Towards Polynomial Time Forgetting and Instance Query Rewriting in Ontology Languages

Sen Zheng Renate A.Schmidt

University of Manchester, Manchester, UK
{sen.zheng,renate.schmidt}@manchester.ac.uk

Abstract: Ontologies can present a conceptual view of a relational database. This ontology-based data access (OBDA) can allow a client to query enterprise ontologies directly. The problem of rewriting and optimisation of such queries against ontologies is insufficiently studied in database research. In this paper, we discuss using uniform interpolation to forget some symbols, especially role symbols, to rewrite instance queries against ontologies. In particular, when there is no nesting in an ontology, our forgetting algorithm is guaranteed to terminate in a polynomial time. We introduce Ackermann's lemma-based algorithm to preserve semantic equivalence during query rewriting. We further extend our approach to linear Datalog$^{\pm}$ rules (existential rules with equality) and also the guarded fragment of first-order logic. These two languages can be regarded as generalisations of description logics, which provide bases of ontology languages.

1 Introduction

An ontology is a form of graph-based database management system, and it allows automated processing and reasoning. In the ontology-based data access (OBDA), an ontology is used as a conceptual layer of relational databases, allowing clients to manage and query data more directly. Such querying is called the ontology-based query answering (OBQA). It is now an insufficiently studied problem in database research.

In OBDA, an instance query q is answered against a database D with an ontology Σ such that $D \cup \Sigma \models q$. An instance query is a unary atomic query such as A(x). In this setting, Description Logics (DLs) are used as an ontology language. However, the best known decidable fragments of DLs are 2EXPTIME-complete, which makes querying very hard. Most OBQA systems are based on a lightweight DLs, such as DL-Lite [5] and $\mathcal{EL}$ [2] families. These DLs are designed to guarantee decidability and polynomial time data complexity for the query answering. DLs only allow unary and binary relations, Calì [3] argues that the Datalog$^{\pm}$ language, which have multi-ary and unary relations, is a strong tool for query answering. In Datalog$^{\pm}$, guarded Datalog$^{\pm}$ and its subclass linear Datalog$^{\pm}$ show good decidability results [4]. Having lightweight DLs and linear Datalog$^{\pm}$ ontologies, researchers focus on rewriting and optimising queries to make querying more effective. In [5], authors proposed the *perfect reformulation* algorithm to do rewriting. [7] gives a polynomial rewriting approach for linear Datalog$^{\pm}$.

Since DLs $\mathcal{ALCOI}$ ($\mathcal{ALC}$ with nominals and inverse roles) can be seen as a fragment of first-order logic, its translation in first-order logic is generalised as the Guarded Fragment (GF) [1]. The guards in GF correspond to role symbols in DLs. Moreover, GF is also a superclass of the linear Datalog$^{\pm}$ that follows GF format.

In this paper, we use a *forgetting algorithm* to rewrite queries while preserving semantic equivalence, and we are interested in a new different class from previous ones, GF without nested formulas, to forget guard symbols. In particular, when guards do not occur in non-guard positions, the data complexity and combined complexity of our algorithm is tractable.

2 Forgetting and GF

Forgetting is a non-standard reasoning procedure to remove the forgetting signatures from original formulas, and keep the remaining formulas semantically equivalent to the original formulas. In other words, the result formulas are equivalent to the original formulas up to the forgetting signatures. This work is motivated by [11] and [10] that extends the forgetting algorithm to first-order logic.

GF is robustly decidable [8], but it does not have the Craig Interpolation Property, thus the Uniform Interpolation property (a.k.a the forgetting property) [9]. That means forgetting some predicates in GF does not guarantee that the result still belongs to GF. Recent research use the model theory to show that the forgetting signature can only be non-guard predicates and it fails to forget guards.

In this paper, we show that by introducing $\exists$-guard, guards can be forgotten without losing semantic equivalence.

3 Forgetting Guards in GF

We define a guarded formula without any nested formulas as a flat guarded formula. In particular, for flat guarded formulas with equality and constants, we concern forgetting guards when there is no guard occurring at a non-guard position in other formulas. The input is a set of formulas N combined by formulas mentioned above, and the forgetting signatures are a set of guards G in N. Our algorithm has 4 major steps:

1. **Normalisation** In this step, every input formula in N is formalised as a formula without any free variables.

We add universal quantifications to free variables in N, and then transform these formulas into their negation normal forms N_1.

2. **Structural Transformation** In this step, each formula in N_1 is transformed into its clausal form. We introduce new predicates, also known as definers, to do structural transformation. During structural transformation, each formula in N_1 is transformed differently depending on the root of its formula tree. For some formulas in N_1 that contain constants and equalities, we use the term abstraction rule and the equality elimination rule as follows.

$$\frac{N \cup \{C(\bar{x}, \bar{a})\}}{N \cup \{C(\bar{x}, \bar{y}) ... \vee y_n \not\approx a_n\}}$$

where $\bar{y}$ is disjoint with $\bar{x}$.

$$\frac{C \vee x \not\approx a}{C \vee Q_i(x), \neg Q_i(a)}$$

where Q_i is a fresh predicate.

We also introduce some special definers $\approx$-*guard* eqG and $\exists$-*guard* eG. An eqG is used to define equalities such as $x \not\approx a$, and an eG is used as a guard for existential quantified unguarded clause such as $\exists xy(A(x) \wedge B(y))$. An eG is used to transform it into $\exists xy(eG(x, y) \wedge A(x) \wedge B(y))$. After structural transformation, the clausal form of formulas in N_1 is ground or is positive conjunction of atoms or contains a negative literal that has all variables in this clause. We call the set of result clauses N_2.

3. **Forgetting Guards** In this step, the set of guard symbols G in the forgetting signatures are eliminated. We use Ackermann's Lemma to eliminate guards one at a time. The set of result formulas are called N_3.

4. **Eliminating Definers** In this step, the aim is to eliminate definer symbols introduced in step 2. Ackermann's Lemma is also used to eliminate these definers.

Given flat guarded formulas, possibly with equality and constants, and assuming there is no guard occurring at a non-guard position in other formulas. We can claim that:

Claim 3.1 *This forgetting algorithm is sound, terminating and forgetting complete.*

Claim 3.2 *The complexity of our algorithm is polynomial.*

4 Conclusion and Ongoing Work

In this paper, we show that we can forget guards in flat GF in a polynomial time without losing semantic equivalence. Because we consider instance queries in this paper, this approach can be used as a query rewriting algorithm to forget guards in Datalog$^{\pm}$ that follows GF format, and to forget roles in $\mathcal{ALCOI}$. In future, we will focus on forgetting a non-guard predicates in GF and apply our algorithm to other possible applications like abduction reasoning [6].

References

[1] Hajnal Andréka, István Németi, and Johan van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[2] Franz Baader. Terminological cycles in a description logic with existential restrictions. In *IJCAI*, volume 3, pages 325–330, 2003.

[3] Andrea Calì. Ontology querying: datalog strikes back. In *Reasoning Web International Summer School*, pages 64–67. Springer, 2017.

[4] Andrea Calı, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res*, 48:115–174, 2013.

[5] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated reasoning*, 39(3):385–429, 2007.

[6] Diego Calvanese, Magdalena Ortiz, Mantas Simkus, and Giorgio Stefanoni. Reasoning about explanations for negative query answers in dl-lite. *Journal of Artificial Intelligence Research*, 48:635–669, 2013.

[7] Georg Gottlob, Marco Manna, and Andreas Pieris. Polynomial rewritings for linear existential rules. In *IJCAI*, pages 2992–2998, 2015.

[8] Erich Grädel. Why are modal logics so robustly decidable? In *Bulletin EATCS*. Citeseer, 1999.

[9] Eva Hoogland and Maarten Marx. Interpolation and definability in guarded fragments. *Studia Logica*, 70(3):373–409, 2002.

[10] Patrick Koopmann. Practical uniform interpolation for expressive description logics. 2015.

[11] Yizheng Zhao and Renate A Schmidt. Role forgetting for alcoqh (δ)-ontologies using an ackermann-based approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1354–1361. AAAI Press, 2017.