



# Ontology Editors (1)

# 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 support for reasoning, viZ, and more
- Protégé is a very popular IDE
  - From Stanford, free, lots of plugins
- The W3C has a page on Ontology editors
  - Somewhat outdated, tho

# Protégé 5.5



untitled-ontology-122 (http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122) : [http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122]

untitled-ontology-122 (http://www.semanticweb.org/ontologies/2016/9/untitled-ontology-122) Search...

Active Ontology x Entities x Classes x Object Properties x Data Properties x Individuals by class x OWLViz x DL Query x

Annotations Selected entailments Rules

Ontology header: ⌵ ⌵ ⌵ ⌵

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 +

Ontology imports General axioms RDF/XML rendering OWL/XML rendering OWL functional syntax rendering

Imported ontologies: ⌵ ⌵ ⌵ ⌵

Direct Imports +

Indirect Imports

To use the reasoner click Reasoner > Start reasoner  Show Inferences

# Protégé 5.5

- <http://protege.stanford.edu/>
- Free, open-source ontology editor and KB framework
- Early versions predate OWL, so still supports earlier Frames representations
- 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 `http://ebiq.org/ontologies/peeps/`. The main window is titled 'peeps (http://ebiq.org/ontologies/peeps/)' and contains several tabs: 'Active Ontology', 'Entities', 'Individuals by class', 'DL Query', 'Individual Hierarchy Tab', and 'OWLviz'. The 'Class hierarchy' tab is active, showing a tree view of classes: `owl:Thing` (parent), `Person` (child), `Adult` (child of Person), `Man` (child of Person, highlighted in blue), `Boy` (child of Man), `Minor` (child of Person), `Boy` (child of Minor), and `Woman` (child of Person). The 'Annotations: Man' panel shows two annotations: `rdfs:label` with the value 'Male person' and `rdfs:comment` with the value 'A Man is defined as a person with a has sex value equal to "male"'. The 'Description: Man' panel shows the class description: `Equivalent To` `Person and (hasSex value "male")`. The 'SubClass Of' panel shows that `Man` is a subclass of `Person` and has two general class axioms: `hasParent exactly 1 Man` and `hasParent exactly 1 Woman`. A tooltip indicates that the description is asserted in `http://ebiq.org/ont`. The bottom status bar includes 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 application interface. At the top, the browser address bar shows the URL `webprotege.stanford.edu/#Edit:...`. The application header includes the Protégé logo and navigation buttons for **Project**, **Share**, **Tim Finin**, and **Help**. Below the header, a breadcrumb trail shows **WebProtégé** > **UMBC691PeepsExample**. A horizontal menu contains tabs for **Classes**, **Properties**, **Individuals**, **Notes and Discussions**, **Changes By Entity**, and **Project Dashboard**. A toolbar on the right offers **Add content to this tab** and **Add tab** options.

The main workspace is divided into three panels:

- Classes Panel:** Shows a tree view of classes under **owl:Thing**, including **Person** and **Sex**. Buttons for **Create**, **Delete**, and **Watch Br** are visible.
- Class description for Person Panel:** Contains fields for **Display name** (Person) and **IRI** (`http://webprotege.stanford.edu/RCcwwdIsdJMKB`). The **Annotations** section shows **rdfs:label** with a dropdown menu set to **Person** and a **lang** field. Below it are input fields for **Enter property** and **Enter** with a **lang** field. The **Properties** section has an **Enter property** field and **Ent** and **lang** fields.
- Discussions for Person Panel:** Features 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, used in the [OWL 2 Primer](#)
- Example:

```
Class: man
```

```
Annotations: rdfs:label "man"
```

```
EquivalentTo: adult and male and person
```

# Manchester OWL syntax (1)

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



# Manchester OWL syntax (2)

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

# Example: People with just boys

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

Do we really need need 2 and 3?

# 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 has a child with children who are all male

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

# 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)

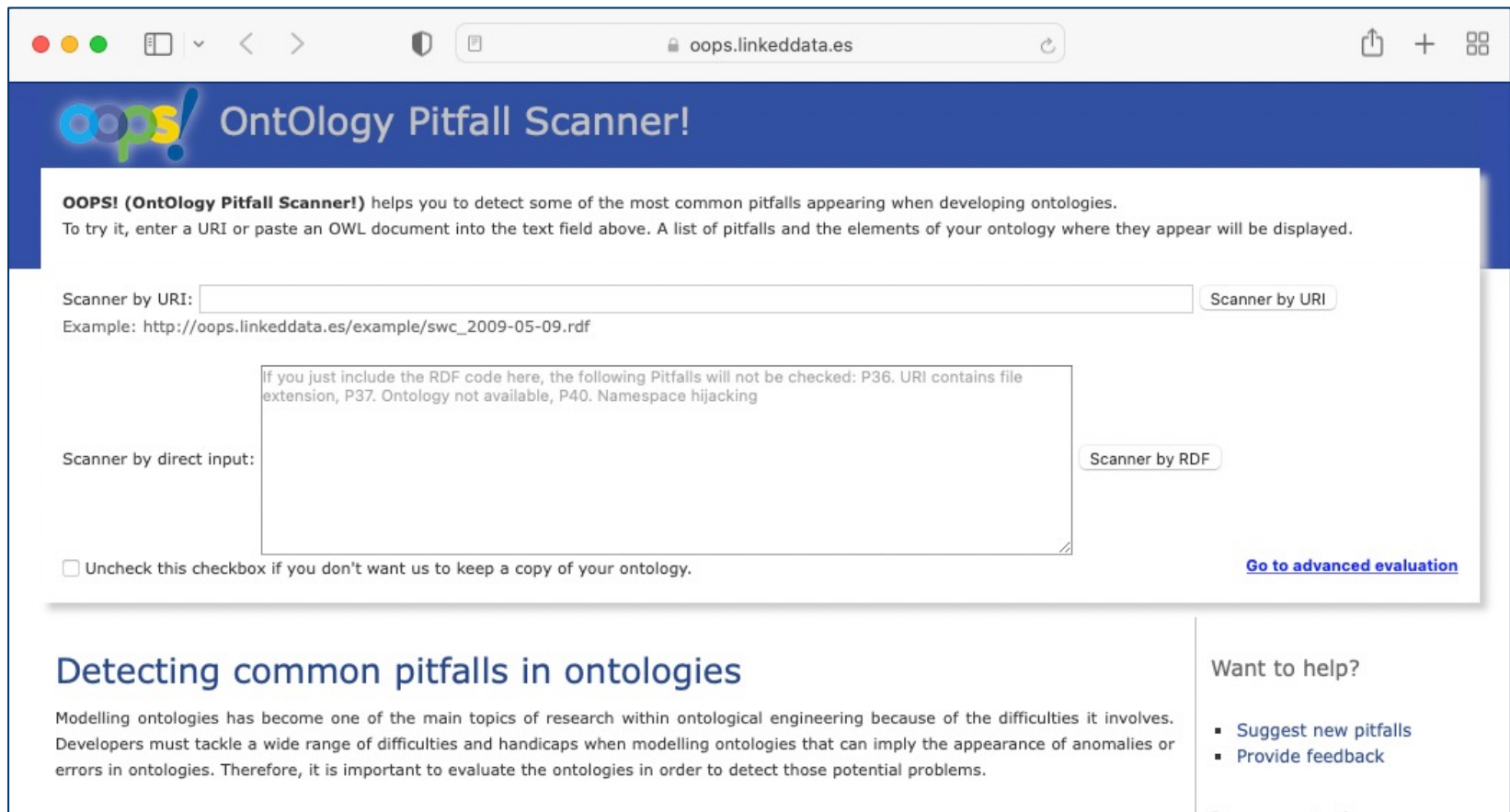


# A basic workflow

- 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 problems introduced when developing ontologies



The screenshot shows the OOPS! Ontology Pitfall Scanner web application. The browser address bar displays `oops.linkeddata.es`. The page header features the OOPS! logo and the title "Ontology Pitfall Scanner!".

The main content area contains the following text:

**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://oops.linkeddata.es/example/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](#)

A note in a text box states: "If you just include the RDF code here, the following Pitfalls will not be checked: P36. URI contains file extension, P37. Ontology not available, P40. Namespace hijacking"

**Detecting common pitfalls in ontologies**

Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

**Want to help?**

- Suggest new pitfalls
- Provide feedback

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

← → ↻ 🏠 ⓘ Not Secure | [oops.linkeddata.es/response.jsp#](http://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](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](#)

# More workflow steps

- Link concepts (and individuals) to common ontologies (e.g., DBpedia, Freebase, foaf)
  - Use owl:sameAs to connect *identical* concepts or properties
  - E.g.: peeps:Person owl:sameAs foaf:Person
- Generate visualizations
- Produce documentation
- Develop examples with your use case(s)
- Encode data, describe in [VoID](#) (Vocabulary of Interlinked Datasets), add to [LOD cloud](#)

# 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 `owl:Thing` as the root, with `foaf:Person` and `Person` as subclasses. `Person` has several subclasses: `Adult`, `Man`, `Minor`, and `Woman`. `schema:Person` is also listed. The `Foo` class is highlighted in blue in the hierarchy.


The detailed view of the `Foo` class shows its description, which is `Equivalent To owl:Nothing`. This is circled in red, indicating an impossible class. Below the description, the `SubClass Of` section lists `Man`, `Person`, and `Woman`. The `General class axioms` section lists `hasParent exactly 1 Woman`, `foaf:Person`, `schema:Person`, `hasParent exactly 1 Man`, and `Person`.

The bottom of the interface shows `Git: master` on the left and `Reasoner active` and `Show Inferences` on the right.

# Inconsistent Ontology

The screenshot displays the Protege interface for an ontology named 'mypeeps.ttl'. The main window shows the 'Object property hierarchy: hasChild' and the 'Annotations: hasChild' tab. A red oval highlights the 'Annotations: hasChild' tab and the error dialog box. The error dialog box contains the following text:

An error occurred during reasoning

 InconsistentOntologyException: Cannot do reasoning with inconsistent ontologies!  
Reason for inconsistency: Irreflexive property inv(hasChild)

Below the error dialog, the 'Annotations: hasChild' tab shows the following properties:

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

The 'Inverse Of' property is set to 'hasParent'. The 'Domains (intersection)' and 'Ranges (intersection)' are both set to 'Person'. The 'Disjoint With' and 'SuperProperty Of (Chain)' properties are also visible.

At the bottom of the interface, the status bar shows 'Git: master' and 'Reasoner active but the ontology is inconsistent'. A red oval highlights the 'Show Inferences' checkbox and the warning icon in the status bar.

# Reasoner fails

The screenshot shows a web-based ontology editor interface. The browser address bar displays the URL: `mypeeps.ttl (https://raw.githubusercontent.com/finin/master/mypeeps.ttl) : [/Users/finin/Teaching/691/peeps/mypeeps.ttl]`. The page title is `mypeeps.ttl (https://raw.githubusercontent.com/finin/master/mypeeps.ttl)`. The main content area is titled `Object property hierarchy: hasChild` and shows a tree view of properties under `owl:topObjectProperty`, including `hasChild`, `hasGrandparent`, `hasParent`, `olderThan`, and `youngerThan`. The `hasChild` property is selected, and its annotations are displayed: `rdfs:comment` with the text "Asserts that the subject has a child who is the object; it is the inverse of hasParent". The `Characteristics` section for `hasChild` shows the following settings:  Functional,  Inverse functional,  Transitive,  Symmetric,  Asymmetric,  Reflexive, and  Irreflexive. The `Description` section shows the property is `Equivalent To`, `SubProperty Of`, and `Inverse Of` `hasParent`. It also shows `Domains (intersection)` and `Ranges (intersection)` both set to `Person`. At the bottom right, the `Show inferences` checkbox is checked and circled in red.



# Reasoner fails

The screenshot shows the Protege IDE interface. A 'Log' window is open, displaying the following text:

```
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]
```

Below the log window are buttons for 'Show log file', 'Preferences', 'Time stamp', and 'Clear log'. An 'OK' button is at the bottom right of the log window.

The background shows the Protege main interface with a search bar, a toolbar, and a workspace area. At the bottom, there are buttons for 'Disjoint With +', 'SuperProperty Of (Chain) +', and a status bar with 'Git: master', 'To use the reasoner click Reasoner > Start reasoner', 'Show Inferences' (checked), and a warning icon.

# Reasoner fails

```
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

SuperProperty Of (Chain) +

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