

Ontology Editors

IDEs for Ontologies

- Some people use simple text editors
 - Working with XML serialization will drive you crazy
 - Using Turtle or an abstract syntax works well
- Others prefer an IDE
 - Good IDEs include support for reasoning, visualization, and more
- Protégé is a very popular IDE
 - From Stanford, free, lots of plugins
- TopQuadrant Composer is also good
 - Feature rich but expensive (\$600 for a single license)

Protégé 5.5

The screenshot displays the Protégé 5.5 web interface. At the top, the browser address bar shows the URL `http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122`. Below the address bar, a search bar contains the text `untitled-ontology-122`. The main interface is divided into several sections:

- Active Ontology:** A tabbed menu with options: `Active Ontology`, `Entities`, `Classes`, `Object Properties`, `Data Properties`, `Individuals by class`, `OWL Viz`, and `DL Query`.
- Annotations:** A tabbed menu with options: `Annotations`, `Selected entailments`, and `Rules`.
- Ontology header:** A purple header section containing:
 - Ontology IRI:** `http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122`
 - Ontology Version IRI:** `e.g. http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122/1.0.0`
- Annotations:** A large white area with a `+` icon, currently empty.
- Ontology imports:** A tabbed menu with options: `Ontology imports`, `General axioms`, `RDF/XML rendering`, `OWL/XML rendering`, and `OWL functional syntax rendering`.
- Imported ontologies:** A purple header section containing:
 - Direct Imports:** A white area with a `+` icon, currently empty.
 - Indirect Imports:** A white area, currently empty.

At the bottom of the interface, a footer message reads: `To use the reasoner click Reasoner > Start reasoner` with a checked checkbox for `Show Inferences`.

Protégé 5.5

- <http://protege.stanford.edu/>
- Free, open source ontology editor and KB framework
- Predates OWL, still supports earlier Frames representation
- In Java, extensible, large community of users
 - Requires Java Runtime Environment
- [Desktop](#) and [Web](#) versions
 - Works will under Linux, Mac OS X and Windows

Desktop Protégé

The screenshot displays the Protégé desktop application interface for editing the 'peeps' ontology. The browser address bar shows the URL: `peeps (http://ebiq.org/ontologies/peeps/) : [Users/finin/Sites/691f17/examples/owl_examples/peeps/peeps.owl]`. The main window contains several panes:

- Class hierarchy (inferred):** Shows a tree structure starting with `owl:Thing`, which includes `Person`. Under `Person`, there are subclasses `Adult`, `Man`, and `Minor`. `Man` has subclasses `Boy` and `Woman`. The `Man` class is currently selected and highlighted in blue.
- Annotations: Man:** Displays the class annotations for `Man`. It includes:
 - `rdfs:label` with the value "Male person".
 - `rdfs:comment` with the value "A Man is defined as a person with a has_sex value equal to 'male'".
- Description: Man:** Shows the logical description of the `Man` class:
 - Equivalent To: `Person and (hasSex value "male")`
 - SubClass Of: `Person`
 - General class axioms:
 - `hasParent exactly 1 Man`
 - `hasParent exactly 1 Woman`
- Object property:** Shows the `owl:topObjectProp` property.

At the bottom right of the interface, there is a status bar with the text: "To use the reasoner click Reasoner > Start reasoner" and a checked checkbox for "Show Inferences".

Web Protégé

The screenshot displays the WebProtégé web interface in a browser window. The browser's address bar shows the URL `webprotege.stanford.edu/#Edit:...`. The interface includes a top navigation bar with the Protégé logo, a 'Project' dropdown menu, 'Share' and 'Help' buttons, and a user profile for 'Tim Finin'. Below this is a tabbed interface with the active tab labeled 'UMBC691PeepsExample'. A secondary navigation bar contains tabs for 'Classes', 'Properties', 'Individuals', 'Notes and Discussions', 'Changes By Entity', and 'Project Dashboard'. A toolbar at the top right offers options to 'Add content to this tab' and 'Add tab'. The main workspace is divided into three panels:

- Classes:** A tree view on the left showing a hierarchy starting with 'owl:Thing', which includes 'Person' and 'Sex'.
- Class description for Person:** A central panel with the following fields:
 - Display name:** Person
 - IRI:** `http://webprotege.stanford.edu/RCcwwdIsdJMKB`
 - Annotations:** A table with one row:

rdfs:label	Person	lang
------------	--------	------

 Below this is an input field for 'Enter property' and 'Enter' buttons.
 - Properties:** An input field for 'Enter property' and 'Ent lang' buttons.
- Discussions for Person:** A panel on the right with a 'Post new topic...' button.

YAS: Yet Another Syntax

- Neither OWL's official abstract syntax nor XML serialization is easy to read or use
- Protégé uses the Manchester syntax
- Simpler and more compact: “some” and “only”, not “someValuesFrom” and “allValuesFrom”
- A W3C recommendation (<http://bit.ly/manSyn>), used in the OWL 2 Primer (<http://bit.ly/OWL2Pri>)

Class: man

Annotations: rdfs:label "man"

EquivalentTo: adult and male and person

Manchester OWL syntax

OWL	DL Symbol	Manchester OWL Syntax Keyword	Example
someValuesFrom	\exists	some	hasChild some Man
allValuesFrom	\forall	only	hasSibling only Woman
hasValue	\ni	value	hasCountryOfOrigin value England
minCardinality	\geq	min	hasChild min 3
cardinality	$=$	exactly	hasChild exactly 3
maxCardinality	\leq	max	hasChild max 3

Manchester OWL syntax

OWL	DL Symbol	Manchester OWL Syntax Keyword	Example
intersectionOf	\sqcap	and	Doctor and Female
unionOf	\sqcup	or	Man or Woman
complementOf	\neg	not	not Child

Example 1

How can we define a class that is people who have children and all of them are male?

Example: People with just boys

How can we define a class that is people who have children and all of them are male?

Define as the union of three classes

1. Person
2. Things that have children
3. Things where all of their children are male

Example: People with just boys

How can we define a class that is people who have children and all of them are male?

Define as the union of three classes

1. Person
2. Things that have children
3. Things where all of their children are male

An `owl:someValuesFrom` restriction on `hasChild` property

An `owl:allValuesFrom` restriction on `hasChild` property

Example: People with just boys

How can we define a class that is people who have children and all of them are male?

```
Person and  
  (hasChild only Man) and  
  (hasChild some Person)
```

Example 2

```
Person and
  hasChild some
    (Person and
      (hasChild only Man) and
        (hasChild some Person) )
```

The set of people who have at least one child that has some children that are only men (i.e., grandparents that only have a Pjb)

Data values and datatypes

- Data values typed or untyped (e.g., int, boolean, float)
- Constants w/ or w/o type, e.g.: hasAge value "21"^^long
- Use datatype names as classes: hasAge some int
- XSD facets, e.g.: Person and hasAge some int[>= 65]
- Ranges: Person and hasAge some int[>= 18, <= 30]

XSD facet	Meaning
< x, <= x	less than, less than or equal to x (more info)
> x, >= x	greater than, greater than or equal to x (more info)
length x	For strings, the number of characters must be equal to x (more info)
maxLength x	For strings, the number of characters must be less than or equal to x (more info)
minLength x	For strings, the number of characters must be greater than or equal to x (more info)
pattern regexp	The lexical representation of the value must match the regular expression, regexp (more info)
totalDigits x	Number can be expressed in x characters (more info)
fractionDigits x	Part of the number to the right of the decimal place can be expressed in x characters (more info)

Demonstration

- We'll use Protégé OWL v5.5 to implement a tiny ontology for people
- Start by downloading and installing Protégé 5.5
(You will need the JRE installed)
- You may want to install Graphviz
- Configure Protégé
 - E.g., select a reasoner to use (e.g., Hermit)

A basic workflow

- Think about usecases
- Preliminaries
 - Choose namespace URL, import other ontologies used
- Identify and define classes
 - Place in hierarchy, add **axioms** and run reasoner to check for errors or omissions
- Identify and define properties
 - Place in hierarchy, add **axioms**, run reasoner
- Add individuals & reasoner to check for problems
- Add comments and labels
- Export in desired formats, maybe upload to Web

More workflow steps

- Use [OOPS](#) to find common ontology pitfalls
 - Ontology Pitfall Scanner detect many common pitfalls introduced when developing ontologies
- Link concepts (and individuals) to common ontologies (e.g., DBpedia, Freebase, foaf)
 - Use owl:sameAs
- Generate visualizations
- Produce documentation
- Develop examples with your use case(s)
- Encode data, describe in [VoID](#) (Vocabulary of Interlinked Datasets), add to LOD cloud

<http://oops.linkeddata.es/>

← → ↻ 🏠 ⓘ Not Secure | oops.linkeddata.es/response.jsp# ☆ 📶 🌐 ⓘ 🗑️ 📄 📱 📧 ⋮

Ontology Pitfall Scanner!

OOPS! (Ontology Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI: Scanner by URI

Example: http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf

Scanner by direct input: Scanner by RDF

Uncheck this checkbox if you don't want us to keep a copy of your ontology. [Go to advanced evaluation](#)

Evaluation results

It is obvious that not all the pitfalls are equally important; their impact in the ontology will depend on multiple factors. For this reason, each pitfall has an importance level attached indicating how important it is. We have identified three levels:

- **Critical** 🚫 : It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important** ⚠️ : Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor** 🟡 : It is not really a problem, but by correcting it we will make the ontology nicer.

[\[Expand All\]](#) | [\[Collapse All\]](#)

Results for P08: Missing annotations.	11 cases Minor 🟡
Results for P11: Missing domain or range in properties.	1 case Important ⚠️
Results for P13: Inverse relationships not explicitly declared.	2 cases Minor 🟡
Results for P34: Untyped class.	7 cases Important ⚠️
Results for P36: URI contains file extension.	ontology* Minor 🟡
Results for P38: No OWL ontology declaration.	ontology* Important ⚠️
Results for P41: No license declared.	ontology* Important ⚠️
SUGGESTION: symmetric or transitive object properties.	2 cases

Want to help?

- Suggest new pitfalls
- Provide feedback

Documentation:

- Pitfall catalogue
- User guide
- Technical report

Related papers:

- IJSWIS 2014
- EKAW 2012
- ESWC 2012 Demo
- Ontoqual 2010
- CAEPIA 2009

What to watch out for

- After editing your ontology or data you should (1) stop the reasoner and (2) run it again
- Look for any of the following problems
 - Unexpected inferences
 - Missing inferences
 - Reasoner stops with an error
 - Reasoner stops after finding a contradiction
 - Reasoner concludes a class is equivalent to owl:Nothing

Error: Impossible Class

The screenshot shows a web-based ontology editor interface. The browser address bar displays the URL `https://raw.githubusercontent.com/finin/master/mypeeps.ttl`. The page title is `mypeeps.ttl`. The interface includes a navigation menu with tabs for `Active ontology`, `Entities`, `Classes`, `Object properties`, `Data properties`, and `Individuals by class`. The `Classes` tab is active, showing a class hierarchy on the left and a detailed view of the `Foo` class on the right.

The class hierarchy on the left shows the following structure:

- owl:Thing
 - foaf:Person
 - Person
 - Foo** (highlighted)
 - Adult
 - Man
 - Minor
 - Woman
 - schema:Person

The detailed view of the `Foo` class shows the following properties:

- Description: Foo**
- Equivalent To:** owl:Nothing (highlighted with a red circle)
- SubClass Of:** Man, Person, Woman
- General class axioms:** hasParent exactly 1 Woman, foaf:Person, schema:Person, hasParent exactly 1 Man, Person and (hasSex value "male")

The interface also includes a search bar, a status bar at the bottom indicating `Git: master` and `Reasoner active`, and a `Show Inferences` checkbox.

Inconsistent Ontology

The screenshot displays a web-based ontology editor interface. The main window shows the ontology hierarchy for `hasChild`. A red oval highlights the `hasChild` property in the hierarchy and the `Annotations: hasChild` section, which states: "Asserts that the subject has a child who is the object; it is the inverse of `hasParent`".

An error dialog box is overlaid on the interface, titled "An error occurred during reasoning". The message reads: "InconsistentOntologyException: Cannot do reasoning with inconsistent ontologies! Reason for inconsistency: Irreflexive property `inv(hasChild)`".

The dialog box has an "OK" button. In the bottom right corner of the editor, there is a status bar that says "Reasoner active but the ontology is inconsistent" and a "Show Inferences" checkbox, which is also circled in red.

Reasoner fails

The screenshot shows a web-based ontology editor interface. The main content area displays the configuration for the property `hasChild`. The `Annotations` section shows an `rdfs:comment` stating: "Asserts that the subject has a child who is the object; it is the inverse of hasParent". The `Description` section shows the following settings:

- Functional
- Inverse functional
- Transitive
- Symmetric
- Asymmetric
- Reflexive
- Irreflexive

Below the description, there are sections for `Equivalent To`, `SubProperty Of`, `Inverse Of` (set to `hasParent`), `Domains (intersection)` (set to `Person`), `Ranges (intersection)` (set to `Person`), `Disjoint With`, and `SuperProperty Of (Chain)`.

At the bottom right, the `Show inferences` checkbox is checked and circled in red. The status bar at the bottom indicates "Git: master" and "To use the reasoner click Reasoner > Start reasoner".

Reasoner fails

mypeeps.ttl (https://raw.githubusercontent.com/finin/master/mypeeps.ttl) : [/Users/finin/Teaching/691/peeps/mypeeps.ttl]

Log

```
INFO 15:16:55 [GitRepo] Git repository detected: /Users/finin/Teaching/691/peeps/.git
INFO 15:16:55 [GitRepo] On branch: master
INFO 15:16:59 ----- Running Reasoner -----
INFO 15:16:59
ERROR 15:16:59 An error occurred during reasoning: Axiom: TransitiveObjectProperty(<https://raw.githubusercontent.com/finin/peeps/master/peeps.ttl#hasChild>),
org.mindswap.pellet.exceptions.UnsupportedFeatureException: Axiom: TransitiveObjectProperty(<https://raw.githubusercontent.com/finin/peeps/master/peeps.ttl#hasChild>)
  at com.clarkparsia.pellet.owlapiv3.PelletVisitor.addUnsupportedAxiom(PelletVisitor.java:119) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletVisitor.verify(PelletVisitor.java:160) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasoner.refresh(PelletReasoner.java:969) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasoner.<init>(PelletReasoner.java:345) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasonerFactory.createReasoner(PelletReasonerFactory.java:69) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasonerFactory.createReasoner(PelletReasonerFactory.java:33) ~[na:na]
  at org.protege.editor.owl.model.inference.ReasonerUtilities.createReasoner(ReasonerUtilities.java:20) ~[na:na]
  at org.protege.editor.owl.model.inference.OWLReasonerManagerImpl$ClassificationRunner.ensureRunningReasonerInitialized(OWLReasonerManagerImpl.java:428) ~[na:na]
  at org.protege.editor.owl.model.inference.OWLReasonerManagerImpl$ClassificationRunner.run(OWLReasonerManagerImpl.java:386) ~[na:na]
  at java.lang.Thread.run(Thread.java:745) ~[na:1.8.0_121]
```

Show log file Preferences Time stamp Clear log

OK

Disjoint With +

SuperProperty Of (Chain) +

Git: master

To use the reasoner click Reasoner > Start reasoner Show Inferences

Reasoner fails

mypeeps.ttl (https://raw.githubusercontent.com/finin/master/mypeeps.ttl) : [/Users/finin/Teaching/691/peeps/mypeeps.ttl]

Log

```
INFO 15:16:55 [GitRepo] Git repository detected: /Users/finin/Teaching/691/peeps/.git
INFO 15:16:55 [GitRepo] On branch: master
INFO 15:16:59 ----- Running Reasoner -----
INFO 15:16:59
ERROR 15:16:59 An error occurred during reasoning: Axiom: TransitiveObjectProperty(<https://raw.githubusercontent.com/finin/peeps/master/peeps.ttl#hasChild>),
org.mindswap.pellet.exceptions.UnsupportedFeatureException: Axiom: TransitiveObjectProperty(<https://raw.githubusercontent.com/finin/peeps/master/peeps.ttl#hasChild>)
  at com.clarkparsia.pellet.owlapiv3.PelletVisitor.addUnsupportedAxiom(PelletVisitor.java:119) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletVisitor.verify(PelletVisitor.java:160) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasoner.refresh(PelletReasoner.java:969) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasoner.<init>(PelletReasoner.java:345) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasonerFactory.createReasoner(PelletReasonerFactory.java:69) ~[na:na]
  at com.clarkparsia.pellet.owlapiv3.PelletReasonerFactory.createReasoner(PelletReasonerFactory.java:33) ~[na:na]
  at org.protege.editor.owl.model.inference.ReasonerUtilities.createReasoner(ReasonerUtilities.java:20) ~[na:na]
  at org.protege.editor.owl.model.inference.OwlReasonerManagerImpl$ClassificationRunner.ensureRunningReasonerInitialized(OwlReasonerManagerImpl.java:428) ~[na:na]
  at org.protege.editor.owl.model.inference.OwlReasonerManagerImpl$ClassificationRunner.run(OwlReasonerManagerImpl.java:386) ~[na:na]
  at java.lang.Thread.run(Thread.java:745) ~[na:1.8.0_121]
```

- Owl reasoners don't like a property to be transitive and irreflexive or asymmetric
- Restriction is necessary in order to guarantee decidability of reasoning problems for OWL 2 DL
- You can make a property transitive via rules as a work around

Show log file Preferences

SuperProperty Of (Chain) +

Git: master To use the reasoner click Reasoner > Start reasoner Show Inferences

Demonstration/HW4

Use Protégé OWL (v5.5) to build a simple ontology for people based on the following

- People have just one sex that's either *male* or *female*, an integer age, and two parents, one male, one female
- A person's grandparent is the parent of their parent
- Every person is either a man or a woman but not both
- A man is defined as any person whose sex is male and a woman as any person whose sex is female
- A boy is defined as a person whose sex is male and whose age is less than 18, a girl is ...
- A person is either an adult or (age >18), minor (age <18), ...

Test cases

All Different people

Alice F

Bob M

Carol F

Don M

Edith F

Pat ?

Other people

Frank M

Gwen F

Some possible test cases

- Alice parent Bob . Bob parent Carol
 - Alice grandparent Carol
- Alice parent Bob . Alice parent Don.
 - Contradiction
- Alice parent Bob . Pat parent Bob
 - Pat a female
- Alice parent Bob . Gwen parent Bob .
 - Alice owl:sameAs Gwen