OWLConflictSolver

Generated by Doxygen 1.8.12

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

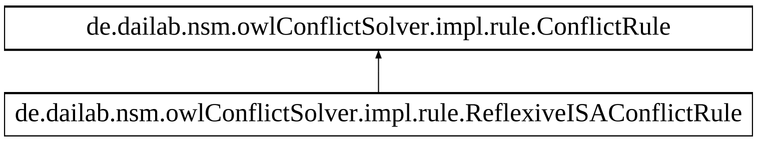
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule Interface Reference

Inheritance diagram for de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule:


de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule
de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule

Public Member Functions

- boolean **isApplicable** (OWLOntologyChange change)
- List< OWLOntologyChange > **resolveConflicts** (PelletReasoner reasoner)
- List< OWLOntologyChange > **resolveConflictIncrementally** (PelletReasoner reasoner, OWLOntology↩ Change change)

3.1.1 Detailed Description

Interface each new ConflictRule has to implement.

The documentation for this interface was generated from the following file:

- src/main/java/de/dailab/nsm/owlConflictSolver/impl/rule/ConflictRule.java

3.2 de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRuleRegistry Class Reference

Static Public Member Functions

- static List<? extends ConflictRule > **getRegisteredConflictRules** ()

3.2.1 Detailed Description

Simple rule registry to encapsulate ConflictRule's and providing an interface for ConflictSolver.

The documentation for this class was generated from the following file:

- src/main/java/de/dailab/nsm/owlConflictSolver/impl/rule/ConflictRuleRegistry.java

3.3 de.dailab.nsm.owlConflictSolver.impl.ConflictSolver Class Reference

Public Member Functions

- void resolveConflicts (PelletReasoner pellet)
- void resolveConflicts (MultivaluedMap< OWLOntology, OWLOntologyChange > changesByOntology)

3.3.1 Detailed Description

The ConflictSolver is the main entry point for solving given owl conflicts. While initialization it loads all applied conflict rules from the ConflictRuleRegistry.

The ConflictSolver can be invoked in two different ways: 1.ConflictSolver#resolveConflicts(PelletReasoner) to search for all conflicts in the whole ontology wrapped by the pellet reasoner

2.<OWLOntology, OWLOntologyChange> changesByOntology) (invoked through OWLOntologyChange↩ ListenerImpl) after any changes where applied to search only in relevant classes for conflicts.

3.3.2 Member Function Documentation

3.3.2.1 resolveConflicts() [1/2]

```
void de.dailab.nsm.owlConflictSolver.impl.ConflictSolver.resolveConflicts (
            PelletReasoner pellet )
```

Finds all conflicts for each given rule, they repair their known conflicts and return the resulting ontology changes.

Parameters

reasoner	Passes pellet reasoner instead of only the ontology for further implementations of other rules, so one can use the pre-implemented functions of pellet.

Returns

Resulting ontology changes for resolving the found conflicts.

3.3.2.2 resolveConflicts() [2/2]

```
void de.dailab.nsm.owlConflictSolver.impl.ConflictSolver.resolveConflicts (
            MultivaluedMap< OWLOntology, OWLOntologyChange > changesByOntology )
```

Finds all conflicts for each given rule, they repair their known conflicts and return the resulting ontology changes. It only concerns relevant entities in the signature of the given changes.

Parameters

changesByOntology	each ontology change mapped to the concerning ontology.

Returns

Resulting ontology changes will be applied to their ontologies.

The documentation for this class was generated from the following file:

- src/main/java/de/dailab/nsm/owlConflictSolver/impl/ConflictSolver.java

3.4 de.dailab.nsm.owlConflictSolver.test.ConflictSolverIntegrationTest Class Reference

Public Member Functions

- void **reflexiveISAConflictDetectionTest** () throws OWLOntologyCreationException
- void **reflexiveISAConflictDetectionByListenerTest** () throws OWLOntologyCreationException
- void **resolveConflictInExtendedOntologyIntegrationTest** () throws OWLOntologyCreationException, O↩ WLOntologyStorageException, IOException

3.4.1 Detailed Description

Tests the ConflictSolver concerning the implemented ISA relation conflict cases. There are two tests for each conflict case: For manual invocation of the ConflictSolver and for invocation by the OWLOntologyChangeListenerImpl.

The resolveConflictInExtendedOntologyIntegrationTest takes an ontology, solves conflicts and saves the result as a temp owl file. You can compare before and result in an ontology editor of your choice (e.g. Prot).

The documentation for this class was generated from the following file:

- src/test/java/de/dailab/nsm/owlConflictSolver/test/ConflictSolverIntegrationTest.java

3.5 de.dailab.nsm.owlConflictSolver.test.ConflictSolverPerformanceTest Class Reference

Public Member Functions

- void **reflexiveISAConflictDetectionPerformanceTest** () throws OWLOntologyCreationException

3.5.1 Detailed Description

Configure by changing the AMOUNT_TEST_CYCLES value.

The test assumes that the regarding ontology with the given amount of cycles was already created by the Cyclic↩ OntologyGenerator and can be found in the same directory the generator saved it in.

The documentation for this class was generated from the following file:

- src/test/java/de/dailab/nsm/owlConflictSolver/test/ConflictSolverPerformanceTest.java

3.6 de.dailab.nsm.owlConflictSolver.test.CyclicOntologyGenerator Class Reference

Static Public Member Functions

- static void **main** (String[ ] args)

Protected Member Functions

- final String **REFLEXIVE_ISA_CONFLICT_GENERATED_ONTOLOGY** (long cyles)

3.6.1 Detailed Description

Generates an ontology with [CYLES_AMOUNT] reflexive class membership (is_a) cycles. Each created owl concept contains only one reflexive is_a relation.

Configure the CYLES_AMOUNT value and run the main() to start.

The resulting ontology will be saved as [FILE_NAME_SAVING_PREFIX] + [CYLES_AMOUNT] + [FILE_NAME_↩ SUFFIX].

You need to prepare the file name prefix a bit (see [FILE_NAME_LOADING_PREFIX]) to load the ontology again because of the building configuraton in the pom.xml.

The concepts will be created as: Entity0; Entity1 is_a Entity1 && is_a Entity0; Entity2 is_a Entity2 && is_a Entity1; ...

The documentation for this class was generated from the following file:

- src/test/java/de/dailab/nsm/owlConflictSolver/test/CyclicOntologyGenerator.java

3.7 de.dailab.nsm.owlConflictSolver.util.Messages Class Reference

Static Public Attributes

- static String **resolvedNonReflexiveISAConflict** = "Resolved reflexive ISA conflict for class: "

3.7.1 Detailed Description

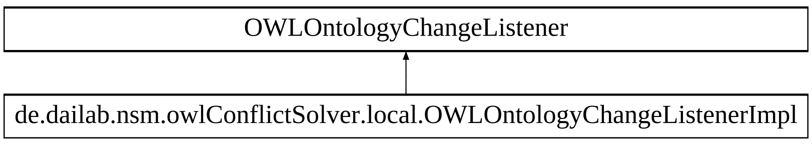
Combines all messages of the conflict solving routine to add Native Language Support (NLS).

The documentation for this class was generated from the following file:

- src/main/java/de/dailab/nsm/owlConflictSolver/util/Messages.java

3.8 de.dailab.nsm.owlConflictSolver.local.OWLOntologyChangeListenerImpl Class Reference

Inheritance diagram for de.dailab.nsm.owlConflictSolver.local.OWLOntologyChangeListenerImpl:


OWLOntologyChangeListener
de.dailab.nsm.owlConflictSolver.local.OWLOntologyChangeListenerImpl

Public Member Functions

- void **ontologiesChanged** (List< ? extends OWLOntologyChange > impendingChanges) throws OWL↩ Exception

3.8.1 Detailed Description

Need to be applied to an ontology. Then it listens to ontology changes and will get invoked after the changes will get applied. It maps the changes to the related ontologies and passes them to the ConflictSolver to resolve known conflicts. It will get invoked as long as there are no conflict changes anymore, so there is a last call after applying the conflict changes to check that no new conflicts were created.

The documentation for this class was generated from the following file:

- src/main/java/de/dailab/nsm/owlConflictSolver/local/OWLOntologyChangeListenerImpl.java

3.9 de.dailab.nsm.owlConflictSolver.test.PelletConflictDetectionTest Class Reference

Public Member Functions

- void **reflexiveISAConflictDetectionTest** () throws OWLOntologyCreationException
- void **disjointISAConflictDetectionTest** () throws OWLOntologyCreationException
- void **cyclicISAConflictDetectionTest** () throws OWLOntologyCreationException
- void **multipleInheritanceConflictDetectionTest** () throws OWLOntologyCreationException
- void **inheritancePropertyConflictDetectionTest** () throws OWLOntologyCreationException

3.9.1 Detailed Description

Test if pellet already detects certain conflict cases concerning the ISA relation. If pellet won't detect anything, it must hold that:

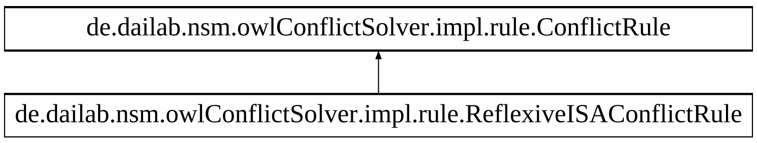
1. The only unsatisfiable class for pellet should be owl:Nothing.
2. Pellet shouldn't detect an inconsistency (pellet remains consistent).

The documentation for this class was generated from the following file:

- src/test/java/de/dailab/nsm/owlConflictSolver/test/PelletConflictDetectionTest.java

3.10 de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule Class Reference

Inheritance diagram for de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule:


de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule
de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule

Public Member Functions

- boolean isApplicable (OWLOntologyChange change)
- List< OWLOntologyChange > resolveConflicts (PelletReasoner reasoner)
- List< OWLOntologyChange > resolveConflictIncrementally (PelletReasoner reasoner, OWLOntology←Change change)

3.10.1 Detailed Description

Rule to resolve reflexive ISA relation conflicts by removing them. Rule can be invoked in two different ways← : 1.ReflexiveISAConflictRule#resolveConflicts(PelletReasoner) to search for all conflicts in the whole ontology wrapped by the pellet reasoner

2.ReflexiveISAConflictRule#resolveConflictsIncrementally(PelletReasoner, List) (invoked through OWLOntology←ChangeListenerImpl) after any changes where applied to search only in relevant classes for conflicts. In this case the rule determines if its applicable ReflexiveISAConflictRule#isApplicable(List) for at least one of the changes to save unnecessary computation time.

3.10.2 Member Function Documentation

3.10.2.1 isApplicable()

```
boolean de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule.isApplicable (
            OWLOntologyChange change )
```

Rule is only applicable to AddAxioms (because a reflexive IS_A relation won't appear by removing axioms).

Implements de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule.

3.10.2.2 resolveConflictIncrementally()

```
List<OWLOntologyChange> de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule.↩
resolveConflictIncrementally (
            PelletReasoner reasoner,
            OWLOntologyChange change )
```

Finds all reflexive ISA conflicts, repairs them (by removing) and returns the resulting ontology changes. It only concerns relevant OWLClasses in the signature of the given change.

Parameters

reasoner	Passes pellet reasoner instead of only the ontology for further implementations of other rules, so one can use the pre-implemented functions of pellet.
change	Ontology was changed anyhow, the OWLOntologyChangeListenerImpl registered them and invokes this function. The changes are already applied to the ontology, so this is a post-changes function. The given change is one of these changes and applicable to this conflict rule (see ReflexiveISAConflictRule.isApplicable(change)).

Returns

Resulting ontology changes for resolving the found conflicts.

Implements de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule.

3.10.2.3 resolveConflicts()

```
List<OWLOntologyChange> de.dailab.nsm.owlConflictSolver.impl.rule.ReflexiveISAConflictRule.↩
resolveConflicts (
            PelletReasoner reasoner )
```

Finds all reflexive ISA conflicts, repairs them (by removing) and returns the resulting ontology changes.

Parameters

reasoner	Passes pellet reasoner instead of only the ontology for further implementations of other rules, so one can use the pre-implemented functions of pellet.

Returns

Resulting ontology changes for resolving the found conflicts.

Implements de.dailab.nsm.owlConflictSolver.impl.rule.ConflictRule.

The documentation for this class was generated from the following file:

- src/main/java/de/dailab/nsm/owlConflictSolver/impl/rule/ReflexiveISAConflictRule.java

3.11 de.dailab.nsm.owlConflictSolver.test.TestDataProvider Class Reference

Protected Member Functions

- PelletReasoner **loadOntologyWithPellet** (String filePath) throws OWLOntologyCreationException

Static Protected Member Functions

- static OWLOntology **loadOntology** (String path) throws OWLOntologyCreationException

Protected Attributes

- final String **NON_CONFLICTED_ANIMALS_ONTOLOGY** = getClass().getResource("/owl/pinguin_vogel_↩ without_conflict.owl").getPath()
- final String **REFLEXIVE_ISA_CONFLICT_ONTOLOGY** = getClass().getResource("/owl/reflexive_isa_↩ conflict.owl").getPath()
- final String **DISJOINT_ISA_CONFLICT_ONTOLOGY** = getClass().getResource("/owl/disjoint_conflict.↩ owl").getPath()
- final String **CYCLIC_ISA_CONFLICT_ONTOLOGY** = getClass().getResource("/owl/extended_cyclic_isa_↩ conflict.owl").getPath()
- final String **MULTIPLE_INHERITANCE_CONFLICT_ONTOLOGY**
- final String **INHERITENCE_PROPERTY_CONFLICT_ONTOLOGY** = getClass().getResource("/owl/property↩ _conflict.owl").getPath()
- final String **COMPLETE_EXTENDED_CONFLICTED_ONTOLOGY**

Static Protected Attributes

- static final String **VOGEL_ONTOLOGY_ENTITY** = "http://www.dailab.de/ontologies/class#Vogel"
- static final String **PINGUIN_ONTOLOGY_ENTITY** = "http://www.dailab.de/ontologies/class#Pinguin"

3.11.1 Detailed Description

Manages loading of test datas and test ontologies.

3.11.2 Member Data Documentation

3.11.2.1 COMPLETE_EXTENDED_CONFLICTED_ONTOLOGY

```
final String de.dailab.nsm.owlConflictSolver.test.TestDataProvider.COMPLETE_EXTENDED_CONFLIC↩
TED_ONTOLOGY  [protected]
```

Initial value:

```
= getClass().getResource("/owl/complete_extended_animal_ontology.owl")
        .getPath()
```

3.11.2.2 MULTIPLE_INHERITANCE_CONFLICT_ONTOLOGY

```
final String de.dailab.nsm.owlConflictSolver.test.TestDataProvider.MULTIPLE_INHERITANCE_CONF↩
LICT_ONTOLOGY  [protected]
```

Initial value:

```
= getClass().getResource("/owl/multiple_inheritance_conflict.owl")
        .getPath()
```

The documentation for this class was generated from the following file:

- src/test/java/de/dailab/nsm/owlConflictSolver/test/TestDataProvider.java

Index