ORE 2014 Specification

May 16, 2014

This document describes the reasoner specification (primarily input and output formats) for the ORE 2014 Competition (see `http://vsl2014.at/pages/ORE-index.html`).

1 Reasoning Tasks

For ORE 2014, the following reasoning tasks are evaluated:

- Classification: arranging all classes of an ontology in a class hierarchy according to their subsumption relations.
- Consistency: checking the consistency of an ontology.
- Realisation: identifying the types of all individuals in an ontology.

The reasoning task are evaluated for those OWL 2 Profiles for which enough ontologies can be gathered and enough profile-specific reasoners participate in the ORE 2014 Competition.

2 Execution Environment

The evaluation will be carried out on a Linux 64bit system. For this, it is required that the reasoning systems must fulfil the following specifications:

- The reasoning systems must be executable on older Linux 64bit systems (e.g., Fedora 12, Java version 1.6).
- The reasoners must be encapsulated by a (shell) wrapper script, i.e., the execution of the wrapper script must trigger the reasoner.
- The reasoners should not use a hard-coded memory/time limit.

3 Input Format

The input format for the ORE 2014 Competition is specified as follows:

- The encoding format for ontologies and command line arguments is UTF-8.
- The ontologies are available in all fully fledged OWL 2 serialisation formats, i.e., reasoners can use the OWL 2 Functional Style, the OWL 2 XML, or the RDF/XML serialisation format.
- Ontologies may contain datatypes, but do not contain DL-safe/SWRL rules.
- All file paths are absolute, contain only ASCII-characters that are valid for files, and do not contain blank characters.
- The wrapper script is executed with the following arguments:
  1. The name of the reasoning task (<Operation>), i.e., either 'classification', 'consistency', or 'realisation'.
  2. Input file path of the ontology (<OntologyFile>).
  3. Output file path for the result (<Output>).

  For example, a wrapper script 'execReasoner.sh' is started with the command line:
  ./execReasoner.sh classification /ore/ont/pizza.owl /ore/out/result.dat

4 Output Format

The output format for the ORE 2014 Competition is specified as follows:

- The results must be encoded with UTF-8.
- Warnings/errors thrown by the reasoner must be written into the file <Output>_err, i.e., the given output file suffixed with '_err'. In particular, if the processing of the given ontology might be incomplete (e.g., the ontology contains axioms that are not supported by the reasoner), then this should be reflected by corresponding error/warning messages.
- It is guaranteed that all directories for the output files exist. If a file already exists, then the reasoner should overwrite the file.
- The reasoner should report to the console output (to stdout):
  1. The start message: 'Started <Operation> on <OntologyFile>'.
  2. The operation time: 'Operation time: <Time>'.
  3. The completion message: 'Completed <Operation> on <OntologyFile>'.

- The operation time (<Time>) should be measured in wall clock time, and should only be the value in milliseconds. Moreover, the operation time should represent the time elapsed from the moment preceding reasoner creation to the completion of the task at hand. That is, do not include ontology parsing time (unless some reasoner-specific pre-processing is done at this point), nor file serialization or socket communication time (where applicable).
- The result must be written to the <Output> file and should be as defined below for the different reasoning tasks.

4.1 Consistency Result Output Format

For consistency, the result output file (<Output>) must contain 'true' if the ontology is consistent and 'false' if the ontology is inconsistent.

4.2 Classification Result Output Format

For classification, the result output file (<Output>) must be an OWL 2 file (in OWL 2 Functional Syntax, or any other serialization format that can be read with the OWL API), which contains the class hierarchy with EquivalentClasses and SubClassOf axioms, i.e.,

- Classes that are computed as semantically equivalent, must be expressed as equivalent by EquivalentClasses/SubClassOf axioms.
  - For example, if A and B are equivalent, then this equivalence can be expressed by EquivalentClasses(A B) or also by the axioms SubClassOf(A B) and SubClassOf(B A).
- Classes that are direct sub-classes of other classes must be expressed as sub-classes of these other classes by SubClassOf axioms.
  - For instance, if A is a direct sub-class of B (i.e., there is no class C such that C is a super-class of A and a sub-class of B, and C is not equivalent to A or B), then this sub-class relationship can be expressed by SubClassOf(A B).
- SubClassOf(owl:Nothing A) axioms can be omitted.
- Note, if a class A is a direct sub-class of owl:Thing, then the result ontology must also contain SubClassOf(A owl:Thing).
- If the ontology is inconsistent, then owl:Thing is equivalent to owl:Nothing, i.e., the result ontology should contain SubClassOf(owl:Thing owl:Nothing) or EquivalentClasses(owl:Thing owl:Nothing).
- Note that for inconsistent ontologies the sub-class relationships of all other classes can be omitted.

- Indirect sub-class relationships MUST NOT be expressed, i.e., the reasoner has to perform the transitive reduction.

4.3 Realisation Result Output Format

For realisation, the result output file (<Output>) must be an OWL 2 file (in OWL 2 Functional Syntax, or any other serialization format that can be read with the OWL API), which contains all types of all individuals with ClassAssertion axioms, i.e.,

- If an individual a is an instance of a class A, then the result ontology has to contain the axiom ClassAssertion(A a), i.e., all indirect types of individuals must also be expressed by ClassAssertion axioms.

- If the ontology is inconsistent, then the result ontology must contain an axiom of the form SubClassOf(owl:Thing owl:Nothing), EquivalentClasses(owl:Thing owl:Nothing), or ClassAssertion(owl:Nothing a).