Query Abduction for $\mathcal{ELH}_\perp$ Ontologies

Mahsa Chitsaz, Zhe Wang, Kewen Wang
School of ICT, Griffith University, Australia
mahsa.chitsaz@griffithuni.edu.au, {zhe.wang,k.wang}@griffith.edu.au

Introduction
Efficient Description Logic (DL) reasoners have been developed for reasoning tasks such as classification and query answering. However, the need for DL-based systems to be equipped by explanation algorithms is inevitable. One important explanation problem for DL ontologies, referred to as query explanation, can be formalised as abductive reasoning, or more precisely, query abduction. In particular, given an ontology and an observation (i.e., a query with an answer), an explanation to the observation is a set of facts that together with the ontology can entail the observation.

Several query abduction algorithms have been proposed and some complexity results obtained in the literature. One major stream of such algorithms are the tableau-based ones, and some complexity results obtained in the literature. One important explanation problem for DL ontologies, referred to as query abduction, can be formalised as

Definition 1 (QAP)  An instance of a query abduction problem (QAP) in $\mathcal{ELH}_\perp$ is a tuple $\langle K, Q(\vec{a}), \Sigma, \Delta \rangle$, where $K$ is an $\mathcal{ELH}_\perp$ KB, $Q(\vec{a})$ is a BCQ called the observation, $\Sigma \subseteq N_C \cup N_R$ is a finite set of predicates called abducible, and $\Delta \subseteq N_I$ is a finite set of constants called the domain.

A solution $E$ to the QAP is a set of facts over $\Sigma$ such that $(1) \text{ pred}(E) \subseteq \Sigma$, $(2) \text{ const}(E) \subseteq \Delta$, $(3) K \cup E$ is consistent, and $(4) K \cup E \models Q(\vec{a})$. Let $\text{ sol}(K, Q(\vec{a}), \Sigma, \Delta)$ be the set of solutions to the QAP. Moreover, we call $E$ minimal iff there is no solution $E' \in \text{ sol}(K, Q(\vec{a}), \Sigma, \Delta)$ s.t. $E' \subset E$.

For a datalog program $D$, a QAP in datalog and its (minimal) solutions are defined in the same way, so the set of solutions are denoted as $\text{ sol}(D, Q(\vec{a}), \Sigma, \Delta)$.

Computing Minimal Solutions
For an $\mathcal{ELH}_\perp$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with a normalized TBox, we will first transform $\mathcal{T}$ into a datalog program $D_T$ in a similar way to (Stefanoni, Motik, and Horrocks 2013), by replacing each existential axiom $A \subseteq \exists \vec{y}$ with two datalog rules $A(x) \rightarrow r(x, \vec{c}, B)$ and $A(x) \rightarrow B(\vec{c}, B)$, where $\vec{c}$ is a fresh constant. For an $\mathcal{ELH}_\perp$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the program $D_K = D_T \cup A$ is referred to as its datalog approximation. By this approximation, each solution to the QAP w.r.t. KB $\mathcal{K}$ is a solution to the corresponding QAP w.r.t. datalog program $D_K$ which formally presented as follows.

sets such as extended university ontology LSTW($n$) that has approximately $10^5 n$ ABox assertions. Our experimental results show that the algorithm is capable of handling query abduction problems for LSTW($n$) where $n$ is up to 100.

Query Abduction Problem
Consider countably infinite and mutually disjoint sets $N_C$, $N_R$, $N_I$ of concept names, role names, individuals, and variables respectively. For more details of syntax and semantics of $\mathcal{ELH}_\perp$ ontologies, readers can refer to (Baader, Brandt, and Lutz 2005). A conjunctive query $Q(\vec{x})$ is of the form $\exists \vec{y}. \phi(\vec{x}, \vec{y})$, where $\phi$ is a conjunction of atoms with predicates and terms from $N_C \cup N_R$ and $N_I \cup N_V$, respectively. A boolean conjunctive query (BCQ) is a CQ that answers variable $\vec{x}$ are empty or being instantiated by constants $\vec{a}$ of the same arity. We formally introduce the query abduction problem (Calvanese et al. 2013) as follows.

http://xsb.sourceforge.net/
Proposition 1 Let $K$ be an $\mathcal{ELH}_\bot$ KB, $Q(\bar{a})$ be a BCQ, $\Sigma \subseteq N_C \cup N_R$, and $\Delta \subseteq N_I$. Then, $\text{sol}(K, Q(\bar{a}), \Sigma, \Delta) \subseteq \text{sol}(D_K, Q(\bar{a}), \Sigma, \Delta)$.

However, the solutions of QAP $(D_K, Q(\bar{a}), \Sigma, \Delta)$ are not necessarily sound. The following example illustrates it.

Example 1 Let $K$ consist of TBox $\{A \sqsubseteq \exists R.A\}$, and an empty ABox. Then, the corresponding datalog program $D_T$ contains the following datalog rules $\text{A}(x) \rightarrow \text{r}(x, c_1)$ and $\text{A}(x) \rightarrow \text{A}(c_1)$. Let $\Sigma$ and $\Delta$ be $\{A\}$ and $\{a, b\}$ respectively. For the queries $Q_1(x, y) = \exists z. [r(x, z) \land r(y, z)]$ and $Q_2(x) = \exists y. [r(x, y) \land r(y, y)]$, then $\text{sol}(D_K, Q_1(a, b), \Sigma, \Delta) = \{A(a), A(b)\}$ and $\text{sol}(D_K, Q_2(a), \Sigma, \Delta) = \{A(a)\}$. However, both QAPs $(K, Q_1(a, b), \Sigma, \Delta)$ and $(K, Q_2(a), \Sigma, \Delta)$ have no solution.

To retain the soundness of our ontology transformation, we propose to rewrite observations in a QAP. Our rewriting approach is adapted from (Lutz, Toman, and Wolter 2009) but it is simpler since our datalog approximation is tighter. Intuitively, the unsoundness problem is caused by fork-shaped or cyclic structures (e.g. respectively $Q_1$ and $Q_2$ in Example 1) in the approximated model of $D_K$ but not necessarily in models of $K$. For a CQ, such substructures can be identified in time polynomial to the size of $Q$. Then rewriting for CQ $Q(\bar{x}) = \exists y. [\phi(x, \bar{y})]$ is $Q'(\bar{x}) = \exists y. [\phi \land \phi_1 \land \phi_2]$. Intuitively, filter $\phi_1$ says that fork-shaped substructures in $Q$ should be instantiated in a way that all the legs merge into one, and $\phi_2$ says that a cyclic substructure in $Q$ can not be instantiated with fresh constants introduced in approximation phase. Now, we present a major result in the paper that shows the new rewriting is sufficient to have a sound and complete query abduction system.

Theorem 1 Let $K$ be an $\mathcal{ELH}_\bot$ KB, $Q(\bar{a})$ be a BCQ, $\Sigma \subseteq N_C \cup N_R$, and $\Delta \subseteq N_I$. Then $\text{sol}(K, Q(\bar{a}), \Sigma, \Delta) = \text{sol}(D_K, Q'(\bar{a}), \Sigma, \Delta)$.

Like many other procedures of computing abductive solutions, a resolution-based algorithm is needed for our transformation-based procedure. We implemented our new procedure of computing QAP solutions using the highly optimized Prolog engine XSB. To compute (minimal) solutions to a QAP $(D_K, Q'(\bar{a}), \Sigma, \Delta)$, we encode datalog program $D_K$ and query $Q'(\bar{a})$ into Prolog rules, and use the list structure in Prolog to store the solutions generated during the resolution, and we have also employed the database feature of XSB to allow the initial ABox to be stored in a database, which significantly improves the efficiency.

Experimental Results

We have implemented a prototype QAP solver ABEL (Abduction for EL) for CQs. We have evaluated our system on the LSTW($n$), an extended version of the university ontology LUBM, which has more existential axioms. We used a query generator to generate both atomic and conjunctive queries, respectively 13 and 27 queries were generated. The abducibles are all the concept names in the TBox, and the domain consists all the individuals in the ABox. The sizes of ABoxes ranges from 100 thousands assertions (LSTW(11)) to 10 millions assertions (LSTW(100)). Overall, we ran the first experiment for 200 test cases. Table 1 shows the average and maximum execution times as well as the success rate. All times are in seconds. For difficult CQs, ABEL could not return any answer for the LSTW(50) and LSTW(100) ontologies in 1 hour time limit, because of that the average and maximum computation times are dramatically decreased.

<table>
<thead>
<tr>
<th>Ontologies</th>
<th>Atomic queries</th>
<th>Conjunctive queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTW1</td>
<td>100% 2.4 10.4</td>
<td>56% 2.3 28.3</td>
</tr>
<tr>
<td>LSTW10</td>
<td>100% 25.6 134.7</td>
<td>56% 27.4 390.7</td>
</tr>
<tr>
<td>LSTW50</td>
<td>100% 139.8 720.7</td>
<td>52% 6.2 12.4</td>
</tr>
<tr>
<td>LSTW100</td>
<td>100% 488.0 2589.0</td>
<td>52% 20.9 43.5</td>
</tr>
</tbody>
</table>

Table 1: Evaluate ABEL for CQs.

Since the query generator could not generate fork-shaped or cyclic CQs, we conducted another set of experiments to evaluate the performance of ABEL as shown in Table 2 over the manually crafted queries. The Q1 is a cyclic CQ and the rest is fork-shaped. OM means that XSB ran out of memory.

Table 2: Evaluate ABEL for fork-shaped and cyclic CQs.

<table>
<thead>
<tr>
<th>Query</th>
<th>LSTW1</th>
<th>LSTW10</th>
<th>LSTW50</th>
<th>LSTW100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>Q2</td>
<td>8.6</td>
<td>74.1</td>
<td>419.2</td>
<td>1564.5</td>
</tr>
<tr>
<td>Q3</td>
<td>0.3</td>
<td>0.2</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Q4</td>
<td>1440.5</td>
<td>OM</td>
<td>OM</td>
<td>OM</td>
</tr>
</tbody>
</table>

Table 2: Evaluate ABEL for fork-shaped and cyclic CQs.

Despite the complexity of the query abduction problem with general CQs, our prototype system can efficiently handle most of the CQs over large datasets (10 millions assertions), which shows that our approach can provide efficient query explanation services in real-life applications with realistic ontologies and large ABoxes.

References


Stefanoni, G.; Motik, B.; and Horrocks, I. 2013. Introducing nominals to the combined query answering approaches for EL. In Proc. of the 27th AAAI.