Rewriting Queries with Negated Atoms Appendixes

Enrique Matos Alfonso and Giorgos Stamou

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

Appendix A

Theorem 1. *Let $C^{(1)}$ and $C^{(2)}$ be two clashing clauses with resolvent C' and $C^{(3)}$ a clashing clause with C'. Lets assume that $C^{(3)}$ only clashes with one of the two other clauses and without loss of generality we can assume that clause will be $C^{(2)}$. Then we have that*

$$Res(Res(C^{(1)}, C^{(2)}), C^{(3)}) = Res(C^{(1)}, Res(C^{(2)}, C^{(3)})) \ . \tag{1}$$

Proof. The proof of the theorem can be done by replacing the resolvents with the equivalent set expression as defined in the general resolution rule. Lets assume the clauses $C^{(1)}$ and $C^{(2)}$ clash on the sets L_1 and L_2 :

$$Res(C^{(1)}, C^{(2)}) = (C^{(1)}\sigma \setminus L_1\sigma) \cup (C^{(2)}\sigma \setminus L_2\sigma) \tag{2}$$

and that the clauses $C^{(2)}$ and $C^{(3)}$ clash on the set L'_2 and L_3 :

$$Res(C^{(2)}, C^{(3)}) = (C^{(2)}\sigma' \setminus L'_2\sigma') \cup (C^{(3)}\sigma' \setminus L_3\sigma') \ , \tag{3}$$

with

$$\sigma = mgu(L_1 \cup \overline{L_2}) \tag{4}$$

and

$$\sigma' = mgu(L'_2 \cup \overline{L_3}) \ . \tag{5}$$

Notice that on the clause $Res(C^{(1)}, C^{(2)})$ the elements that were introduced by $C^{(2)}$ are affected by the unifier σ therefore when we try to resolve that clause with $C^{(3)}$ the clashing literals will be $L'_2\sigma$ and L_3 yielding another mgu:

$$\gamma' = mgu(L'_2\sigma \cup L_3) \ . \tag{6}$$

We start by rewriting the left part of (1):

$$((C^{(1)}\sigma \setminus L_1\sigma) \cup (C^{(2)}\sigma \setminus L_2\sigma))\gamma' \setminus L'_2\sigma\gamma' \cup (C^{(3)}\gamma' \setminus L_3\gamma') \ ,$$

where $C^{(3)}\sigma = C^{(3)}$ and $L_3\sigma = L_3$ because the clause $C^{(3)}$ does not share variables with $C^{(1)}$ or $C^{(2)}$. Therefore,

$$((C^{(1)}\sigma \setminus L_1\sigma) \cup (C^{(2)}\sigma \setminus L_2\sigma))\gamma' \setminus L'_2\sigma\gamma' \cup (C^{(3)}\sigma\gamma' \setminus L_3\sigma\gamma') \ .$$

Now since $L_2'\sigma\gamma' = \overline{L_3}\sigma\gamma'$, using (5) we get

$$\sigma\gamma' = \sigma'\gamma \ . \tag{7}$$

Also, because of (7) and (4) we can affirm that $L_1\sigma'\gamma = \overline{L_2}\sigma'\gamma$.

The unifier γ needs to be an mgu for the problem $L_1\sigma'$ and $\overline{L_2}\sigma'$:

$$\gamma = mgu(L_1\sigma' \cup \overline{L_2}\sigma') \ , \tag{8}$$

otherwise it would mean that there is a *different* unifier $\gamma^0 \succeq \gamma$ such that

$$\gamma^0 = mgu(L_1\sigma' \cup \overline{L_2}\sigma') \ . \tag{9}$$

Implying there is another substitution γ^1 such that $\sigma'\gamma^0 = \sigma\gamma^1$ and consequently $(L_2'\sigma)\gamma^1 = (\overline{L_3}\sigma)\gamma^1$.

Yet, because of (6) can affirm that $\gamma' \succeq \gamma^1$ and also $\sigma\gamma' \succeq \sigma\gamma^1$ which would mean that $\sigma'\gamma \succeq \sigma'\gamma^0$ and also $\gamma \succeq \gamma^0$ which ensures that γ will be more general than all the other unifiers of the problem. Thus (8) holds.

Additionally, we know $C^{(1)}$ does not clash with $C^{(3)}$ so $C^{(1)}\sigma\gamma'$ does not contain $L_2'\sigma\gamma'$ nor $C^{(3)}\sigma\gamma'$ contains $L_2\sigma\gamma'$:

$$((C^{(1)}\sigma\gamma' \setminus L_1\sigma\gamma' \setminus L_2'\sigma\gamma') \cup (C^{(2)}\sigma\gamma' \setminus L_2\sigma\gamma' \setminus L_2'\sigma\gamma')) \cup (C^{(3)}\sigma\gamma' \setminus L_3\sigma\gamma') \ ,$$

$$((C^{(1)}\sigma\gamma' \setminus L_1\sigma\gamma') \cup (C^{(2)}\sigma\gamma' \setminus L_2\sigma\gamma' \setminus L_2'\sigma\gamma')) \cup (C^{(3)}\sigma\gamma' \setminus L_3\sigma\gamma') \ ,$$

$$(C^{(1)}\sigma\gamma' \setminus L_1\sigma\gamma') \cup (C^{(2)}\sigma\gamma' \setminus L_2\sigma\gamma' \setminus L_2'\sigma\gamma') \cup (C^{(3)}\sigma\gamma' \setminus L_3\sigma\gamma' \setminus L_2\sigma\gamma') \ ,$$

$$(C^{(1)}\sigma\gamma' \setminus L_1\sigma\gamma') \cup ((C^{(2)}\sigma\gamma' \setminus L_2'\sigma\gamma') \cup (C^{(3)}\sigma\gamma' \setminus L_3\sigma\gamma')) \setminus L_2\sigma\gamma' \ .$$

Then, using (7) we get:

$$(C^{(1)}\sigma'\gamma \setminus L_1\sigma'\gamma) \cup ((C^{(2)}\sigma'\gamma \setminus L_2'\sigma'\gamma) \cup (C^{(3)}\sigma'\gamma \setminus L_3\sigma'\gamma)) \setminus L_2\sigma'\gamma$$

and we can extract γ because σ' is a unifier of L_2' and $\overline{L_3}$ so it will not affect the set subtraction operations based on L_2' and L_3:

$$C^{(1)}\gamma \setminus L_1\gamma \cup ((C^{(2)}\sigma' \setminus L_2'\sigma') \cup (C^{(3)}\sigma' \setminus L_3\sigma'))\gamma \setminus L_2\sigma'\gamma \ ,$$

leading to:

$$(C^{(1)}\gamma \setminus L_1\gamma) \cup Res(C^{(2)}, C^{(3)})\gamma \setminus L_2\sigma'\gamma \ ,$$

where we know γ is the mgu for L_1 and $\overline{L_2}\sigma'$. Thus, finally we get the right part of (1):

$$Res(C^{(1)}, Res(C^{(2)}, C^{(3)})) \ .$$

□

Appendix B

Lemma 1. *The answers of the elements q_i of the constraint saturation of a query $\mathcal{C}_q$ are also answers of the original query q i.e.* $\forall i\ ans(q_i, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \subseteq ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D})$.

Clearly since the elements q_i were built using a linear resolution derivation starting with a clause corresponding to $\neg q$ and using as side clauses the clauses corresponding to the constraints $\mathcal{C}$ of our system, we can affirm that $\mathcal{C} \models q_i \rightarrow q$. Therefore, whenever we have that $() \in ans(q_i, (\mathcal{R}, \mathcal{C}), \mathcal{D})$ then we it will also be the case that $() \in ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D})$.

Theorem 2. *For a knowledge base $(\mathcal{R}, \mathcal{C})$ and a strongly disconnected query q with respect to the knowledge base, if $\mathcal{C}$ contains all the possible rewritings of the queries corresponding to the constraints in it then the constraints saturation $\mathcal{C}_q$ contains all the answers of q:*

$$ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) = ans(\mathcal{C}_q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \textit{ for all } \mathcal{D} \ .$$

Proof. Lemma 1 ensures $ans(\mathcal{C}_q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \subseteq ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D})$. Then, we need to focus on proving $ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \subseteq ans(\mathcal{C}_q, (\mathcal{R}, \mathcal{C}), \mathcal{D})$.

The proof is based on showing that a resolution derivation starting in $\neg q$ and ending in the empty clause can be rearranged using Theorem 1 so that the first resolution steps are applied using constraints from $\mathcal{C}$. After those steps we can affirm that the clause will correspond to a query in the constraint saturation of q, so there will also be a resolution derivation starting from a clause corresponding to a query in $\mathcal{C}_q$ and ending in the empty clause i.e. $() \in ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \rightarrow () \in ans(\mathcal{C}_q, (\mathcal{R}, \mathcal{C}), \mathcal{D})$.

Suppose that:

$$() \in ans(q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \tag{10}$$

and

$$() \notin ans(\mathcal{C}_q, (\mathcal{R}, \mathcal{C}), \mathcal{D}) \ . \tag{11}$$

We can affirm using (10) that $(\mathcal{R}, \mathcal{C}), \mathcal{D} \models q$ and it ensures the existence of a linear resolution derivation starting with the clause $C^{(0)} = \neg q$, ending in the empty clause $\perp$

$$C^{(0)}, C^{(1)}, \ldots, C^{(m')}, \perp \ .$$

that uses the clauses corresponding to $(\mathcal{R}, \mathcal{C}), \mathcal{D}$ and $\neg q$.

The only way to reach the empty clause is by performing resolution with respect to the clauses corresponding to the facts $\mathcal{D}$ that have the following shape: $[a(\mathbf{t}')]$. They decrease the size of the clauses in the linear resolution derivation. The initial clause has also positive atoms (corresponding to the negated atoms in the original query) and the only way to get rid of them is by doing resolution with respect to one of the literals in a clause that corresponds to a constraint. Such step will decrease the number of positive atoms in the clause of the derivation and probably will introduce more negative atoms. Since our query has m negated atoms, in our derivation we need at least m resolution steps to get rid of the positive literals.

If we take a closer look at those resolution steps $i_1, \ldots, i_{m'}$ involving constraint clauses we can try to reorganize them so that they are performed as early as possible.

In case the constraint clause used at step i_j can resolve with the initial clause $C^{(0)}$ we can perform the step on position 1 and shift the other steps of the linear derivation. The resulting clause at step i_j will be the same.

On the other hand, if some resolution steps with clauses corresponding to rules need to be applied before we could apply a certain resolution step with a constraint clause, then based on Theorem 1 we could re-arrange those resolution steps by applying resolution first to the constraint clause and then the resulting clause can be used to apply resolution to the initial clause, resulting in the same clause at step i_j. Note that this modification turns the linear derivation into a tree but the shape of the derivation is not relevant. The clause that we obtain by applying resolution between the constraint clause and the rules will have only negative literals. So it will be equivalent to a constraint C' that can be deduced by the system i.e. $\mathcal{R}, \mathcal{C} \models C'$. But since $\mathcal{C}$ contains all its rewritings, we know that there will be a constraint $C'' \in \mathcal{C}$ such that $C'' \succeq C'$. In that case, the query corresponding to C' is a rewriting of the query corresponding to C''.

In case C'' clashes with $C^{(0)}$, instead of using C' in the resolution derivation we could use C'' which is a side clause that can resolve with the initial query and such step can again be inserted at position 1. On the other hand, if C'' does not clashes with $C^{(0)}$ it means that the resolution with C' yields a query that is also less general than C'' . Therefore, the resulting rewriting will have answers when a constraint of the system is violated i.e. the resulting rewriting would be inconsistent.

The other possible resolution step involving a $C^{(0)}$ would be to revolve it with itself but our query is strongly disconnected with respect to the knowledge base as hypothesis to avoid those cases.

We will end up in a derivation with at least m initial resolution steps with respect to constraint clauses. After those steps we will have a clause that needs to correspond to a query that is less general than one of the queries in the constraints saturation $\mathcal{C}_q$. It implies that also there is a resolution derivation that ends in the empty clause and starts with a clause corresponding to the negation of a query in $\mathcal{C}_q$ i.e. $() \in ans(\mathcal{C}_q, (\mathcal{R}, \mathcal{C}), \mathcal{D})$ which contradicts (11) and proves our theorem. □

Appendix C

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

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.

[1] http://image.ntua.gr/completo/completo.jar

[2] version 1.8 or more recent.

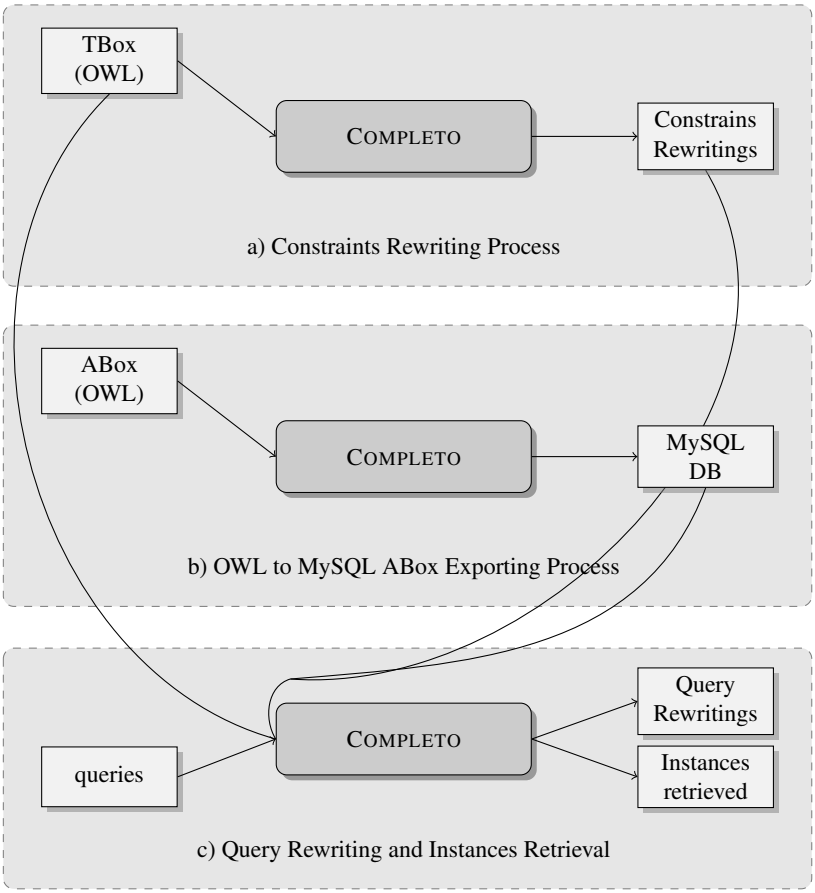

Fig. 1. Diagram of the COMPLETO system

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.

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:

[3] http://image.ntua.gr/completo/tbox.zip
[4] http://image.ntua.gr/completo/ontofile.zip

```
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])
```