DL ABDUCTION API

Zuzana Hlávková
Fakulta matematiky, fyziky a informatiky UK
2022

Školiteľ: doc. RNDr. Martin Homola, PhD.

Konzultant: Mgr. Júlia Pukancová, PhD.

OBSAH

- Motivácia
- Cieľ
- Literatúra
- Práca
- Záver

MOTIVÁCIA, CIEĽ

- Abduktívne inferenčné stroje pre DL (napr. AAA, MXP-MHS, MergeXplain, QuickXplain)
- Ich integrácia do rôznych praktických nástrojov je však zatiaľ len na úrovni ad hoc
- Potreba API (napr. ako OWL API) - článok Bada, Mungall, Hunter 2008

- API pre abdukciu v deskripčných logikách + jeho implementácia

LITERATÚRA

1. Rudolph, S., 2011. **Foundations of description logics**. In Reasoning Web International Summer School (pp. 76-136). Springer, Berlin, Heidelberg.
2. Peirce, C. S. (1878), '**Deduction, induction, and hypothesis'**, Popular science monthly 13, 470–482.
3. Pukancová, J. and Homola, M., 2020**. The AAA ABox Abduction Solver**. KI-Künstliche Intelligenz, pp.1-6.
4. **Horridge, M., Bechhofer, S., 2011. The OWL API: A Java API for OWL ontologies. Semantic Web 2(1):11-21**
5. **Elsenbroich, C., Kutz, O. and Sattler, U., 2006. A case for abductive reasoning over ontologies. In: OWLED*06**
6. **J. Pukancová, M. Homola. Tableau-Based ABox Abduction for the ALCHO Description Logic. In: Description Logics 2017.**
7. **Schekotihin, K., Rodler, P. and Schmid, W., 2018. Ontodebug: Interactive ontology debugging plug-in for protégé. In International Symposium on Foundations of Information and Knowledge Systems (pp. 340-359). Springer, Cham.**
8. **M. Homola, J. Pukancová, Júlia Gablíková, Katarína Fabianová. Merge, Explain, Iterate. In: Description Logics, 2020.**
9. **Del-Pinto, W. and Schmidt, R.A., 2019. ABox abduction via forgetting in ALC. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 2768-2775).**
10. **Bada, Michael & Mungall, Chris & Hunter, Lawrence. (2008). A Call for an Abductive Reasoning Feature in OWL-Reasoning Tools toward Ontology Quality Control..**

PRÁCA

1. Literatúra + problematika
2. Vyabstrahovať funkcionalitu abduktívnych inferenčných strojov pre DL.
3. Navrhnúť knižnicu pre API rozhranie pre integráciu abuduktívnej inferencie do iných softvérov.
4. Implementovať API.
5. Implementovať navrhnuté API rozhranie do jedného inferenčného stroja.

Kapitoly

FUNKCIONALITA ABDUKTÍVNYCH INFERENČNÝCH STROJOV PRE DL

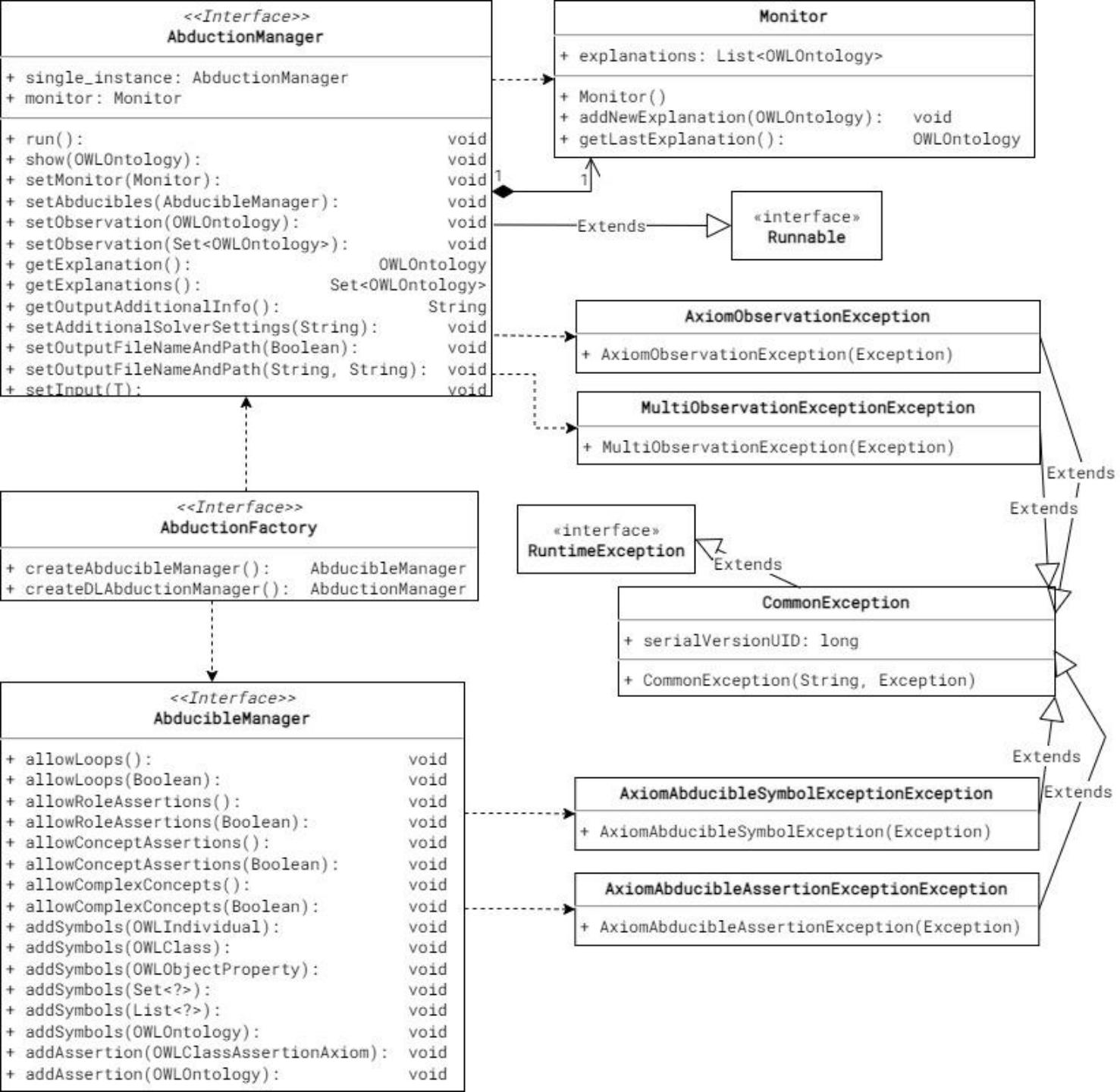
1. Input - K = ontológia $K \cup E \models O$
2. Output – explanations - E = ?
3. Observation - 0 = {jane: Mother}
4. Abducibles (loops, role assertions, concept assertions, complex concepts, concept complement, enumerations of assertions, individuals, axioms, …)
5. Solver's internal settings (reduction approach, timeout, …)

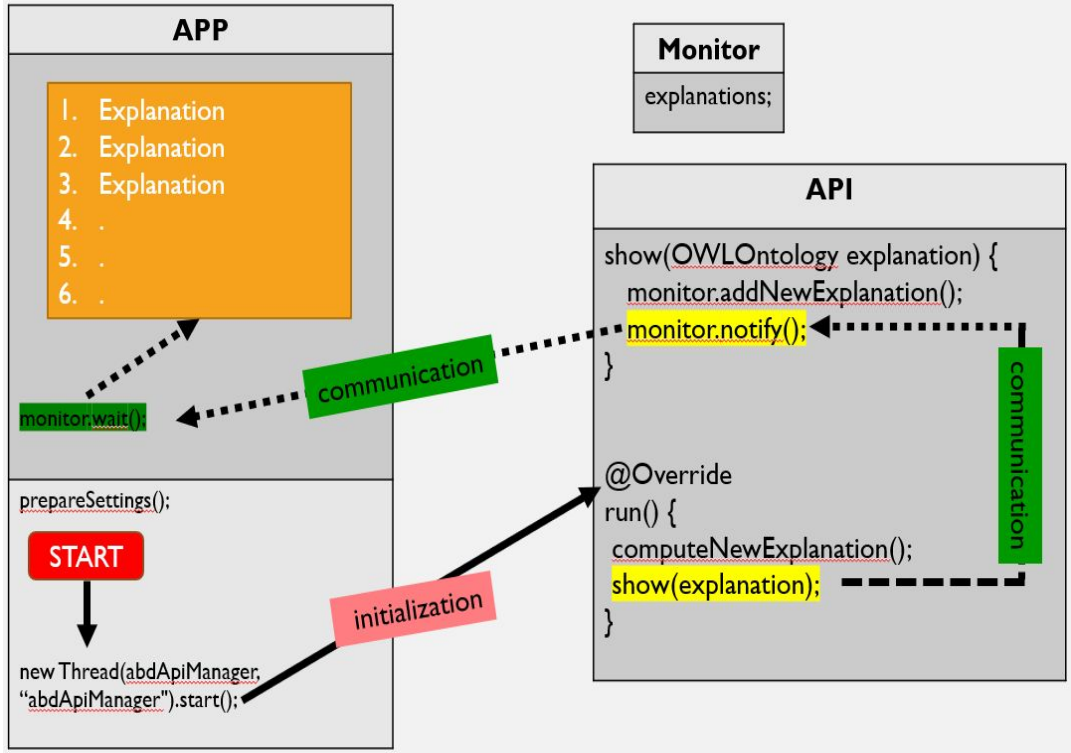
NÁVRH API - java 11

1. input - .owl súbor, OWLOntology z OWLApi
2. output - množiny axiómov ako OWLOntology
3. observation - single, multiple observations (OWLOntology), exceptions
4. abducibles – 3 možnosti ako s nimi pracovať.
   - switches – (loops, role assertions, concept assertions, complex concepts, concept complement)
   - enumerate abducibles – napr. 4 koncepty a 1 individuál
   - enumerate assertions – napr. jack nie je rodičom
5. solver's internal settings (string)

INFERENČNÉ STROJE

1. AAA ABox abduction solver
2. MHS-MXP
3. solvers based on MergeXplain, QuickXplain (e.g.: OntoDebug)
4. ABox Abduction via Forgetting in ALC - F -> internal setting
5. Solvers from our colleagues


<<Interface>>
AbductionManager
+ single_instance: AbductionManager
+ monitor: Monitor
+ run(): void
+ show(OWLOntology): void
+ setMonitor(Monitor): void
+ setAbducibles(AbducibleManager): void
+ setObservation(OWLOntology): void
+ setObservation(Set<OWLOntology>): void
+ getExplanation(): OWLOntology
+ getExplanations(): Set<OWLOntology>
+ getOutputAdditionalInfo(): String
+ setAdditionalSolverSettings(String): void
+ setOutputFileNameAndPath(Boolean): void
+ setOutputFileNameAndPath(String, String): void
+ setInput(T): void
Monitor
+ explanations: List<OWLOntology>
+ Monitor()
+ addNewExplanation(OWLOntology): void
+ getLastExplanation(): OWLOntology
1
1
Extends
«interface»
Runnable
AxiomObservationException
+ AxiomObservationException(Exception)
MultiObservationExceptionException
+ MultiObservationException(Exception)
Extends
Extends
«interface»
RuntimeException
Extends
CommonException
+ serialVersionUID: long
+ CommonException(String, Exception)
Extends
Extends
<<Interface>>
AbductionFactory
+ createAbducibleManager(): AbducibleManager
+ createDLAbductionManager(): AbductionManager
<<Interface>>
AbducibleManager
+ allowLoops(): void
+ allowLoops(Boolean): void
+ allowRoleAssertions(): void
+ allowRoleAssertions(Boolean): void
+ allowConceptAssertions(): void
+ allowConceptAssertions(Boolean): void
+ allowComplexConcepts(): void
+ allowComplexConcepts(Boolean): void
+ addSymbols(OWLIndividual): void
+ addSymbols(OWLClass): void
+ addSymbols(OWLObjectProperty): void
+ addSymbols(Set<?>): void
+ addSymbols(List<?>): void
+ addSymbols(OWLOntology): void
+ addAssertion(OWLClassAssertionAxiom): void
+ addAssertion(OWLOntology): void
AxiomAbducibleSymbolExceptionException
+ AxiomAbducibleSymbolException(Exception)
AxiomAbducibleAssertionExceptionException
+ AxiomAbducibleAssertionException(Exception)


APP
1. Explanation
2. Explanation
3. Explanation
4. .
5. .
6. .
monitor.wait();
communication
prepareSettings();
START
new Thread(abdApiManager, "abdApiManager").start();
initialization
Monitor
explanations;
API
show(OWLOntology explanation) {
monitor.addNewExplanation();
monitor.notify();
}
@Override
run() {
computeNewExplanation();
show(explanation);
}
communication

MANAGERS

```
//OWL API
IRI IOR = IRI.create("http://my-test-web.test");
OWLOntologyManager man = OWLManager.createOWLOntologyManager();
OWLOntology o = man.createOntology(IOR);
OWLDataFactory df = o.getOWLOntologyManager().getOWLDataFactory();

AbductionFactory dLAbductionFactory ;//= new AbductionFactory();
// Abduction API
AbductionManager dLAbductionManager = dLAbductionFactory.createDLAbductionManager();
```

INPUT, OBSERVATIONS

```
// input
// .owl or OWLOntology
dLAbductionManager.setInput(new File("example-input.owl"));
OWLOntology inputOntology = man.createOntology(IOR);
dLAbductionManager.setInput(inputOntology);
// observation/s
OWLOntology observationOntology = man.createOntology(IOR);
Set<OWLOntology> observationOntologySet = new HashSet<>();
try {
   dLAbductionManager.setObservation(observationOntologySet);
} catch (CommonException ex) {
   throw new CommonException("Solver exception: ", ex);
}
```

ABDUCIBLES - switches (1)

```
AbducibleManager abducibleManager =
dLAbductionFactory.createAbducibleManager();;

abducibleManager.allowLoops(); //(boolean default=true)
abducibleManager.allowLoops(false);
abducibleManager.allowRoleAssertions();
abducibleManager.allowConceptAssertions();
abducibleManager.allowComplexConcepts();
abducibleManager.allowConceptComplement();
```

ABDUCIBLES - abducible enumeration (2)

```
-   individual (OWLIndividual)
-   concept (OWLClass)
-   role (OWLObjectProperty)
-   or a hole ontology (list of abducibles - OWLOntology)

-   abducibleManager.addSymbol(<symbol>);
-   abducibleManager.addSymbols(<symbols>);
```

ABDUCIBLES - assertion enumeration (3)

```
-   exclusive option w.r.t. the previous option
-   abducibles info is rewritten
-   assertion (e.x. OWLClassAssertionAxiom)
-   or a hole ontology (list of assertions - OWLOntology)

-   abducibleManager.addAssertion(<assertion>);
-   abducibleManager.addAssertions(<assertions>);
```

SOLVER INTERNAL SETTINGS

```
// addtional settings for solver (optional)
dLAbductionManager.setAdditionalSolverSettings("internalSettings");
```

OUTPUT - NON-THREAD VERSION

```
// output
dLAbductionManager.getOutputAdditionalInfo(); //return solver internal info
(debug, etc.)
dLAbductionManager.getExplanations(); // return Set<OWLOntology>
```

OUTPUT - THREAD VERSION

```
// output - thread version

// At first monitor is set to AbductionManager.
dLAbductionManager.setMonitor(monitor);
// Then a new thread with target of AbductionManager instance is created at the
application.
new Thread(dLAbductionManager, "dLAbductionManager").start();
// Then method run in AbductionManager is executed and new explanations are computed.
// If any new explanation is computed BY a solver (overriding AbductionManager.run), it
will send a notification on a monitor.
// Meanwhile, application monitor is waiting for a new explanation in method Demo.run to
be showed.
```

ZHRNUTIE

Ostáva:

1. Spracovanie výsledkov v diplomovej práci.
2. Komunikácia so zahraničnými kolegami (1. feedback).
3. Prípadná úprava podľa feedback-u.
4. Implementovanie API rozhranie do jedného inferenčného stroja a do jedného vybraného nástroja - návod. (MHS-MXP).

ĎAKUJEM ZA POZORNOSŤ!