Pellet An Owl DL Reasoner

Presented By
Aditya R Joshi
Neha Purohit

Pellet

- What is Pellet?
- Pellet is an OWL-DL reasoner
- Supports nearly all of OWL 1 and OWL 2
- Sound and complete reasoner
- Written in Java and available from http://clarkparsia.com/pellet
- Dual-licensed
- AGPL license for open-source applications
- Proprietary license available for proprietary applications

Pellet

Broader Definition

Pellet is an OWL DL reasoner based on the tableaux algorithms developed for expressive Description Logics. It supports the full expressivity OWL DL including reasoning about nominals (enumerated classes).

It can be used in conjunction with both Jena and OWL API libraries and also provides a DIG interface.

Pellet

- Pellet provides functionalities to see the species validation
- Check consistency of ontologies, classify the taxonomy
- Check entailments and answer a subset of RDQL queries (known as ABox queries in DL terminology).

Pellet

Why Pellet

- The capabilities of Pellet are exposed from a Java API, a command line interface, and a Web form.
- The Web form has been used by a number of people for species validation, consistency checking, and experimenting with OWLDL classification and entailment.
- Pellet provides programmatic access to the reasoning functions through two different interfaces, one for the Jena toolkit and one for the OWLAPI library.

Pellet

Why Pellet

- Pellet is also used to find the inconsistent concept descriptions in browsed ontologies. This feature helps to locate the errors in the ontology

Pellet Architecture

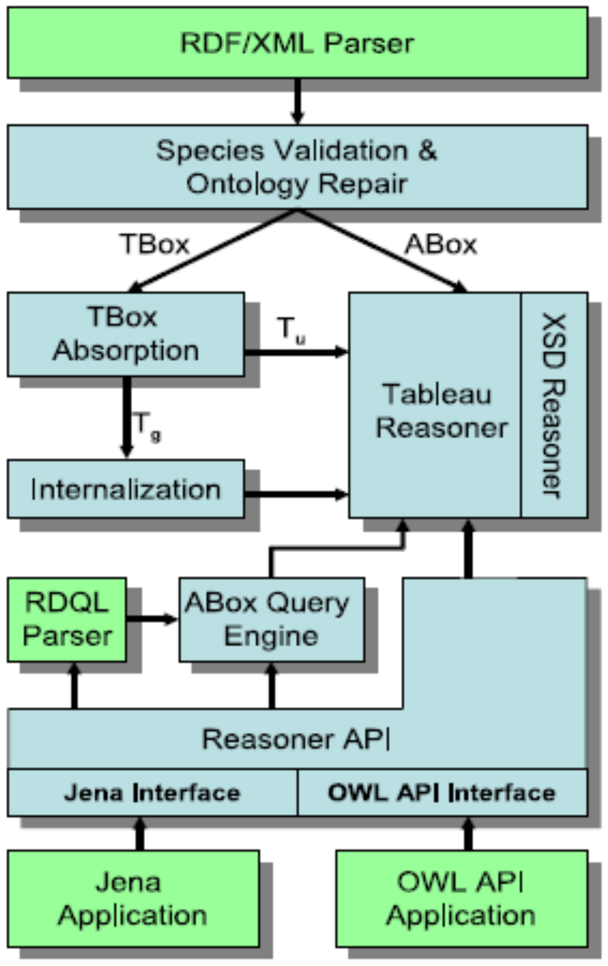


Figure 1: Architecture of Pellet reasoner

Pellet Architecture

- Figure 1 shows the main components of Pellet reasoner.
- An OWL ontology is parsed into RDF triples (RDF/ XML, N3 and N-Triple syntaxes are supported). Pellet validates the species of the ontology while the triples are converted to assertions and axioms in the knowledge base. If the ontology level is OWL Full because of missing type triples then Pellet uses some heuristics to repair the ontology. For example

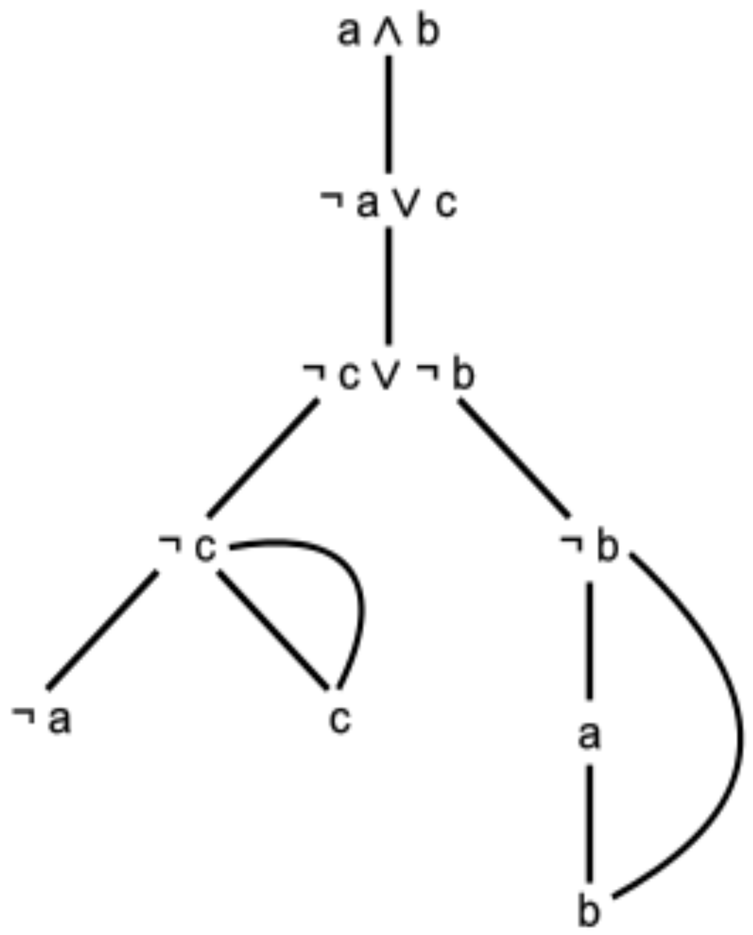
Pellet Architecture

- For example an untyped resource that has been used in the predicate position of a triple will be inferred to be a data type property if the triple has a literal in the object position.
- As usual, Pellet stores the axioms about classes in the TBox component and stores the assertions about individuals in the ABox component. TBox partitioning, absorption and lazy unfolding optimizations are implemented

Pellet Architecture

- The tableau reasoner uses the standard tableau rules (as described above) and includes various standard optimizations such as dependency directed backjumping, semantic branching and early blocking strategies.
- Datatype reasoning for the built-in and derived primitive XML Schema datatypes are supported.Pellet is implemented in pure Java

Small Intro on tableau Algorithm



Small Intro on tableau Algorithm

- In proof theory, the **semantic tableau** (or **truth tree**) is a decision procedure for sentential and related logics, and a proof procedure for formulas of first-order logic. The tableau method can also determine the satisfiability of finite sets of formulas of various logics. It is the most popular proof procedure for modal logics (Girle 2000). The method of semantic tableaux was invented by the Dutch logician Evert Willem Beth.

Owl Terminology

- Class: Person, Organization, Project, Skill, …
- Datatype: string, integer, date, …
- Individual: Evren, C&P, POPS, …
- Literal: "Evren Sirin", 5, 5/26/2008, …
- Object Property: worksAt, hasSkill, …
- Data property: name, proficiencyLevel, …

Owl Terminology

- Class expressions

  and, or, not

  some, only, min, max, exactly, value, Self

  { … }

- Datatype definitions
- and, or, not
- <, <=, >, >=
- { … }

Owl Terminology

- Class axioms
  subClassOf, equivalentTo, disjointWith
- Property axioms
  subPropertyOf, equivalentTo, inverseOf, disjointWith, subPropertyChain, domain, range
- Property characteristics
  Functional, InverseFunctional, Transitive, Symmetric, Asymmetric, Reflexive, Irreflexive
- Individual assertions
  Class assertion, property assertion, sameAs, differentFrom

OWL Example

- Employee **equivalentTo (CivilServant or Contractor)**
- CivilServant **disjointWith Contractor**
- Employee **subClassOf**
  employeeID **some integer**[**>= 100000, <= 999999**]
- Employee **subClassOf employeeID exactly 1**
- worksOnProject **domain Person**
- worksOnProject **range Project**
- Person0853 **type CivilServant**
- Person0853 employeeID 312987
- Person0853 worksOnProject Project2133

OWL Example

- Employee **equivalentTo (CivilServant or Contractor)**
- CivilServant **disjointWith Contractor**
- Employee **subClassOf**
  employeeID **some integer**[**>= 100000, <= 999999**]
- Employee **subClassOf employeeID exactly 1**
- worksOnProject **domain Person**
- worksOnProject **range Project**
- Person0853 **type CivilServant**
- Person0853 employeeID 312987
- Person0853 worksOnProject Project2133

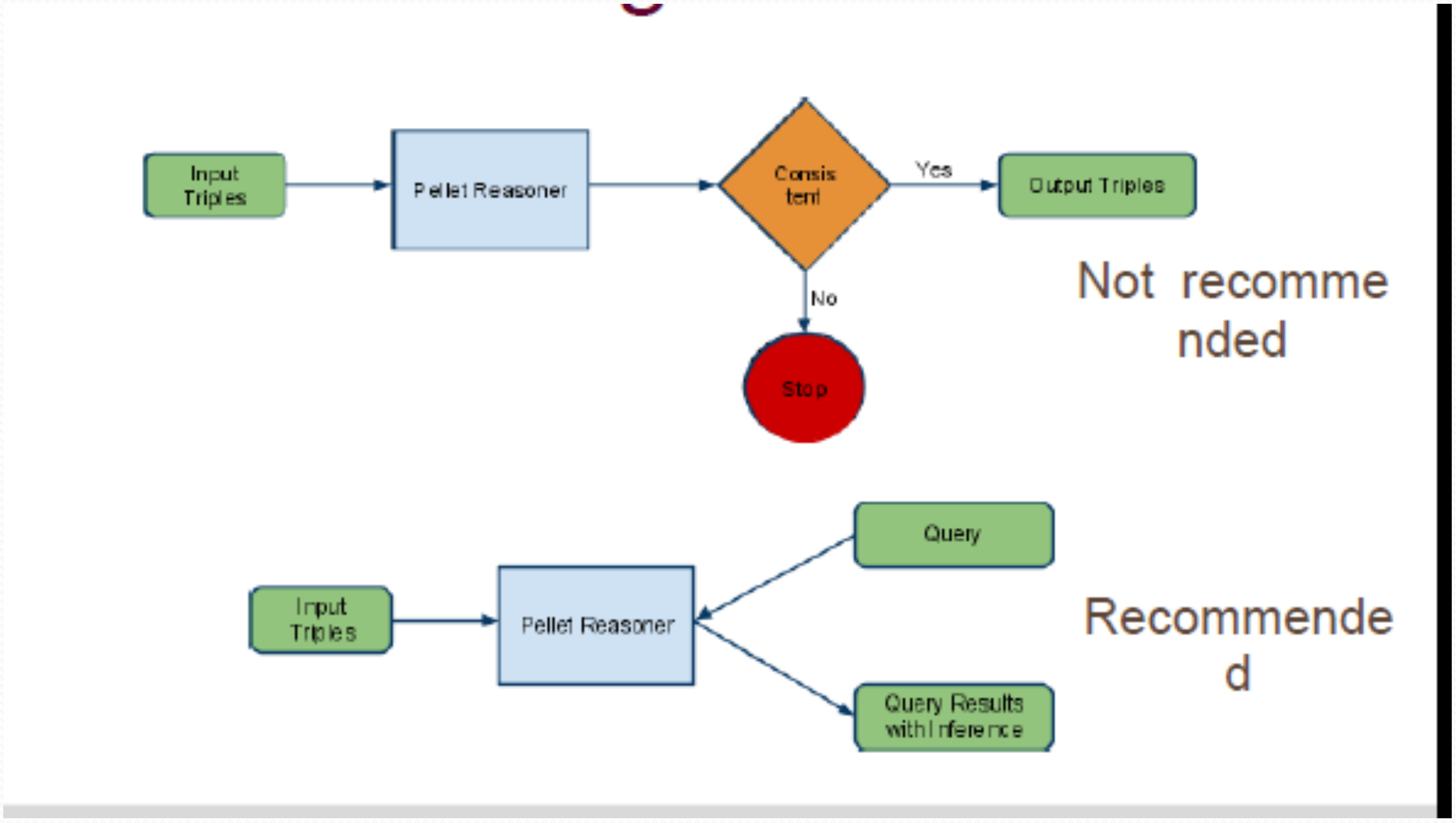
Reasoning in OWL

- Check the consistency of a set of axioms
- Verify the input axioms do not contain contradictions

Inconsistency Examples

- Unsatisfiable class cannot have any instances Consistent ontologies may contain unsatisfiable classes Declaring an instance for an unsatisfiable class causes inconsistency
- Example
- CivilServant **disjointWith Contractor**
- CivilServantContractor **subClassOf** (CivilServant **and Contractor**)

Using Pellet



Programming with Pellet

API for Pellet

- Pellet can be used via three different APIs
- Internal Pellet API
- Manchester OWLAPI
- Jena API

Each API has pros and cons

- Choice will depend on your applications' needs and requirements

Pellet Internal API

API used by the reasoner

- Designed for efficiency, not usability
- Uses ATerm library for representing terms
- Fine-grained control over reasoning
- Misses features (e.g. parsing & serialization)
- Pros: Efficiency, fine-grained control
- Cons: Low usability, missing features

Manchester OWLAPI

- API designed for OWL
- Closely tied to OWL structural specification
- Support for many syntaxes (RDF/XML, OWL/XML, OWL functional, Turtle, …)
- Native SWRL support
- Integration with reasoners
- Support for modularity and explanations
- Pros: OWL-centric API
- Cons: Not as stable, no SPARQL support
- More info: http://owlapi.sf.net

Jena API

- RDF framework developed by HP labs
- An RDF API with OWL extensions
- In-memory and persistent storage
- Built-in rule reasoners and integrated with Pellet
- SPARQL query engine
- Pros: Mature and stable and ubiquitous
- Cons: Not great for handling OWL, no specific OWL 2 support
- More info: http://jena.sf.net

Jena API

- RDF framework developed by HP labs
- An RDF API with OWL extensions
- In-memory and persistent storage
- Built-in rule reasoners and integrated with Pellet
- SPARQL query engine
- Pros: Mature and stable and ubiquitous
- Cons: Not great for handling OWL, no specific
- OWL 2 support
- More info: http://jena.sf.net

Owl2

- Owl2 is similar to owl1 with added capabilities
- keys;
- property chains;
- richer datatypes, data ranges;
- qualified cardinality restrictions;
- asymmetric, reflexive, and disjoint properties; and
- enhanced annotation capabilities

Dealing with Inconsistency

- In semantic web inconsistency are unavoidable
  Distributed data, no single point of enforcement
- Expressive modeling language
- Classical logical formalisms are not good at dealing with inconsistency
- Reasoners refuse to reason with inconsistent ontologies
- Paraconsistent logics not practical

Dealing with Inconsistency

- In semantic web inconsistency are unavoidable
  Distributed data, no single point of enforcement
- Expressive modeling language
- Classical logical formalisms are not good at dealing with inconsistency
- Reasoners refuse to reason with inconsistent ontologies
- Paraconsistent logics not practical

- Typical process for solving a contradiction

- Use Pellet to find which axioms cause contradiction
  Domain expert (human) inspects the axiom set

- Expert edits/deleted incorrect axioms

An automated (and cautious) solution
- Use Pellet to find which axioms cause contradiction
- Delete all reported axioms
- When to use the automated solution
- Pros: Completely automated, guaranteed to retain
- only consistent information
- Cons: May remove too much information

Code Snippet

```
// continue until all inconsistencies are resolved
while (!pellet.isConsistent()) {
// get the explanation for current inconsistency
Graph explanation = pellet.explainInconsistency();
// iterate over the axioms in the explanation
for (Triple triple : explanation.find(Triple.ANY).toList() ) {
// remove any individual assertion that contributes
// to the inconsistency (assumption: all the axioms
// in the schema are believed to be correct and
// should not be removed)
if (isIndividualAssertion(triple))
graph.remove(triple);
}
}
```

Code Snippet

```
// continue until all inconsistencies are resolved
while (!pellet.isConsistent()) {
// get the explanation for current inconsistency
Graph explanation = pellet.explainInconsistency();
// iterate over the axioms in the explanation
for (Triple triple : explanation.find(Triple.ANY).toList() ) {
// remove any individual assertion that contributes
// to the inconsistency (assumption: all the axioms
// in the schema are believed to be correct and
// should not be removed)
if (isIndividualAssertion(triple))
graph.remove(triple);
}
}
```

Closed vs. Open World

- Two different views on truth
- CWA: Any statement that is not known to be true is false
- OWA: A statement is false only if it is known to be false
- Used in different contexts
- Databases use CWA because (typically) you have *complete information*
- Ontologies use OWA because (typically) you have *incomplete information*
- Data validation results significantly

Closed vs. Open World

- CWA or OWA Validation?
- Should I use CWA or OWA?
- Of course use both!
- In the application domain there is complete information about some parts but not others
- In POPS application we have…Complete knowledge about employees
- Incomplete information about external publications
- Retrieved from conference proceedings, etc

Pops interface

- http://nasa.clarkparsia.com/
- http://www.mindswap.org/2003/pellet/demo.shtml

Pops interface

- Code demo by neha

Thanks You