ORE 2014 Competition Framework

August 4, 2014

1 Introduction

OWL is a logic-based ontology language standard designed to promote interoperability, particularly in the context of the (Semantic) Web. The standard has encouraged the development of numerous OWL reasoning systems, and such systems are already key components of many applications. The 2nd OWL Reasoner Evaluation Competition (ORE 2014) has compared and evaluated 11 OWL reasoning systems on a data set containing a wide range of ontologies from the web, obtained through a standard web crawl and the Google Custom Search API, and a snapshot from the well known BioPortal repository. Some user submitted ontologies spiced up the corpus with particularly relevant and difficult test cases. The ontologies in the test set have been binned by profiles to enable the participation of reasoners specialised on the EL and DL profiles of OWL. The evaluation has been performed in three different categories that cover the important tasks of ontology classification, consistency checking, and ontology realisation (i.e., the computation of entailed types of individuals).

This read-me briefly describes the ORE 2014 Competition Framework, i.e., the framework that has been used to execute the ORE 2014 Live Competition, and explains how the evaluation can be reproduced.

Any kind of feedback (e.g., suggestions for improvements, bugs, . . .) can be send to the ORE 2014 competition organisers (e.g., `ore2014@easychair.org`).

2 License

'lgpl-3.0.txt', respectively. An online version is available at http://www.gnu.org/licenses/.

The ORE 2014 Competition Framework uses the following libraries in unmodified form:

1. Simple Logging Facade for Java (SLF4J), http://www.slf4j.org/, released under the MIT license.
2. Apache Logging Library for Java (log4j), http://logging.apache.org/log4j/1.2/, released under The Apache Software License, Version 2.0.
3. Apache Commons Exec, http://commons.apache.org/proper/commons-exec/, released under The Apache Software License, Version 2.0.
4. OWL API, http://owlapi.sourceforge.net, released under LGPL 3.0.
5. HermiT, http://hermit-reasoner.com/, released under LGPL 3.0.
6. Jetty, http://www.eclipse.org/jetty/, released under The Apache Software License, Version 2.0.
7. Joda-Time, http://www.joda.org/joda-time/, released under The Apache Software License, Version 2.0.
8. jCharts, http://jcharts.sourceforge.net/, released under Krysalis jCharts License.
9. Bootstrap, http://getbootstrap.com/, released under the MIT license.
10. WZ_Tooltip, http://www.walterzorn.de/en/tooltip/tooltip_e.htm, released under LGPL 2.1.

3 Overview and Usage

The competition framework is realised with Java and, therefore, it should be runnable on all Java supported platforms. However, it might be necessary to provide different reasoner configurations for different platforms. In particular, the wrapper scripts that trigger the reasoners must be executable on the platform on which the evaluation is executed (e.g., shell scripts on Unix/Linux, batch files on Windows). Of course, the reasoner must be executable as well and the wrapper scripts should also enforce that the time and memory limits are adhered. More details abut the reasoner set-up can be found below.

The organisation of the competition framework is as follows:

- Starter scripts are stored in the 'scripts' folder and they can be used to start many components with default arguments.
- The 'data/reasoners/' and 'data/ontologies/' directories contain the reasoners and the ontologies, respectively.

- The 'data/queries/' directory contains classification, consistency, and realisation queries that refer to the ontologies in the 'data/ontologies/' directory.
- The evaluation data as well as the responses and results of the reasoners are written into sub-directories of the 'data/responses/' folder.
- Expected results are stored in the 'data/expectations/' directory.
- Data about competitions are stored in the 'data/competitions/' folder.
- The 'data/configs/' directory contains configurations for the execution of the evaluations (e.g., timeouts). The 'default-config.dat' configuration is loaded by default, other configurations can be used by passing their file name as additional argument to the competition framework (e.g., './test-classification.sh hermit-linux other-config.dat').
- Logs from the execution of the competition framework are appended to the 'log.txt' file in the 'data/logs/' directory.
- The 'data/conversions/' directory is used to cache converted ontologies in the corresponding serialisation format if this is requested by a reasoner.
- The 'data/templates/' directory contains template data for charts, web pages, etc.

The competition framework works in principle as follows (the examples use the syntax for Linux and should also work for Mac OS X, minor syntactical adjustments are, however, necessary for Windows, e.g., the syntactical replacement of '.sh' and 'linux' with '.bat' and 'windows', respectively):

1. Ontologies have to be stored in the 'data/ontologies/' directory, for which then the corresponding queries can be generated (e.g., with the scripts 'scripts/create-classification-queries.sh', 'scripts/create-consistency-queries.sh', 'scripts/create-realisation-queries.sh').
2. The reasoners have to be configured in the 'data/reasoners/' directory (see also reasoner set-up instructions below).
3. The competition file has to be specified in the 'data/competitions/' directory, i.e., it has to be specified which reasoners are evaluated for which queries.
4. The competition has to be started, e.g., by calling the script 'scripts/evaluate-competition.sh' with the competition file as first argument.
5. The competition framework tries to load the reasoners specified in the competition files and the corresponding queries in the 'data/queries/' directory. If specified in the competition file, then the loaded queries are also filtered and sorted.

6. The queries are then executed for each reasoner by the competition framework and the results of the reasoners are parsed and normalised. If expected results are available for the queries, then the hash codes of the reasoner results are compared to the expected results. The expected results have to be in corresponding subdirectories of the 'data/expectations/' directory.

7. The competition framework generates the evaluation data as soon as all queries for all reasoners have been executed. All results are stored in a subdirectory of the 'data/responses/competition-evaluations/' directory as specified in the competition file. The folder 'reasoner-responses' contains the reasoner responses, the folder 'new-expected-results' contains generated hash codes that can be used as new expected results (e.g., by copying them into the 'data/expectations/' directory), and the 'evaluation-results' folder contains all evaluation data (e.g., statistics, rankings, summaries, etc). Note that there is also a 'query-execution-time-sorted.txt' file in the 'evaluation-results' folder, which contains the list of queries sorted by the average execution time of the reasoners, i.e., the file can be referred by the competition file in order to specify the order in which the queries have to be executed.

Note that the 'scripts/evaluate-competition.sh' executes a competition sequentially. You can also execute a competition in parallel by creating a competition server (e.g., with the script 'scripts/competition-server.sh') and by starting (several) clients (e.g., with the script 'scripts/client.sh'). The clients can be started on different machines/computers, however, it is required that they are started within the same directory and that the path to this directory is the same for all clients and the competition server (otherwise some data can possibly not be correctly accessed). You can also create a web server in order to see the current status of the competition (e.g., by executing the script 'scripts/status-web-server.sh'). Note that as many clients are required as reasoners are used in the competition if 'org.semanticweb.ore.competition.executorperreasoner' is set to 'TRUE' in the configuration file.

4 ORE 2014 Live Competition

The ORE 2014 Live Competition consists of the following components and packages:

- The ORE 2014 Competition Framework, available at `https://github.com/andreas-steigmiller/ore-2014-competition-framework`, encapsulates the binaries, source code, and documentation to reproduce the competition (this read-me is also part of this framework).

- The ORE 2014 Dataset, available at `https://zenodo.org/record/10791`, contains (a superset of) the evaluated ontologies.

- The ORE 2014 Reasoners package, available at `https://zenodo.org/record/10791`, contains the reasoners that have participated in the ORE 2014 Live Competition.
- The ORE 2014 Live Competition Queries package, available at `https://zenodo.org/record/11133`, contains the queries and configuration files for the ORE 2014 Live Competition.
- The ORE 2014 Live Competition Results package, available at `https://zenodo.org/record/10791`, contains the logs, reasoner responses, and evaluation results of the ORE 2014 Live Competition.

Note that, depending on the purpose, only some of the packages are required. A package can be "installed" by downloading and unzipping it in the root directory of the competition framework. However, the ontologies of the ORE 2014 Dataset must be moved/copied into a subdirectory named with 'ore2014' within the ontologies folder of the competition framework (in order to avoid the adaptation of scripts), i.e., the content of 'dataset/files/' must be copied/moved to 'data/ontologies/ore2014/'.

4.1 Replay

All data from the ORE 2014 Live Competition are recorded in log files and it is possible to replay/re-watch the ORE 2014 Live Competition. For this, the competition framework and the results of the ORE 2014 Live Competition (i.e., the ORE 2014 Live Competition Results package) are required. Unzipping the results package extracts the evaluation results into the 'ore2014-live-competition-evaluation-results' folder into the response directory ('data/responses/'). The replay can be started with the script 'scripts/replay-server-ore2014-start-all.sh', which streams the status updates to all clients connected to the this server instance. In particular, a web server can be started as a client (e.g., with the script 'scripts/local-status-web-server.sh') that receives the status updates and allows for re-watching the live competition with a browser at the address `http://localhost:8008/live.html`.

Note that there exist several other scripts that directly jump to other interesting points of time in the replay. For example, the script 'scripts/replay-server-ore2014-finished-all.sh' can be used for directly showing the final results of the competition. Of course, you can also adapt the scripts or create new ones to also jump to arbitrary points of time in the replay by simply adapting the specified time.

4.2 Sequential Execution

By using only one machine/computer, the results of the ORE 2014 Live Competition can be reproduced in a sequential way as follows:

- The reasoners, ontologies, and queries of the ORE 2014 Live Competition must be available in the 'data/reasoners/', 'data/ontologies/ore2014/', and 'data/queries/' directories, respectively, i.e., all packages except the ORE 2014 Live Competition Results package are required.

- The execution of the script 'scripts/evaluate-ore2014-competition.sh' evaluates all disciplines of the ORE 2014 Live competition by running each reasoner for each query of each discipline, i.e., this potentially requires a lot of time (probably several days).

Note that the time constraints of the competitions (e.g., the starting and (enforced) finishing time) are deactivated, which possibly influences the evaluation outcome a little bit. If new (valid) time constraints are set, they can be activated by setting 'AllowProcessingOnlyBetweenDateTimeInterval' in the competition files to 'TRUE'.

4.3 Parallel Execution

The evaluation of the ORE 2014 Live Competition can be reproduced as follows:

- The reasoners, ontologies, and queries of the ORE 2014 Live Competition must be available in the 'data/reasoners/', 'data/ontologies/ore2014/', and 'data/queries/' directories, respectively, i.e., all packages except the ORE 2014 Live Competition Results package are required.

- The competition server has to be started, e.g., by executing the script 'scripts/competition-server-ore2014.sh'.

- At least eleven competition clients have to be started (for each reasoner one), e.g., by executing the script 'scripts/local-client.sh'. The evaluation can also be executed with less clients if 'org.semanticweb.ore.competition.executorperreasoner' in the configuration file 'data/configs/default-config.dat' is set to 'FALSE'. However, then the evaluation is not running in real-time, but each reasoner is evaluated for each query step by step. The clients can also be started on different machines, but they have to be started within the same directory, e.g., by using a network drive that is mounted by all machines within the same directory. Note, if the clients are started on different machines, then the client has to be executed with the correct destination address, e.g., by replacing '127.0.0.1' in the client's starter script with the IP address from the machine where the competition server is running. Also note that if all clients are started on the same machine, then the memory usage potentially sums up, i.e., in the worst-case $11 * 10$ GBytes of RAM are required. The memory limit can also be adapted in the configuration file 'data/configs/default-config.dat' by setting a corresponding value for 'org.semanticweb.ore.execution.memorylimit', which, however, potentially also influences the outcome of the evaluation.

4.4 Query Generation

The queries for the ORE 2014 Live Competition are available with the ORE 2014 Live Competition Queries package, but they can also be generated from scratch by execution the script 'scripts/create-ore2014-queries.sh' (also part of the ORE 2014 Live Competition Queries package). In contrast to the other 'create-XYZ-queries.sh' scripts, this script calls a generator that creates only queries for ontologies in the 'data/ontologies/ore2014/' directory. Furthermore, a FileOrderFile is used to create only a certain number of queries according to the order specified in this file. In particular, the corresponding query generator is called with the following parameters:

- FileOrderFile: the file in which the order of the ontologies is specified.
- NumberOfQueries: the number of queries that have to be generated.
- QueryType: the type of queries that have to be generated. Must be one of 'classification', 'realisation', or 'consistency'.
- ProfileType: the profile for which the queries have to be generated. Must be one of 'DL' or 'EL'.

The 'FileOrderFile's used in 'scripts/create-ore2014-queries.sh' script correspond to the 'fileorder.txt' files from the ORE 2014 Dataset (e.g., the 'ore2014-classification-dl-fileorder.txt' corresponds to 'dataset/pure_dl/classification/fileorder.txt').

The expected results for these queries are obtained by evaluating the competitions and by using the generated hash codes of the directory 'random-majority-voting' within the 'new-expected-results' folder (i.e., by simply coping the content of the 'random-majority-voting' folder into the 'data/expectations/' directory).

5 Reasoner Set-up Instructions

In order to make a reasoner runnable for the framework, the following steps are necessary:

1. Create a new folder in the 'data/reasoners/' directory, named by the reasoner/the reasoner configuration (e.g., 'elk-linux').
2. Specify the settings for the new reasoner in the 'reasoner.dat' file within the reasoner's directory (e.g., 'data/reasoners/elk-linux/reasoner.dat').
   - This can simply be archived by copying an existing reasoner configuration file (e.g., 'data/reasoners/hermit-linux/reasoner.dat') and modify some of the values (e.g., 'OutputPathName' to 'elk-linux', 'ReasonerName' to 'ELK', 'ProfileSupport' to 'EL', and 'DatatypeSupport' to 'FALSE'). Also see comments in the existing configuration files (e.g., 'data/reasoners/hermit-linux/reasoner.dat').
3. Create a starter/wrapper script (e.g., 'execReasoner.sh') in the reasoner folder that triggers the reasoner (also see ORE 2014 Specification).

- The wrapper script must be referenced in the reasoner configuration file ('reasoner.dat') by the 'StarterScript' setting, (e.g., './execReasoner.sh').
- The wrapper script is executed by the evaluation framework, i.e., it must be executable (e.g., apply 'chmod a+x execReasoner.sh').
- For OWL API-based reasoners, the wrapper script can directly execute the OREv2ReasonerWrapper ('OREv2ReasonerWrapper.jar'), which already implements the input and output specifications for the ORE 2014 Competition. Note, the OREv2ReasonerWrapper requires as first argument the corresponding OWLReasonerFactory class (e.g., 'org.semanticweb.elk.owlapi.ElkReasonerFactory') that has to be loaded with the Java Reflection API in order to create the OWLReasoner from the factory. Also note that the OREv2ReasonerWrapper may have to be recompiled such that all libraries of the reasoner are in the Classpath of the 'OREv2ReasonerWrapper.jar' file. The OREv2ReasonerWrapper can be build by the Ant script 'build-wrapper-v2.xml' (the libs of the reasoner must be copied into the 'lib' directory).

4. The execution of the reasoner can be tested by calling a corresponding starter script for a reasoning task with the reasoner folder as first argument (for example, 'scripts/test-classification.sh elk-linux').

   - If the reasoner configuration file is not named 'reasoner.dat' within the reasoner folder, then the path to this file must be used as first argument.
   - Optionally, the evaluation configuration file can be listed as second argument.

5. The reasoner folder (or the path to the reasoner configuration file) can then also be specified in the competition files.