Semantic Web: Lab 3

ZHANG Xin

Table of Contents

Excise 1 (OWL API)

Load an existing ontology into Java by OWL API

by using `OWLManager`:

```
OWLOntologyManager manager=OWLManager.createOWLOntologyManager();
OWLDataFactory dataFactory=manager.getOWLDataFactory();
File inputOntologyFile = new File("$FILEPATH$");
OWLOntology ontology=manager.loadOntologyFromOntologyDocument(inputOntologyFile);
```

Get the numbers of concepts, relations (roles), instances and logical axioms

```
// Number of logical axioms
System.out.println(ontology.getLogicalAxiomCount());
//Number of concepts (classes)
System.out.println(ontology.getClassesInSignature().size());
// Number of instances (individuals)
System.out.println(ontology.getIndividualsInSignature().size());
// Number of relations (roles)
System.out.println(ontology.getObjectPropertiesInSignature().size());
```

for the file `pizza.owl`, the output is

```
712
100
5
8
```

Save an ontology to a file in a particular syntax

By using the `saveOntology()` method of the class `OWLOntologyManager` with the parameter `ontologyFormat` as the example shown below:

```
File newOntologyFile=new File("examples/ontologies/pizza.fss.owl");
//create a buffered stream since the ontology manager can then write to that
stream.
newOntologyFile=newOntologyFile.getAbsoluteFile();
BufferedOutputStream outputStream=new BufferedOutputStream(new
FileOutputStream(newOntologyFile));
//save in a particular syntax
manager.saveOntology(ontology, new OWLFunctionalSyntaxOntologyFormat(),
outputStream);
```

Excise 2 (Ontology Creation)

So the complete code and outputed file are shown below:

```
package org.semanticweb.HermiT.examples;

import org.semanticweb.owlapi.apibinding.*;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.util.*;
import java.io.*;
import java.util.*;

public class OntoCreat {
    public static void main(String[] args) {
        IRI ontologyIRI = IRI.create("http://university.study/");
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();

        try {
            // Create an empty ontology
            OWLOntology ontology = manager.createOntology(ontologyIRI);

            //Create atomic concepts
            OWLDataFactory factory = manager.getOWLDataFactory();
            PrefixManager pm = new DefaultPrefixManager(ontologyIRI.toString());
            OWLClass student = factory.getOWLClass(":Student",pm);
            OWLDeclarationAxiom das = factory.getOWLDeclarationAxiom(student);
            manager.addAxiom(ontology, das);
            OWLClass researcher = factory.getOWLClass(":Researcher",pm);
            OWLDeclarationAxiom dar = factory.getOWLDeclarationAxiom(researcher);
            manager.addAxiom(ontology, dar);

            OWLClass enterprise = factory.getOWLClass(":Enterprise", pm);
            OWLDeclarationAxiom dae = factory.getOWLDeclarationAxiom(enterprise);
            manager.addAxiom(ontology, dae);

            //Create individuals / ABox
            OWLNamedIndividual pierre = factory.getOWLNamedIndividual(":Pierre",
pm);
            OWLClassAssertionAxiom cap =
factory.getOWLClassAssertionAxiom(student, pierre);
            manager.addAxiom(ontology,cap);

            OWLNamedIndividual john = factory.getOWLNamedIndividual(":John", pm);
            OWLClassAssertionAxiom caj =
factory.getOWLClassAssertionAxiom(student, john);
            manager.addAxiom(ontology,caj);
            OWLClassAssertionAxiom cajr =
factory.getOWLClassAssertionAxiom(researcher, john);
            manager.addAxiom(ontology,cajr);

            OWLNamedIndividual ubisoft = factory.getOWLNamedIndividual(":Ubisoft",
pm);
            OWLClassAssertionAxiom cau =
factory.getOWLClassAssertionAxiom(enterprise, ubisoft);
            manager.addAxiom(ontology, cau);

            OWLNamedIndividual thales= factory.getOWLNamedIndividual(":Thales",
pm);
            OWLClassAssertionAxiom cath =
factory.getOWLClassAssertionAxiom(enterprise, thales);
            manager.addAxiom(ontology, cath);

            //Create roles
            OWLDataProperty hasStudentId =
factory.getOWLDataProperty(":hasStudentId", pm );

            OWLObjectProperty hasDiploma =
factory.getOWLObjectProperty(":hasDiploma", pm);
            OWLIndividual engineering =
factory.getOWLNamedIndividual(":Engineering", pm);
            OWLIndividual master = factory.getOWLNamedIndividual(":Master", pm);
            Set<OWLAxiom> diplomas = new HashSet<OWLAxiom>();
            diplomas.add(factory.getOWLObjectPropertyAssertionAxiom(hasDiploma,
pierre ,engineering));
            diplomas.add(factory.getOWLObjectPropertyAssertionAxiom(hasDiploma,
john, master));
            manager.addAxioms(ontology, diplomas);

            OWLObjectProperty workIn = factory.getOWLObjectProperty(":WorkIn",
pm);
```

```
            //ABox
            OWLObjectPropertyAssertionAxiom aa0 =
factory.getOWLObjectPropertyAssertionAxiom(workIn, john, thales);
            manager.addAxiom(ontology, aa0);
            OWLDataPropertyAssertionAxiom aa1 =
factory.getOWLDataPropertyAssertionAxiom(hasStudentId, john, 12345);
            manager.addAxiom(ontology, aa1);
            OWLDataPropertyAssertionAxiom aa2 =
factory.getOWLDataPropertyAssertionAxiom(hasStudentId, pierre, 54321);
            manager.addAxiom(ontology, aa2);


            //Create complex concepts / TBox
            OWLClass phd = factory.getOWLClass(":PhDStudent", pm);
            OWLClassExpression phdStudent =
factory.getOWLObjectIntersectionOf(student, researcher);
            OWLAxiom phdDef = factory.getOWLEquivalentClassesAxiom(phd,
phdStudent);
            manager.addAxiom(ontology, phdDef);

            OWLObjectHasValue hasEngineerDegreeRestriction =
factory.getOWLObjectHasValue(hasDiploma, engineering);
            OWLClass eng = factory.getOWLClass(":Engineer", pm);
            OWLClassExpression engStudent =
factory.getOWLObjectIntersectionOf(student, hasEngineerDegreeRestriction);
            OWLAxiom engDef = factory.getOWLEquivalentClassesAxiom(eng,
engStudent);
            manager.addAxiom(ontology, engDef);

            //make workIn only can be followed by enterprises and make the
property Functional, InverseFunctional, Irreflexive and Asymmetric
            Set<OWLAxiom> domainsAndRanges = new HashSet<OWLAxiom>();
            domainsAndRanges.add(factory.getOWLObjectPropertyDomainAxiom(workIn,
enterprise));
            domainsAndRanges.add(factory.getOWLObjectPropertyRangeAxiom(workIn,
enterprise));
            manager.addAxioms(ontology, domainsAndRanges);
            Set<OWLAxiom> workInAxioms = new HashSet<OWLAxiom>();
            workInAxioms.add(factory.getOWLFunctionalObjectPropertyAxiom(workIn));
            workInAxioms.add(factory
                    .getOWLInverseFunctionalObjectPropertyAxiom(workIn));
            workInAxioms
                    .add(factory.getOWLIrreflexiveObjectPropertyAxiom(workIn));
            workInAxioms.add(factory.getOWLAsymmetricObjectPropertyAxiom(workIn));
            // Add all of the axioms that specify the characteristics of workIn
            manager.addAxioms(ontology, workInAxioms);

            OWLClass apprentice = factory.getOWLClass(":Apprentice", pm);
            OWLObjectAllValuesFrom workInRestriction =
factory.getOWLObjectAllValuesFrom(workIn, enterprise);
            OWLClassExpression appStu =
factory.getOWLObjectIntersectionOf(student, workInRestriction);
            OWLAxiom appDef = factory.getOWLEquivalentClassesAxiom(apprentice,
appStu);
```

```
            manager.addAxiom(ontology, appDef);

            File file =new File("C:\\Users\\Hsin
Chang\\Desktop\\HermiT\\project\\examples\\ontologies\\university.owl");
            if (!file.exists())
                file.createNewFile();
            file = file.getAbsoluteFile();
            BufferedOutputStream fileStreamOut = new BufferedOutputStream(new
FileOutputStream(file));
            manager.saveOntology(ontology, fileStreamOut);
            } catch (OWLOntologyCreationException | IOException |
OWLOntologyStorageException e) {
            e.printStackTrace();
             }
         }
}
```

And the `owl` file generated:

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://university.study/"
     xml:base="http://university.study/"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:university="http://university.study/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://university.study/"/>



    <!--

///////////////////////////////////////////////////////////////////////////////////
/////
    //
    // Object Properties
    //

///////////////////////////////////////////////////////////////////////////////////
/////
     -->




    <!-- http://university.study/WorkIn -->

    <owl:ObjectProperty rdf:about="http://university.study/WorkIn">
        <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#AsymmetricProperty"/>
        <rdf:type
```

```
rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdf:type 
rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
        <rdf:type 
rdf:resource="http://www.w3.org/2002/07/owl#IrreflexiveProperty"/>
        <rdfs:range rdf:resource="http://university.study/Enterprise"/>
        <rdfs:domain rdf:resource="http://university.study/Enterprise"/>
    </owl:ObjectProperty>



    <!-- http://university.study/hasDiploma -->

    <owl:ObjectProperty rdf:about="http://university.study/hasDiploma"/>



    <!--
    
///////////////////////////////////////////////////////////////////////////////////
/////
    //
    // Data properties
    //
    
///////////////////////////////////////////////////////////////////////////////////
/////
     -->




    <!-- http://university.study/hasStudentId -->

    <owl:DatatypeProperty rdf:about="http://university.study/hasStudentId"/>



    <!--
    
///////////////////////////////////////////////////////////////////////////////////
/////
    //
    // Classes
    //
    
///////////////////////////////////////////////////////////////////////////////////
/////
     -->




    <!-- http://university.study/Apprentice -->
    <owl:Class rdf:about="http://university.study/Apprentice">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://university.study/Student"/>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://university.study/WorkIn"/>
                        <owl:allValuesFrom
rdf:resource="http://university.study/Enterprise"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://university.study/Engineer -->

    <owl:Class rdf:about="http://university.study/Engineer">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="http://university.study/Student"/>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://university.study/hasDiploma"/>
                        <owl:hasValue
rdf:resource="http://university.study/Engineering"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://university.study/Enterprise -->

    <owl:Class rdf:about="http://university.study/Enterprise"/>



    <!-- http://university.study/PhDStudent -->

    <owl:Class rdf:about="http://university.study/PhDStudent">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description
rdf:about="http://university.study/Researcher"/>
                    <rdf:Description rdf:about="http://university.study/Student"/>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://university.study/Researcher -->

    <owl:Class rdf:about="http://university.study/Researcher"/>



    <!-- http://university.study/Student -->

    <owl:Class rdf:about="http://university.study/Student"/>



    <!--
///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Individuals
    //
///////////////////////////////////////////////////////////////////////////////////
////
     -->




    <!-- http://university.study/Engineering -->

    <owl:NamedIndividual rdf:about="http://university.study/Engineering"/>



    <!-- http://university.study/John -->

    <owl:NamedIndividual rdf:about="http://university.study/John">
        <rdf:type rdf:resource="http://university.study/Researcher"/>
        <rdf:type rdf:resource="http://university.study/Student"/>
        <hasStudentId 
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">12345</hasStudentId>
        <hasDiploma rdf:resource="http://university.study/Master"/>
        <WorkIn rdf:resource="http://university.study/Thales"/>
    </owl:NamedIndividual>



    <!-- http://university.study/Master -->
    <owl:NamedIndividual rdf:about="http://university.study/Master"/>
    


    <!-- http://university.study/Pierre -->

    <owl:NamedIndividual rdf:about="http://university.study/Pierre">
        <rdf:type rdf:resource="http://university.study/Student"/>
        <hasStudentId 
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">54321</hasStudentId>
        <hasDiploma rdf:resource="http://university.study/Engineering"/>
    </owl:NamedIndividual>
    


    <!-- http://university.study/Thales -->

    <owl:NamedIndividual rdf:about="http://university.study/Thales">
        <rdf:type rdf:resource="http://university.study/Enterprise"/>
    </owl:NamedIndividual>
    


    <!-- http://university.study/Ubisoft -->

    <owl:NamedIndividual rdf:about="http://university.study/Ubisoft">
        <rdf:type rdf:resource="http://university.study/Enterprise"/>
    </owl:NamedIndividual>
</rdf:RDF>



<!-- Generated by the OWL API (version 3.4.3) http://owlapi.sourceforge.net -->
```

Excise 3. (Ontology Reasoning by HermiT)

Consistency and simple reasoning tasks

```
package org.semanticweb.HermiT.examples;

import org.semanticweb.HermiT.*;
import org.semanticweb.owlapi.apibinding.*;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.*;

import java.io.File;

public class OntoCheck {
    public static void main(String[] args) throws Exception {
        // First, we create an OWLOntologyManager object. The manager will load
and
        // save ontologies.
        OWLOntologyManager manager= OWLManager.createOWLOntologyManager();
        // I will create several things, so we save an instance of the data
factory
        OWLDataFactory dataFactory=manager.getOWLDataFactory();
        // Now, we create the file from which the ontology will be loaded.
        // Here the ontology is stored in a file locally in the ontologies
subfolder
        // of the examples folder.
        File inputOntologyFile = new File("C:\\Users\\Hsin
Chang\\Desktop\\HermiT\\project\\examples\\ontologies\\university.owl");
        // Use the OWL API to load the ontology.
        OWLOntology
ontology=manager.loadOntologyFromOntologyDocument(inputOntologyFile);

        Reasoner.ReasonerFactory factory = new Reasoner.ReasonerFactory();
        // I don't want HermiT to thrown an exception for inconsistent ontologies
because then we
        // can't explain the inconsistency. This can be controlled via a
configuration setting.
        Configuration configuration=new Configuration();
        configuration.throwInconsistentOntologyException=false;
        // The factory can now be used to obtain an instance of HermiT as an
OWLReasoner.
        OWLReasoner reasoner=factory.createReasoner(ontology, configuration);
        reasoner.precomputeInferences(InferenceType.CLASS_HIERARCHY);

        //Check consistency
        System.out.println("Is the ontology consistent?
"+reasoner.isConsistent());

        //Some reasoning tasks
        OWLClassExpression student =
manager.getOWLDataFactory().getOWLClass(IRI.create("http://university.study/Studen
t"));
        OWLClassExpression phd =
manager.getOWLDataFactory().getOWLClass(IRI.create("http://university.study/PhDStu
dent"));
        OWLAxiom axiom = manager.getOWLDataFactory().getOWLSubClassOfAxiom(phd,
student);
        System.out.println("Is PhDStudetnt subsumed by student?
"+reasoner.isEntailed(axiom));

        OWLObjectProperty hasdiploma =
manager.getOWLDataFactory().getOWLObjectProperty(IRI.create("http://university.stu
dy/hasDiploma"));
        OWLIndividual eng =
manager.getOWLDataFactory().getOWLNamedIndividual(IRI.create("http://university.st
udy/Engineering"));
        OWLObjectHasValue hasEngineerDegreeRestriction =
manager.getOWLDataFactory().getOWLObjectHasValue(hasdiploma, eng);
```

```
        OWLClassExpression engStu =
manager.getOWLDataFactory().getOWLClass(IRI.create("http://university.study/Engine
er"));
        OWLAxiom axiom1 =
manager.getOWLDataFactory().getOWLSubClassOfAxiom(engStu,
hasEngineerDegreeRestriction);
        System.out.println("Are Engineers subsumed by all those who have obtained
Enginnering diploma? "+reasoner.isEntailed(axiom1));


    }
}
```

Output:

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe" ...
Is the ontology consistent? true
Is PhDStudetnt subsumed by student? true
Are Engineers subsumed by all those who have obtained Enginnering diploma? true

Process finished with exit code 0
```

Classification

Classification means computing the subsumption relations between all concept names in TBoxs.

```
//Continued from line 50 of the former file in this Excise
        //Classification
        System.out.println("Number of axioms before classification: " +
ontology.getLogicalAxiomCount());
        List<InferredAxiomGenerator<? extends OWLAxiom>> generators=new
ArrayList<InferredAxiomGenerator<? extends OWLAxiom>>();
        generators.add(new InferredSubClassAxiomGenerator());
        generators.add(new InferredClassAssertionAxiomGenerator());
        InferredOntologyGenerator iog=new
InferredOntologyGenerator(reasoner,generators);
        OWLOntology inferredAxiomsOntology=manager.createOntology();
        iog.fillOntology(manager, inferredAxiomsOntology);
        System.out.println("New axioms from classification: " +
inferredAxiomsOntology.getLogicalAxiomCount());

        File inferredOntologyFile=new File("C:\\Users\\Hsin
Chang\\Desktop\\HermiT\\project\\examples\\ontologies\\university-inferred.owl");
        if (!inferredOntologyFile.exists())
            inferredOntologyFile.createNewFile();
        inferredOntologyFile=inferredOntologyFile.getAbsoluteFile();
        OutputStream outputStream=new FileOutputStream(inferredOntologyFile);
        manager.saveOntology(inferredAxiomsOntology,
manager.getOntologyFormat(ontology), outputStream);
```

result:

```
Number of axioms before classification: 19
New axioms from classification: 25
```

Excise 4. (Testing Additional Functionality of OWL API)

Profile Validation

Tested on the ontology created in exercise 2.

```
package org.semanticweb.HermiT.examples;

import org.semanticweb.owlapi.apibinding.*;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.profiles.OWL2DLProfile;
import org.semanticweb.owlapi.profiles.OWLProfileReport;
import org.semanticweb.owlapi.profiles.OWLProfileViolation;

import java.io.*;

public class owlProfileValidation {
    public static void main(String[] args) throws Exception {
        OWLOntologyManager manager= OWLManager.createOWLOntologyManager();
        // I will create several things, so we save an instance of the data
factory
        OWLDataFactory dataFactory=manager.getOWLDataFactory();
        // Now, we create the file from which the ontology will be loaded.
        // Here the ontology is stored in a file locally in the ontologies
subfolder
        // of the examples folder.
        File inputOntologyFile = new File("C:\\Users\\Hsin
Chang\\Desktop\\HermiT\\project\\examples\\ontologies\\university.owl");
        // Use the OWL API to load the ontology.
        OWLOntology
ontology=manager.loadOntologyFromOntologyDocument(inputOntologyFile);

        //Profile validation
        OWL2DLProfile profile = new OWL2DLProfile();
        OWLProfileReport report = profile.checkOntology(ontology);
        //If there is no violations, the ontology is valid without profile
violations
        if(report.getViolations().isEmpty()){
            System.out.println("The ontology is valid without profile
violations.");
        }
        else {
            for (OWLProfileViolation v : report.getViolations()) {
                System.out.println(v);
            }
        }
    }
}
```

Output:

```
The profile is valid without violations.

Process finished with exit code 0
```

Explanation

The test is run with a newly created class `Dummy`, we will make it a subclass of `Phd Student`, but disjoint with `Student`, therefore it will be unsatisfiable.

```
package org.semanticweb.HermiT.examples;

import java.io.File;
import java.util.Set;

import org.semanticweb.HermiT.Configuration;
import org.semanticweb.HermiT.Reasoner.ReasonerFactory;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.OWLReasoner;

import com.clarkparsia.owlapi.explanation.BlackBoxExplanation;
import com.clarkparsia.owlapi.explanation.HSTExplanationGenerator;
import org.semanticweb.owlapi.util.DefaultPrefixManager;

public class Explanations {

    public static void main(String[] args) throws Exception {
        OWLOntologyManager manager=OWLManager.createOWLOntologyManager();
        OWLDataFactory dataFactory=manager.getOWLDataFactory();

        File inputOntologyFile = new File("C:\\Users\\Hsin
Chang\\Desktop\\HermiT\\project\\examples\\ontologies\\university.owl");
        OWLOntology
ontology=manager.loadOntologyFromOntologyDocument(inputOntologyFile);

        // We will firstly create an unsatisfiable class "Dummy" in the ontology
        OWLDataFactory dFactory = manager.getOWLDataFactory();
        IRI ontologyIRI = IRI.create("http://university.study/");
        PrefixManager pm = new DefaultPrefixManager(ontologyIRI.toString());
        OWLClassExpression student =
manager.getOWLDataFactory().getOWLClass(IRI.create("http://university.study/Studen
t"));
        OWLClassExpression phd =
manager.getOWLDataFactory().getOWLClass(IRI.create("http://university.study/PhDStu
dent"));
        OWLClass dummy = dFactory.getOWLClass(":Dummy",pm);
        OWLDeclarationAxiom das = dFactory.getOWLDeclarationAxiom(dummy);
        manager.addAxiom(ontology, das);
        OWLAxiom dummyExp = dFactory.getOWLSubClassOfAxiom(dummy,phd);
        manager.addAxiom(ontology, dummyExp);
        OWLAxiom dummyX = dFactory.getOWLDisjointClassesAxiom(dummy, student);
        manager.addAxiom(ontology, dummyX);

        //Then we test and run the explanation
        IRI classIRI=IRI.create("http://university.study/Dummy");
        OWLClass class_test =dataFactory.getOWLClass(classIRI);
        ReasonerFactory factory = new ReasonerFactory();
        Configuration configuration=new Configuration();
        configuration.throwInconsistentOntologyException=false;
        OWLReasoner reasoner=factory.createReasoner(ontology, configuration);
        System.out.println("Is Dummy satisfiable?
"+reasoner.isSatisfiable(class_test));
        System.out.println("Computing explanations...");
        BlackBoxExplanation exp=new BlackBoxExplanation(ontology, factory,
reasoner);
        HSTExplanationGenerator multExplanator=new HSTExplanationGenerator(exp);
        Set<Set<OWLAxiom>>
explanations=multExplanator.getExplanations(class_test);
        for (Set<OWLAxiom> explanation : explanations) {
            System.out.println("------------------");
            System.out.println("Axioms causing the unsatisfiability: ");
            for (OWLAxiom causingAxiom : explanation) {
                System.out.println(causingAxiom);
            }
            System.out.println("------------------");
        }
    }
}
```

Output

```
Is Dummy satisfiable? false
Computing explanations...
------------------
Axioms causing the unsatisfiability:
SubClassOf(<http://university.study/Dummy> <http://university.study/PhDStudent>)
EquivalentClasses(<http://university.study/PhDStudent>
ObjectIntersectionOf(<http://university.study/Researcher>
<http://university.study/Student>) )
DisjointClasses(<http://university.study/Dummy> <http://university.study/Student>)
```

```
------------------

Process finished with exit code 0
```

Modularity

```
package org.semanticweb.HermiT.examples;

import org.semanticweb.HermiT.*;
import org.semanticweb.owlapi.apibinding.*;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.*;
import org.semanticweb.owlapi.util.*;
import uk.ac.manchester.cs.owlapi.modularity.*;

import java.io.File;
import java.util.*;

public class owlModula {
    public static void main(String[] args) throws Exception {
        IRI ontologyIRI = IRI.create("http://university.study/");
        OWLOntologyManager manager=OWLManager.createOWLOntologyManager();
        OWLDataFactory dataFactory=manager.getOWLDataFactory();
        File inputOntologyFile = new File("C:\\Users\\Hsin
Chang\\Desktop\\HermiT\\project\\examples\\ontologies\\university.owl");
        OWLOntology
ontology=manager.loadOntologyFromOntologyDocument(inputOntologyFile);

    // extract a module for Student
    // start by creating a signature "Student"
        OWLClass student = dataFactory.getOWLClass(IRI.create(ontology +
"Student"));
// We now add all subclasses of the chosen classes.
        Set<OWLEntity> seedSig = new HashSet<OWLEntity>();

        Reasoner.ReasonerFactory factory = new Reasoner.ReasonerFactory();
        Configuration configuration=new Configuration();
        configuration.throwInconsistentOntologyException=false;
        OWLReasoner reasoner=factory.createReasoner(ontology, configuration);

        Set<OWLEntity> sig = new HashSet<OWLEntity>();
        sig.add(student);
        for (OWLEntity ent : sig) {
            seedSig.add(ent);
            if (ent.isOWLClass()) {
                NodeSet<OWLClass> subClasses =
reasoner.getSubClasses(ent.asOWLClass(),false);
                seedSig.addAll(subClasses.getFlattened());
            }
        }
        // Extract a locality-based module
        SyntacticLocalityModuleExtractor sme = new
SyntacticLocalityModuleExtractor(manager, ontology, ModuleType.STAR);
        Set<OWLAxiom> mod = sme.extract(seedSig);
        System.out.println("The module of Student is");
        System.out.println(mod);
        System.out.println("Module size "+ mod.size());

    }
}
```

Output

```
The module of Student is
[IrreflexiveObjectProperty(<http://university.study/WorkIn>),
Declaration(ObjectProperty(<http://university.study/WorkIn>)),
ObjectPropertyAssertion(<http://university.study/WorkIn>
<http://university.study/John> <http://university.study/Thales>),
Declaration(NamedIndividual(<http://university.study/Thales>)),
AsymmetricObjectProperty(<http://university.study/WorkIn>),
InverseFunctionalObjectProperty(<http://university.study/WorkIn>),
Declaration(NamedIndividual(<http://university.study/John>)),
FunctionalObjectProperty(<http://university.study/WorkIn>)]
Module size 8

Process finished with exit code 0
```