Pellet reasoner using SWRL

BY SUSANA R. DE NOVOA

Pellet Overview

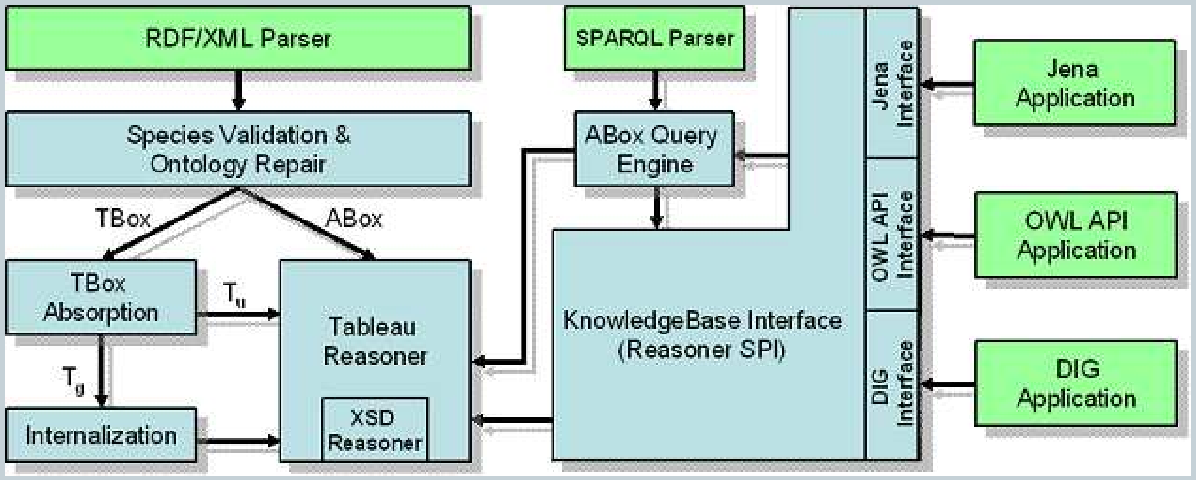
- Pellet is an open-source Java based OWL DL reasoner and it can be used with Jena and OWL API libraries.

- Features:

  - **Consistency checking:** ensures that an ontology does not contain any contradictory facts.
  - **Concept satisfiability:** checks if it is possible for a class to have any instances.
  - **Classification:** computes the subclass relations between every named class to create the complete class hierarchy.
  - **Realization:** computes the direct types for each of the individuals.

Architecture of Pellet

- Terms:
  - **Assertional Box** (Abox) contains assertions about individuals, i.e. OWL facts such as type, property-value…
  - **Terminological Box** (Tbox) contains axioms about classes, i.e. OWL axioms such as subclass, equivalent class…
  - **Knowledge Base** (KB) A combination of an ABox and a TBox, i.e. a complete OWL ontology.



Architecture of Pellet

- Main modules:
  - **Parsing and Loading:** Pellet include different submodules that can load ontologies from different representations (Jena, WonderWeb..).
  - **Tableaux Reasoner:** has the functionality of checking the consistency of an ontology by constructing a graph from the Abox. The reasoner repeatedly applies the tableaux expansion rules until a contradiction is detected.
  - **Datatype Reasoner:** is responsible for checking if the intersection of datatypes is consistent or not.
  - **Knowledge Base Interface:** makes decisions to check the consistency of the ABox, when to classify all the concepts or when to realize all the individuals and answer queries.
  - **ABox Query Engine:** The KnowledgeBase interface is coupled with an ABox Query Engine that answers queries.

Architecture of Pellet

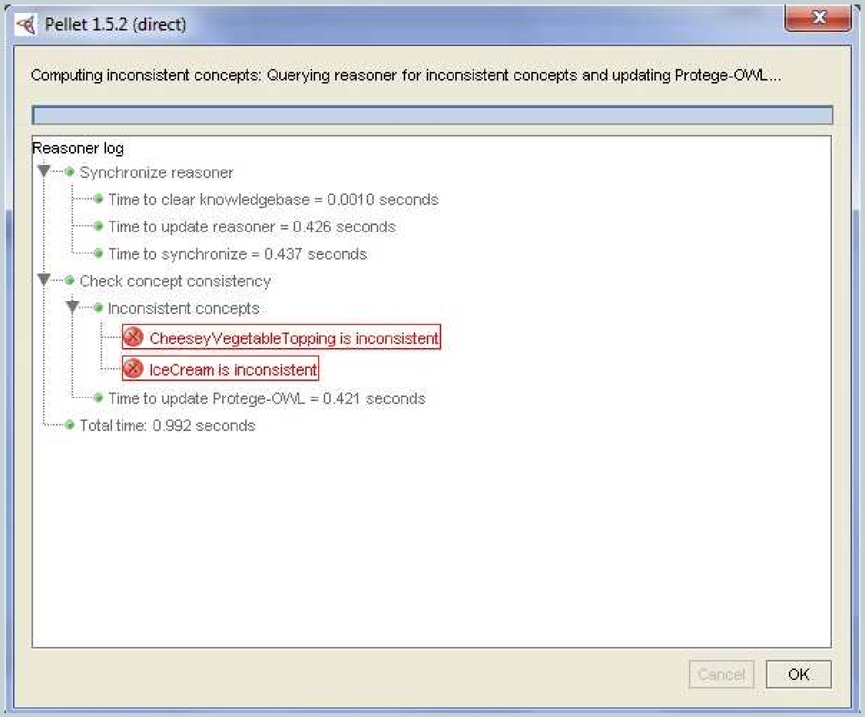
- Operation:
  - The core of the system is the tableaux reasoner that checks the consistency of a knowledge base.
  - The OWL ontologies are loaded into the reasoner after ontology validation and repair. This step ensures that all the resources have an appropriate type triple and missing type declarations are added.
  - During the loading phase, axioms about classes are put into the TBox component and assertions about individuals are stored in the ABox component.
  - The system provides a thin layer for programmatic access through the Service Programming Interface (SPI) that provides convenience functions to access the reasoning services provided.

Debugging Support

- Reasoners detect inconsistent ontologies but the diagnosis and resolution of the bug is not supported at all.
- Pellet contains two debugging services that help explain why the inconsistency occurs:
  - the first service clash detection is used to pinpoint the root contradiction or clash in the completion graph;
  - and the second – axiom tracing is used to extract the relevant source axioms from the ontology responsible for the clash.
- These services are used in the OWL Ontology Editor, Swoop, as a debugging aid for ontology users and modelers: http://code.google.com/p/swoop/

Pizza Example

- Reasoning -> Check Consistency
  - Protégé-OWL will have performed a consistency check of the pizza ontology:



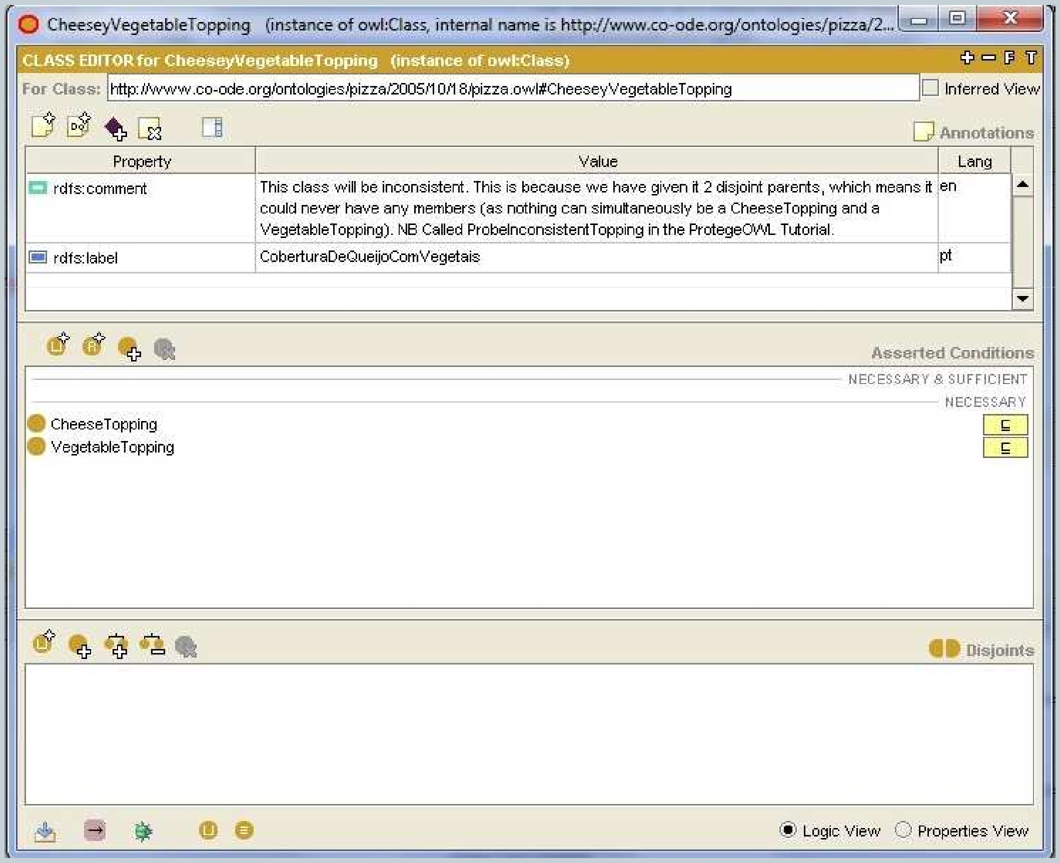
Pellet 1.5.2 (direct)

Computing inconsistent concepts: Querying reasoner for inconsistent concepts and updating Protege-OWL...

```
Reasoner log
▼ Synchronize reasoner
    Time to clear knowledgebase = 0.0010 seconds
    Time to update reasoner = 0.426 seconds
    Time to synchronize = 0.437 seconds
▼ Check concept consistency
  ▼ Inconsistent concepts
      CheeseyVegetableTopping is inconsistent
      IceCream is inconsistent
    Time to update Protege-OWL = 0.421 seconds
  Total time: 0.992 seconds
```

Pizza Example

- CheeseyVegetableTopping



CheeseyVegetableTopping (instance of owl:Class, internal name is http://www.co-ode.org/ontologies/pizza/2...

CLASS EDITOR for CheeseyVegetableTopping (instance of owl:Class)

For Class: http://www.co-ode.org/ontologies/pizza/2005/10/18/pizza.owl#CheeseyVegetableTopping

Inferred View

Annotations

Property	Value	Lang
rdfs:comment	This class will be inconsistent. This is because we have given it 2 disjoint parents, which means it could never have any members (as nothing can simultaneously be a CheeseTopping and a VegetableTopping). NB Called ProbeInconsistentTopping in the ProtegeOWL Tutorial.	en
rdfs:label	CoberturaDeQueijoComVegetais	pt

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

CheeseTopping ⊑

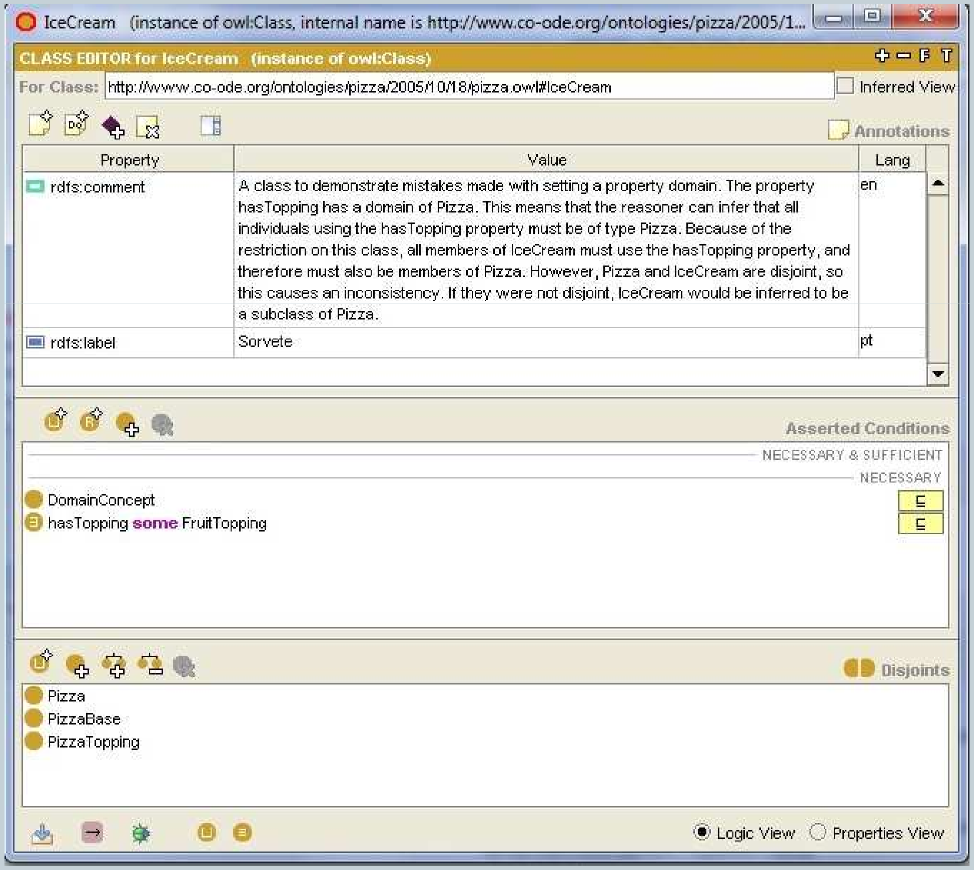
VegetableTopping ⊑

Disjoints

Logic View ● Properties View ○

Pizza Example

- Ice Cream



IceCream (instance of owl:Class, internal name is http://www.co-ode.org/ontologies/pizza/2005/1...

CLASS EDITOR for IceCream (instance of owl:Class)

For Class: http://www.co-ode.org/ontologies/pizza/2005/10/18/pizza.owl#IceCream ☐ Inferred View

Annotations

Property	Value	Lang
rdfs:comment	A class to demonstrate mistakes made with setting a property domain. The property hasTopping has a domain of Pizza. This means that the reasoner can infer that all individuals using the hasTopping property must be of type Pizza. Because of the restriction on this class, all members of IceCream must use the hasTopping property, and therefore must also be members of Pizza. However, Pizza and IceCream are disjoint, so this causes an inconsistency. If they were not disjoint, IceCream would be inferred to be a subclass of Pizza.	en
rdfs:label	Sorvete	pt

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

DomainConcept ⊑

hasTopping **some** FruitTopping ⊑

Disjoints

Pizza

PizzaBase

PizzaTopping

● Logic View ○ Properties View

THE END

THANK YOU!