for Knowledge Based System Workshop (KAW95).

[8] J. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez, WebODE: a scalable ontological engineering workbench, First International Conference on Knowledge Capture (K-CAP).

[9] J. Domingue, Tadzebao and webonto: Discussing, browsing, and editing ontologies on the web, in: 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, 1998.

[10] R. J. Erich Gamma, Richard Helm, J. Vlissides, Design patterns: Elements of reusable object-oriented software, addison-wesley.

[11] S. Bechhofer, P. Lord, R. Volz, Cooking the semantic web with the owl api, Proceedings of the International Semantic Web Conference.

[12] F. Manola, E. Miller, RDF Primer W3C Recommendation. http://www.w3.org/tr/rdf-primer/.

[13] P. Patel-Schneider, P. Hayes, I. Horrocks, OWL Web Ontology Language Semantics and Abstract Syntax W3C Recommendation. http://www.w3.org/tr/2004/rec-owl-semantics-20040210/.

[14] D. Beckett, Turtle — Terse RDF Triple Language. http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/.

[15] A. Kalyanpur, C. Halaschek-Wiener, V. Kolovski, J. Hendler, Effective nl paraphrasing of ontologies on the semantic web(Technical Report).
URL `http://www.mindswap.org/papers/nlpowl.pdf`

[16] B. Parsia, E. Sirin, Pellet: An owl dl reasoner (poster). http://www.mindswap.org/2003/pellet.

[17] V. Haarslev, R. Möller, Description of the racer system and its applications, in: Proceedings International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August, 2001, pp. 131–141.

[18] I. Horrocks, The FaCT system, 1998, pp. 307–312.
URL `download/1998/t98-paper.ps.gz`

[19] B. Parsia, E. Sirin, A. Kalyanpur, Debugging owl ontologies, in: The 14th International World Wide Web Conference (WWW2005), Chiba, Japan, 2005, to Appear.
URL `http://www.mindswap.org/papers/debuggingOWL.pdf`

[20] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, R. Swick, Annotea: An open RDF infrastructure for shared web annotations, Proc. of the WWW10 International Conference.

[21] R. Swick, E. Prud'Hommeaux, M.-R. Koivunen, J. Kahan, Annotea protocols, http://www.w3.org/2001/Annotea/User/Protocol.html.

[22] M. Klein, N. Noy, A component-based framework for ontology evolution, Workshop on Ontologies and Distributed Systems at IJCAI.

[23] L. Stojanovic, A. Maedche, B. Motik, N. Stojanovic, User-driven ontology evolution management, in: EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, Springer-Verlag, London, UK, 2002, pp. 285–300.

[24] H. L. B. Andy Seaborne, RDQL - A Query Language for RDF. http://www.w3.org/tr/owl-ref/.

[25] H. B. S. T. B. G. M. D. Ian Horrocks, Peter F. Patel-Schneider, SWRL: A Semantic Web Rule Language Combining OWL and RuleM. http://www.daml.org/2003/11/swrl/.

[26] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, P. B., J. Oberthaler, The national cancer institute's thesaurus and ontology, Journal of Web Semantics 1 (1). URL `http://www.mindswap.org/papers/WebSemantics-NCI.pdf`