The Protégé OWL Plugin

Holger Knublauch
Stanford University
July 07 2004
Overview

The Semantic Web and OWL

OWL in the Protégé Community

The OWL Plugin

Future Directions
The Semantic Web

Shared ontologies help to exchange data and meaning between web-based services

(Image by Jim Hendler)
OWL

- Web Ontology Language
- Official W3C Standard since Feb 2004

- A Web Language: Based on RDF(S)
- An Ontology Language: Based on logic
OWL Overview

- **Classes**
  - Subclass relationships
  - Disjoint classes

- **Properties**
  - Characteristics (transitive, inverse)
  - Range and Domain

- **Individuals**
  - Property values

- **Class Descriptions**
  - Restrictions
  - Logical expressions

OWL for data exchange

OWL for knowledge sharing
Example Semantic Web

- Tourism Ontology
  - Destination
  - Activity
  - Accommodation

- Web Services

- OWL Metadata (Individuals)
  - Destination
  - Activity
  - Accommodation

- OWL Metadata (Individuals)
Description Logics

• Classes can be defined using logical expressions about their members.

Restrictions
- allValuesFrom
- someValuesFrom
- hasValue
- minCardinality
- maxCardinality
- cardinality

Logical Expressions
- unionOf
- intersectionOf
- complementOf

Enumerations
- \{red, green, blue\}
Description Logics Example

- Asserted definitions:
  - “National parks provide hiking trails”
  - “Hiking is a sport”
  - “Those destinations with sporting facilities are backpacker’s destinations”

- Automatically inferred:
  - National parks are backpacker’s destinations
Class Descriptions: Why?

• Make knowledge sharable with machines
• Make explicit intentions and modeling decisions (comparable to test cases)
• Make sure that individuals fulfill conditions
• Tool-supported reasoning
  – Classification of classes and individuals
  – Consistency Checking
Protégé OWL Community

Manchester: CO-ODE tools

Stanford: OWL Plugin

NCI, NLM: Main funding
OWL Plugin

• Large Protégé Plugin (>560 classes)
• Extends base system with
  – OWL language capabilities (metamodel, files)
  – Custom-tailored user interface
  – Access to description logic reasoners
  – Code generators etc
• Many features are native to OWL
• Backwards compatible where possible
OWL Plugin Architecture

OWL File Storage → Jena API (Parsing, Reasoning)

Jena API (Parsing, Reasoning) → Protégé API (Classes, properties, individuals, etc.)

Protégé API (Classes, properties, individuals, etc.) → Protégé OWL API (Logical class definitions, restrictions, etc.)

Protégé OWL API (Logical class definitions, restrictions, etc.) → OWL Extension APIs (SWRL, OWL-S, etc.)

OWL Extension APIs (SWRL, OWL-S, etc.) → Protégé OWL GUI (Expression Editor, Conditions Widget, etc.)

Protégé OWL GUI (Expression Editor, Conditions Widget, etc.) → Protégé GUI (Tabs, Widgets, Menus)

Protégé GUI (Tabs, Widgets, Menus) → Protégé Core System

Protégé Core System → OWL Plugin

OWL GUI Plugins (SWRL Editors, ezOWL, OWLViz, Wizards, etc.) → Protégé OWL GUI (Expression Editor, Conditions Widget, etc.)
Logic View

In the image, a screenshot of the Protege 2.1.1 software is shown, displaying a view of a travel ontology. The ontology includes classes such as BackpackersDestination, Accommodation, BedAndBreakfast, BudgetAccommodation, and Hotel. The interface shows asserted hierarchy, relationships, properties, and conditions for the classes. For example, BackpackersDestination is defined as a destination that provides budget accommodation and offers sport or adventure activities. The software provides a visual representation of the relationships and properties among these classes, making it easier to understand the structure of the travel ontology.
Prose Generation

![Diagram showing asserted conditions]

- **Destination**
  - NECESSARY & SUFFICIENT
  - Is not a family destination

- **Destination** and
  - Exists hasAccomodation
  - Exists BudgetAccomodation
  - exists hasActivity (Sports & Adventure)

  - Has a budget accommodation as its accomodation
Properties View
Classification

The image shows a screenshot of a software interface named Protégé 2.1.1, which is used for creating and editing ontologies. The interface displays a classification hierarchy with various classes and sub-classes, including:

- NationalPark
- Campground
- Capital
- RuralArea
- UrbanArea
- Activity
- Accommodation
- BudgetAccommodation
- LuxuryHotel
- AccommodationRating
- Destination
- BackpackersDestination
- FamilyDestination
- QuietDestination
- RetireeDestination

The interface also highlights changes and moved superclasses, indicating that:

- Campground has been moved from Accommodation to BudgetAccommodation.
- Capital has been added as a RetireeDestination.
- NationalPark has been added as a BackpackersDestination.

The software appears to be used for organizing and classifying data in an ontology format, likely for travel destinations.
Consistency Checking
TODO Lists
Individuals
RDF(S) Editing

- OWL extends RDF, OWL Plugin can edit RDF
- Select Language Profile RDF(S)
Code Generators

• Create Java classes to easier access OWL ontologies
  – Jena Schema generator
  – Kazuki interfaces

• Better integration into other software development activities
Extending the OWL Plugin
Other OWL Plugins

OWLViz (Manchester)

ezOWL (Korea)

Under Development:
- OWL-S
- SWRL
- Semantic Debugging
- Joseki (Jena Database)
Integrating Web Services

• Arbitrary extensions can benefit from the Protégé infrastructure as a platform
OWL Benefits

• OWL is the W3C ontology standard
• Interoperable with RDF and XML
• Growing community and tool support
• OWL has formal semantics and built-in reasoning support
• Semantics support maintenance of large ontologies / knowledge models
Protégé OWL Benefits

• A de facto standard tool in the OWL world
• Growing number of plugins / adaptations
• Custom-tailored Open-Source API
• Online support
• Robust platform
• Compatibility with Jena
• Scalable (Database backend)
OWL Risks

• Steep learning curve
  – Difficult syntax
  – Difficult semantics (→ OWL Tutorial)

• Semantic Web hype /
  Lack of real-world example applications

• Chicken-and-egg problem of Semantic Web
Getting Started

Protégé OWL Plugin
Ontology Editor for the Semantic Web

Documentation

Online Documents about the OWL Plugin

- OWL Plugin Frequently Asked Questions
- The Protégé OWL Tutorial by Matthew Horridge
- Slides from another Protégé OWL Tutorial by Holger Knublauch (from the Protégé conference July 2004)
- A small and partly outdated online Tutorial (using foaf ontology)
- Mostly for programmers: How to configure the OWL Plugin
- For programmers: How to develop Plugins
- For programmers: JavaDoc API

Publications about the OWL Plugin

- Holger Knublauch, Olivier Damerow, Mark A. Musen
  Weaving the Biomedical Semantic Web with the Protégé OWL Plugin
  First International Workshop on Formal Biomedical Knowledge Representation,
  Whistler, BC, Canada (2004) – An introduction for users and domain modelers

- Holger Knublauch, Mark A. Musen, Alan L. Rector
  Editing Description Logic Ontologies with the Protégé OWL Plugin
  International Workshop on Description Logics – DL2004,
  Whistler, BC, Canada (2004) – A technical discussion for logicians

- Holger Knublauch, Ray W. Ferguson, Natasha F. Noy, Mark A. Musen
  The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications
  Submitted for publication (2004) – An architectural overview for developers and decision-makers
OWL Tutorial (Manchester)

http://www.co-ode.org/resources/

Figure 4.5: The Disjoint Classes Widget

located in the lower right hand corner of the 'OWL Classes' tab is used. (See Figure 4.5).

Exercise 4: Make Pizza, PizzaTopping and PizzaBase disjoint from each other

1. Select the class Pizza in the class hierarchy.
2. Press the 'Add siblings' button on the disjoint classes widget. This will make PizzaTopping and PizzaBase (the sibling classes of Pizza) disjoint from Pizza.

Notice that the disjoint classes widget now displays PizzaTopping and PizzaBase. Select the class PizzaBase. Notice that the disjoint classes widget displays the classes that are now disjoint to PizzaBase, namely Pizza and PizzaTopping.
Outlook

• OWL will remain a focus at Stanford
  – Simpler user interface
  – Better Workflow/Versioning support
  – Optimized access to reasoners (RACER)
  – Numeric range restrictions
  – Access to UML/Model Driven Architecture
• More plugins will be available
• Collaborations?