Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

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

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

Introduction

With the current upward trend in semantically annotated data, ontology-based data access (OBDA) was formulated to tackle the problem of data integration and query answering, where an ontology is formalized as a description logic (DL) TBox. In order to meet usability requirements set by users, efforts have been made to equip OBDA system with explanation facilities. One important explanation tool for DL ontologies, referred to as query abduction, can be formalised as *abductive reasoning*. In particular, given an ontology and an observation (i.e., a query with an answer that called boolean query), 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 in the literature. One major stream of such algorithms are the tableau-based ones, which are however not efficient enough to deal with realistic ontologies and large data sets. Query abduction algorithms could rarely scale over relative large data sets until the work of Du et al.(2011; 2014). The algorithm of Du et al. (2011) supports conjunctive queries (CQs) without existentially quantified variables, but it is not guaranteed to compute all explanations for general CQs. In a more recent work (2014), the authors suggested representative explanations to reduce the number of computed explanations. However, the approach applies only to a special class of ontologies, name first-order(FO) rewritable ones, which does not include all $\mathcal{ELH}_\bot$ ontologies.

In this paper, we develop a practical algorithm of query abduction for general CQs in $\mathcal{ELH}_\bot$ ontologies. This is achieved through ontology approximation and query rewriting. First, we approximate a given $\mathcal{ELH}_\bot$ ontology into a plain datalog program which is complete but not necessarily sound. We adapt a query rewriting technique to rewrite the observation so as to filter out those incorrect explanations. As a result, our approach is sound and complete for query abduction with general CQs as observations. We implemented a prototypical system using the highly optimized Prolog engine XSB[1]. We evaluated our algorithm over some realistic ontologies with reasonably large data sets such as extended university benchmark 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.

[1]http://xsb.sourceforge.net/

Query Abduction Problem

We consider countably infinite and mutually disjoint sets N_C, N_R, N_I and N_V of concept names, role names, individuals, and variables respectively. For details of $\mathcal{ELH}_\bot$ readers can refer to (Baader, Brandt, and Lutz 2005). A conjunctive query $\mathcal{Q}(\vec{x})$ is of the form $\exists\vec{y}.\phi(\vec{x},\vec{y})$, where ϕ 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 answer variables $\vec{x}$ are empty. In Definition 1, we formally introduce a *query abduction* problem. In contrast to the definition in (Calvanese et al. 2013), our definition has a finite domain Δ as an additional attribute, and contains all the constants occurring in a solution in order to avoid infinitely many (minimal) solutions for a QAP.

Definition 1 (QAP) *An instance of a* query abduction problem *(QAP) in $\mathcal{ELH}_\bot$ is a tuple $\langle \mathcal{K}, \mathcal{Q}(\vec{a}), \Sigma, \Delta\rangle$, where $\mathcal{K}$ is an $\mathcal{ELH}_\bot$ KB, $\mathcal{Q}(\vec{a})$ is a BCQ called the* observation. *$\Sigma \subseteq N_C \cup N_R$ is a finite set of predicates called* abducibles, *and $\Delta \subseteq N_I$ is a finite set of constants called the* domain. *A* solution *$\mathcal{E}$ to the QAP is a set of facts over Σ such that (1) $pred(\mathcal{E}) \subseteq \Sigma$, (2) $const(\mathcal{E}) \subseteq \Delta$, (3) $\mathcal{K} \cup \mathcal{E}$ is consistent, and (4) $\mathcal{K} \cup \mathcal{E} \models \mathcal{Q}(\vec{a})$. Let $sol(\mathcal{K}, \mathcal{Q}(\vec{a}), \Sigma, \Delta)$ be the set of solutions to the QAP. Moreover, we call $\mathcal{E}$* set minimal *iff there is no solution $\mathcal{E}' \in sol(\mathcal{K}, \mathcal{Q}(\vec{a}), \Sigma, \Delta)$ s.t. $\mathcal{E}' \subset \mathcal{E}$.*

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

Computing Minimal Solutions

For an $\mathcal{ELH}_\bot$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we will first transform $\mathcal{T}$ into a datalog program $\mathcal{D}_\mathcal{T}$ in a similar way to (Stefanoni, Motik, and Horrocks 2013). The program $\mathcal{D}_\mathcal{K} = \mathcal{D}_\mathcal{T} \cup \mathcal{A}$ 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 $\mathcal{D}_\mathcal{K}$ which formally presents as follows.

Proposition 1 *Let $\mathcal{K}$ be an $\mathcal{ELH}_\bot$ KB, $\mathcal{Q}(\vec{a})$ be a BCQ, $\Sigma \subseteq N_C \cup N_R$, and $\Delta \subseteq N_I$. Then, $sol(\mathcal{K}, \mathcal{Q}(\vec{a}), \Sigma, \Delta) \subseteq sol(\mathcal{D}_\mathcal{K}, \mathcal{Q}(\vec{a}), \Sigma, \Delta)$.*

However, the solutions of QAP $\langle \mathcal{D}_\mathcal{K}, \mathcal{Q}(\vec{a}), \Sigma, \Delta\rangle$ are not necessarily sound. The following example illustrates it.

Example 1 *Let $\mathcal{K}$ consist of TBox $\{A \sqsubseteq \exists r.A\}$, and an empty ABox. Then, the corresponding datalog program $\mathcal{D}_\mathcal{T}$ contains the following datalog rules $A(x)\to r(x,c_1)$ and $A(x)\to A(c_1)$. Let Σ and Δ be $\{A\}$ and $\{a,b\}$ respectively. For the queries $\mathcal{Q}_1(x,y)=\exists z.[r(x,z)\wedge r(y,z)]$ and $\mathcal{Q}_2(x)=\exists y.[r(x,y)\wedge r(y,y)]$, then $\{A(a), A(b)\} \in sol($ $\mathcal{D}_\mathcal{K}$,$\mathcal{Q}_1(a,b)$,Σ,$\Delta)$ and $\{A(a)\} \in sol(\mathcal{D}_\mathcal{K}$,$\mathcal{Q}_2(a)$, Σ, $\Delta)$. However, neither $\langle\mathcal{K}$,$\mathcal{Q}_1(a,b)$, Σ, $\Delta\rangle$ nor $\langle\mathcal{K}$, $\mathcal{Q}_2(a)$, Σ, $\Delta\rangle$ has a solution.*

To retain the soundness of our ontology transformation, we propose to rewrite observations in a QAP to filter out incorrect solutions. 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 $\mathcal{Q}_1$ and $\mathcal{Q}_2$ in Example 1) in the models of approximation $\mathcal{D}_\mathcal{K}$ but not necessarily in all models of $\mathcal{K}$. For a CQ $\mathcal{Q}$, such substructures can be identified in time polynomial to the size of $\mathcal{Q}$ (2009).

Then *rewriting* for CQ $\mathcal{Q}(\vec{x}) = \exists\vec{y}.\phi(\vec{x},\vec{y})$ is $\mathcal{Q}^*(\vec{x}) = \exists\vec{y}.[\phi \wedge \phi_1 \wedge \phi_2]$. Intuitively, filter ϕ_1 says that fork-shaped substructures in $\mathcal{Q}$ should be instantiated in a way that all the legs merge into one, and ϕ_2 says that a cyclic substructure in $\mathcal{Q}$ 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 for the correctness of query abduction over the datalog approximation.

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

Like many other procedures of computing abductive solutions, a resolution-based algorithm is needed for our procedure. We implemented our new procedure of computing QAP solutions using the highly optimized Prolog engine XSB. To compute all minimal solutions to a QAP $\langle\mathcal{D}_\mathcal{K}, \mathcal{Q}^*(\vec{a}), \Sigma, \Delta\rangle$, we encode the QAP into Prolog rules, and use the list structure to store the solutions generated during the resolution. 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.

Ontologies	Atomic queries			
	#	Succ.	Avg.	Max.
LSTW1	13	100%	2.4	10.4
LSTW10		100%	25.6	134.7
LSTW50		100%	139.8	720.7
LSTW100		100%	488.0	2589.0
	Conjunctive queries			
LSTW1	27	56%	2.3	28.3
LSTW10		56%	27.4	390.7
LSTW50		52%	6.2	12.4
LSTW100		52%	20.9	43.5

Table 1: Evaluate ABEL for CQs.

Query	LSTW1	LSTW10	LSTW50	LSTW100
Q1	0.2	0.6	0.2	0.6
Q2	8.6	74.1	419.2	1564.5
Q3	0.3	0.2	0.6	0.6
Q4	1440.5	OM	OM	OM

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

Experimental Results

We have evaluated our system on the LSTW(n), an extended version of the university ontology LUBM, which has axioms with existentially quantified variables on the right-hand sides. We used a query generator to generate 13 atomic and 27 conjunctive queries. The abducibles are all the concept names in the TBox, and the domain consists of all individuals in the ABox. The sizes of ABoxes range from 100 thousands (LSTW(1)) to 10 millions assertions (LSTW(100)), the sizes of domain are between 17 thousands (LSTW(1)) and 1.7 million (LSTW(100)) and the abducible size is 132. Overall, we ran the first experiment for 200 test cases. Table 1 shows the average and maximum execution times as well as success rates. All times are in seconds, and the time limit is one hour. The overall success rates of ABEL for both atomic and conjunctive queries over LSTW(1) is 70%, and for the rest of ontologies is 67%. ABEL failed to return all explanations for general CQs that have atoms from the recursive axiom $\exists advisor.Professor \sqsubseteq Professor$. However, it was successful to compute all explanations for the atomic query $Q(x) \leftarrow Professor(x)$.

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 manually crafted queries. Q1 is a cyclic CQ and the rest is fork-shaped. OM means that XSB ran out of memory.

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

References

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th IJCAI*, 364–369.

Calvanese, D.; Ortiz, M.; Šimkus, M.; and Stefanoni, G. 2013. Reasoning about explanations for negative query answers in dl-lite. *JAIR* 48:635–669.

Du, J.; Qi, G.; Shen, Y.-D.; and Pan, J. Z. 2011. Towards practical abox abduction in large owl dl ontologies. In *Proc. of the 25th AAAI*.

Du, J.; Wang, K.; and Shen, Y.-D. 2014. Tractable approach to abox abduction over description logic ontologies. In *Proc. of the 28th AAAI*.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic el using a relational database system. In *Proc. of the 20th IJCAI*.

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