COMPLETO
User Manual

Enrique Matos Alfonso and Giorgos Stamou

National Technical University of Athens (NTUA)
gardero@image.ntua.gr

Figure 1 shows a description of the COMPLETO system and its main workflow.

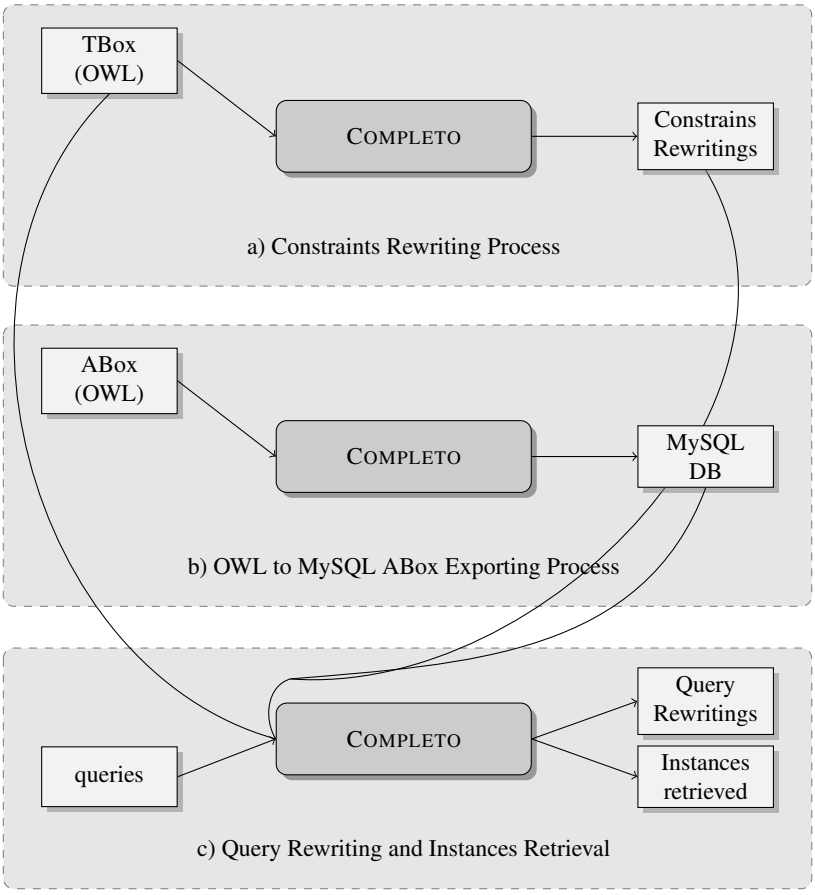

Fig. 1. Diagram of the COMPLETO system

Installing the Software

The software is contained in a jar[1] file and it needs to be executed by a java[2] compiler:

```
java -jar completo.jar \
   [other options]
```

The option `-Xmx` can help us increase the memory assigned to the process.

Running the Experiments

For running the experiments we need to know how to execute the main actions provided by the COMPLETO system. The TBoxes[3] and the ontologies with assertions[4] used in the experiments can be downloaded to provide the necessary inputs to the system.

Constraints Rewriting. Given a owl ontology with constraints in it, we can generate a file with the rewriting of the constraints. The command to execute is the following:

```
java -jar completo.jar \
   -o [ontology path] \
   (-q [queries path] -nconcepts) \
   -c [constraints path]
```

where the path for the constraints should refer to a non existing file where all the rewriting of the constraints will be written as queries. Also, if we want to generate a queries file with the negation of all the concepts in the ontology we then use `-q [queries path] -nconcepts`. Files with queries have the following format:

```
Q(?X) <- -AdministrativeStaff(?X).
Q(?X) <- -Article(?X).
Q(?X) <- -AssistantProfessor(?X).
Q(?X) <- -AssociateProfessor(?X).
```

and files with constraints have boolean queries describing the constraints:

```
_answer <- College(?X), ResearchGroup(?X).
_answer <- College(?X), researchProject(?X, ?_u0).
_answer <- AdministrativeStaff(?X), Article(?X).
_answer <- AdministrativeStaff(?X), Book(?X).
```

where negated atoms come with a minus "-" sign, variables with a "?" sign before the identifier. Queries files can also be built manually.

---

[1] http://image.ntua.gr/completo/completo.jar
[2] version 1.8 or more recent.
[3] http://image.ntua.gr/completo/tbox.zip
[4] http://image.ntua.gr/completo/ontofile.zip

OWL to MySQL ABox Exporting Process. In order to generate a SQL database with the assertions contained in an owl ontology we need to run:

```
java -jar completo.jar \
   -o [ontology path]  -eabox \
   -aboxname [abox SQL name]
```

The command will create a SQL database using MySQL software. The database will be used in the instance retrieval process for the queries. Additionally, we need a configuration file (`sqlconfig.properties`) containing some important data used to establish the connection to MySQL:

```
username=[username]
password=[password]
url=[url:port of the installed MySQL instance]
```

Also, for big databases we will need to increase the limits of the `thread_stack` and `max_allowed_packet` on the configurations files of MySQL.

Query Rewriting and Instance Retrieval. The main task in the experiments is related to rewriting queries and finding the instances of the result in SQL databases. In order to execute the task we need to use the following command:

```
java -jar completo.jar \
   -o [ontology path] -q [queries path]  \
   (-qi [index of the query to be rewritten])  \
   -c [constraints path] -aboxname [abox SQL name] \
   (-apath [path to output the answers of the queries])
```

where all the queries in the file will be rewritten one by one unless an index is specified with the `-qi` option is specified and in such a case only the query in the corresponding index will be rewritten.

In case we are only interested in the rewritings of the queries we should provide the file to output the rewritings and remove the information about the Abox:

```
java -jar completo.jar \
   -o [ontology path] -q [queries path]  \
   (-qi [index of the query to be rewritten])  \
   -c [constraints path] \
   (-qrewritings [path to output the rewritings])
```