Polynomial Datalog Rewritings for Ontology Mediated Queries with Closed Predicates*
(Extended Abstract)

Shqiponja Ahmetaj, Magdalena Ortiz, and Mantas Šimkus

Institute of Information Systems, TU Wien, Austria

In *ontology-mediated queries* (OMQs), a database query is enriched with an ontology, providing knowledge to obtain more complete answers from incomplete data. OMQs are the focus of intensive research, particularly when the ontology is expressed in Description Logics (DLs) or in rule-based formalisms like *existential rules* and DATALOG±, see e.g., [5, 4, 10] and their references. The *open-world semantics* of these formalisms makes them suitable for handling incomplete knowledge, but viewing *all* data as incomplete can result in too few *certain answers*. For this reason, *closed predicates* have been advocated as a powerful tool to combine complete and incomplete knowledge, by explicitly specifying predicates assumed complete, thus given a *closed-world* semantics [8, 13]. For example, take the following self-explanatory ontology $\mathcal{T}$ (formally, a DL *TBox*):

$$\mathsf{BScStud} \sqsubseteq \mathsf{Student}, \mathsf{Student} \sqsubseteq \exists \mathsf{attends}.\mathsf{Course}, \mathsf{BScStud} \sqsubseteq \forall \mathsf{attends}.\neg \mathsf{GradCourse}$$

and the following set of facts $\mathcal{A}$ (an *ABox*):

$$\mathsf{Course}(c_1),\ \mathsf{Course}(c_2),\ \mathsf{GradCourse}(c_2),\ \mathsf{BScStud}(a)$$

The *instance query* $q(x, y) = \mathsf{attends}(x, y)$ mediated by $\mathcal{T}$ does not retrieve (a, c_1) as a *certain answer*, but if c_1 and c_2 are known to be the *only* courses, then we can declare Course a *closed predicate*, and then (a, c_1) becomes a certain answer.

We investigate the relative expressiveness of OMQs in terms of more traditional query languages like DATALOG. More precisely, we are interested in the following problem: given an OMQ Q (specified by a query and a TBox, possibly with closed predicates), obtain a DATALOG query Q'—in a suitable fragment—such that, for any ABox $\mathcal{A}$, the certain answers to Q and Q' coincide. The existence of such a Q' and its size are crucial for understanding the expressive power and succinctness of different families of OMQs. It is also very relevant in practice, since it allows to reuse existing database technologies to support OMQ answering. For example, the research into OMQs that can be rewritten into *first-order (FO) queries* has produced the successful *DL-Lite* family [6], which has been extensively studied, and laid the base for developing other FO-rewritable query languages, e.g., [9, 11]. Many DLs are not FO-rewritable, but can be rewritten into monotonic DATALOG queries, leading to implemented systems, e.g., [17, 7, 19]. The pioneering work in [12] showed that instance queries in an expressive extension of $\mathcal{ALC}$ (without closed predicates) can be rewritten into a disjunctive DATALOG program, using a constant number of variables per rule, but exponentially many rules. For *union*

* This work was supported by the Austrian Science Fund (FWF) projects P25207, T515 and W1255.

of conjunctive queries in $\mathcal{ALC}$, the existence of exponential DATALOG rewritings was shown recently [5]. A polynomial DATALOG translation of instance queries was proposed in [16], but for a so-called *Horn-DL* that lacks disjunction; to our knowledge, this was the only polynomial rewriting for a DL that is not FO-rewritable until now. In the presence of closed predicates, the only rewritability results are FO-rewritability for the core fragment of DL-Lite [14], and a rewriting algorithm for queries that satisfy some strong *definability* criteria [18]. Other works on OMQs with closed predicate have focused on the complexity of their evaluation, e.g., [15, 13, 8]; since answering these queries is coNP-hard in data complexity for most lightweight DLs, the existence of FO-rewritings is ruled out.

We consider OMQs of the form $(\mathcal{T}, \Sigma, q)$, where q is an instance query and $\mathcal{T}$ is a TBox in the very expressive DL $\mathcal{ALCHIO}$ with closed predicates Σ. Observe that these queries are *non-monotonic*: if $\Sigma = \{\mathsf{Course}\}$ are the closed predicates in the above example, then (a, c_1) is a certain answer to $(\mathcal{T}, \Sigma, q)$ over $\mathcal{A}$, but it is not a certain answer over $\mathcal{A}' = \mathcal{A} \cup \{\mathsf{Course}(c_3)\}$. This shows that these queries cannot be rewritten into monotonic variants of DATALOG, like positive DATALOG (with or without disjunction). The main contribution of this paper is a polynomial time translation of queries in $\mathcal{Q}$ into *disjunctive* DATALOG *extended with negation as failure* DATALOG$^{\vee\neg}$. Our translation is modular: if no closed predicates are present—i.e., for regular instance queries in $\mathcal{ALCHIO}$—our translation yields a positive disjunctive DATALOG program of polynomial size. To our knowledge, this is the first polynomial time translation of an expressive (non-Horn) DL into DATALOG. The full version of this abstract can be found in [1] and a simplified version of the translation for $\mathcal{ALCHI}$ can be found in [2].

1 OMQs with Closed Predicates

We assume familiarity with DLs [3]. We use p for *role names* in the set $\mathsf{N_R}$, $A_{(i)}$ for *concept names* in $\mathsf{N_C}$, and a, b for *individuals* in $\mathsf{N_I}$. *$\mathcal{ALCHIO}$ knowledge bases (KBs) with closed predicates* take the form $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$, where $\Sigma \subseteq \mathsf{N_R} \cup \mathsf{N_C}$ are the *closed predicates*, $\mathcal{T}$ is a *(normalized) TBox* with axioms of the forms:

(**N1**) $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B_{n+1} \sqcup \cdots \sqcup B_k$

(**N2**) $A_1 \sqsubseteq \exists r.A_2$ (**N3**) $A_1 \sqsubseteq \forall r.A_2$ (**N4**) $r \sqsubseteq s$

where $B_{(i)}$ are concept names or *nominals* $\{a\}$, and r and s are *roles* of the form p or p^-; the ABox $\mathcal{A}$ is a set of assertions of the forms $A(a)$ and $p(a, b)$. Models of axioms, assertions, TBoxes and ABoxes are defined as usual. For an ABox $\mathcal{A}$ and a set of closed predicates Σ, we write $\mathcal{I} \models_\Sigma \mathcal{A}$ if: *(a)* $\mathcal{I} \models \mathcal{A}$, *(b)* for all concept names $A \in \Sigma$, if $e \in A^\mathcal{I}$, then $A(e) \in \mathcal{A}$, and *(c)* for all role names $r \in \Sigma$, if $(e_1, e_2) \in r^\mathcal{I}$, then $r(e_1, e_2) \in \mathcal{A}$. For a KB $\mathcal{K}$, we write $\mathcal{I} \models \mathcal{K}$ if the following hold:[1] *(i)* $a \in \Delta^\mathcal{I}$ and $a^\mathcal{I} = a$ for each $a \in \mathsf{N_I}$ occurring in $\mathcal{K}$, *(ii)* $\mathcal{I} \models \mathcal{T}$, and *(iii)* $\mathcal{I} \models_\Sigma \mathcal{A}$. For an assertion α, we write $\mathcal{K} \models \alpha$ if $\mathcal{I} \models \alpha$ for all $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$. Finally, an *(ontology mediated) instance query* is a triple $Q = (\mathcal{T}, \Sigma, q)$, where $q \in \mathsf{N_C} \cup \mathsf{N_R}$. Let $\boldsymbol{a} \in \mathsf{N_I}$ in case $q \in \mathsf{N_C}$, and $\boldsymbol{a} \in \mathsf{N_I}^2$ otherwise. Then $\boldsymbol{a}$ is a *certain answer* to Q over an ABox $\mathcal{A}$ if $(\mathcal{T}, \Sigma, \mathcal{A}) \models q(\boldsymbol{a})$; note that if $\Sigma = \emptyset$, this boils down to the usual DL instance checking problem.

[1] We make the *standard name assumption (SNA)*for the individuals occurring in $\mathcal{K}$.

2 Rewriting OMQs into Datalog$^{\vee\neg}$

Assume a KB $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$ and an assertion q. Deciding $\mathcal{K} \not\models q$ amounts to deciding whether there exists a *counterexample (for the entailment of q from $\mathcal{K}$)*, which is an $\mathcal{I}$ with $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \not\models q$. In this section we give a brief explanation of how this can be decided, and reduced to evaluating a DATALOG$^{\vee\neg}$ query.

Below, a *core interpretation for* $\mathcal{K}$ is an interpretation $\mathcal{I}$ with domain $\mathsf{N_I}(\mathcal{K})$ such that $\mathcal{I} \models_\Sigma \mathcal{A}$, and which satisfies some additional conditions (e.g., it models the TBox axioms of the forms (**N1**), (**N3**), (**N4**), and also (**N2**) when the role is closed); see [1]. Intuitively, core interpretations fix how the individuals participate in concepts and roles, and models of $\mathcal{K}$ can be seen as core interpretations extended by adding *anonymous objects* to satisfy all TBox axioms.

To decide the existence of a counterexample, we proceed in two steps:

(1) Guess a *core interpretation* $\mathcal{I}$ for $\mathcal{K}$, such that $\mathcal{I} \not\models q$.
(2) Check that $\mathcal{I}$ can be *extended* to satisfy all axioms in $\mathcal{T}$. Since the extension coincides with $\mathcal{I}$ on the assertions they entail, this preserves the non-entailment of q.

Given $\mathcal{T}$, Σ and q, defining DATALOG$^{\vee\neg}$ rules that do (1) for any input $\mathcal{A}$ is not so hard. For example, rules like the following 'guess' how individuals participate in concepts and roles: $A(x) \vee \overline{A}(x) \leftarrow \mathsf{ind}(x)$ $\quad\quad$ $\leftarrow A(x), \overline{A}(x)$

$r(x,y) \vee \overline{r}(x,y) \leftarrow \mathsf{ind}(x), \mathsf{ind}(y)$ $\quad$ $\leftarrow r(x,y), \overline{r}(x,y)$

and other rules then verify the additional conditions in the definition of core interpretation. The latter simulate the TBox axioms in a rather direct way (e.g., an axiom $r \sqsubseteq s$ becomes $s(x,y) \leftarrow r(x,y)$). Their only interesting feature is that, to ensure that no instances are added to closed predicates in the extension of $\mathcal{I}$, we use constraints with negative body atoms. For example, we use $\leftarrow r(x,y), \mathit{not}\ s(x,y)$ instead of the rule above if s is closed.

Now, step (2) is harder: given $\mathcal{T}$, Σ, and $\mathcal{I}$, verifying whether $\mathcal{I}$ can be extended into a full model of $\mathcal{T}$ while respecting Σ is EXPTIME-hard already for fragments of $\mathcal{ALCHOI}$ (as it is a generalization of *consistency testing*). In order to obtain a polynomial set of rules that solves this EXPTIME-hard problem, we characterize it as a game, revealing a simple algorithm that admits an elegant implementation in DATALOG$^{\vee\neg}$.

The game is played over a set $\mathsf{LC}(\mathcal{T}, \Sigma, \mathcal{I})$ of *locally consistent types*, which are sets of atomic concepts satisfying conditions such as having no explicit inconsistencies and satisfying all axioms of type (**N1**) in $\mathcal{T}$. Additionally, a type τ that contains a nominal $\{a\}$ must be *the type realized by a in $\mathcal{A}$*, that is, τ is *exactly* the set of all B such that $a \in B^{\mathcal{I}}$. Moreover, τ must be realized in $\mathcal{I}$ by some individual whenever it contains a closed concept, or a concept occurring on the left-hand-side of an axiom (**N2**) that has closed role on the right.

We now describe the game, which is played by Bob (the builder), who wants to extend $\mathcal{I}$ into a model, and Sam (the spoiler), who wants to spoil all Bob's attempts. The game on $\mathcal{I}$ starts by Sam choosing an individual $a \in \Delta^{\mathcal{I}}$, and $\tau = \mathit{type}(a, \mathcal{I})$ is set to be the *current type*. Then:

(♦) If $\tau \notin \mathsf{LC}(\mathcal{T}, \Sigma, \mathcal{I})$, then Sam is declared winner. Otherwise, Sam chooses an inclusion $A \sqsubseteq \exists r.A' \in \mathcal{T}$ with $A \in \tau$; if there is no such inclusion, Bob wins the game. Otherwise, Bob chooses a new type τ' such that:

(C1) $A' \in \tau'$, and

(C2) for all inclusions $A_1 \sqsubseteq \forall s.A_2 \in \mathcal{T}$:
- if $r \sqsubseteq s \in \mathcal{T}$ and $A_1 \in \tau$ then $A_2 \in \tau'$,
- if $r^- \sqsubseteq s \in \mathcal{T}$ and $A_1 \in \tau'$ then $A_2 \in \tau$.

τ' is set as current type is and we go back to ♦ to continue with a new round.

It can be proved that a core $\mathcal{I}$ can be extended into a model iff Bob has a non-losing strategy for the game played on $\mathcal{I}$. To decide the existence of a non-losing strategy, we implement in DATALOG$^{\vee\neg}$ a procedure to *mark* all types from which there is no non-winning strategy. The rules that do this can be found in the full version. Here we discuss a few sample rules. For example, a rule

$$\mathsf{Marked}(\boldsymbol{x}) \leftarrow \mathsf{Type}(\boldsymbol{x}), B_1 \in \boldsymbol{x}, \ldots, B_n \in \boldsymbol{x}, B'_1 \notin \boldsymbol{x}, \ldots, B'_n \notin \boldsymbol{x}$$

marks types that violate axioms of type (**N1**), where Type is a k-ary relation that contains (bit vectors denoting) the different combinations of the k concepts occurring in $\mathcal{T}$, and $B \in \boldsymbol{x}$ (resp. $B \notin \boldsymbol{x}$) is shortcut for the atom testing if B is (not) in this type. For testing the local consistency conditions that involve *realized types*, we use a k-ary predicate RealizedType that gathers all types realized in $\mathcal{I}$. Then, for example, a rule $\mathsf{Marked}(\boldsymbol{x}) \leftarrow \mathsf{Type}(\boldsymbol{x}), A \in \boldsymbol{x}, \mathit{not}\ \mathsf{RealizedType}(\boldsymbol{x})$ marks all non-realized types that contain a closed A, and similar rules handled enforces s-neighbors with closed s and nominals $\{a\}$. The most interesting part is to go beyond the local consistency, propagating the markings to types that don't allow Bob to pick an unmarked successor type. We need to mark a type τ if $A \in \tau$ for some $\alpha = A \sqsubseteq \exists r.A' \in \mathcal{T}$, and *each* type τ' has either been marked, or violates one of (C1) and (C2). We use an auxiliary $(2k+1)$ relation MarkedOne to collect all such τ': $\mathsf{MarkedOne}(\boldsymbol{x}, a_\alpha, \boldsymbol{y}) \leftarrow \mathsf{Type}(\boldsymbol{x}), \mathsf{Marked}(\boldsymbol{y})$ and similar rules for collecting types that violate (C1) or (C2). We now want to ensure that $\mathsf{Marked}(\boldsymbol{t})$ in case $\mathsf{MarkedOne}(\boldsymbol{t}, a_\alpha, \boldsymbol{v})$ is true for all types $\boldsymbol{v}$. This needs a set of rules that generate a *linear order* over types, and use it to iterate over all types. If we manage to reach the last type, the current type is marked. To this end, we need another $(2k+1)$-ary relation MarkedUntil. We add:

$$\mathsf{MarkedUntil}(\boldsymbol{x}, a_\alpha, \boldsymbol{z}) \leftarrow \mathsf{MarkedOne}(\boldsymbol{x}, a_\alpha, \boldsymbol{z}), \mathsf{first}(\boldsymbol{z})$$
$$\mathsf{MarkedUntil}(\boldsymbol{x}, a_\alpha, \boldsymbol{u}) \leftarrow \mathsf{MarkedUntil}(\boldsymbol{x}, a_\alpha, \boldsymbol{z}), \mathsf{next}(\boldsymbol{z}, \boldsymbol{u}), \mathsf{MarkedOne}(\boldsymbol{x}, a_\alpha, \boldsymbol{u})$$
$$\mathsf{Marked}(\boldsymbol{x}) \leftarrow \mathsf{MarkedUntil}(\boldsymbol{x}, a_\alpha, \boldsymbol{z}), A \in \boldsymbol{x}, \mathsf{last}(\boldsymbol{z})$$

Finally, there is a set of rules that check that all types realized in the core are not marked. Putting all the rules together, we can show the following:

Theorem 1. *For an instance query $(\mathcal{T}, \Sigma, q)$, where $\mathcal{T}$ is an $\mathcal{ALCHIO}$ TBox, we can build in polynomial time a* DATALOG$^{\vee\neg}$ *program P such that:*

(i) The certain answers to $(\mathcal{T}, \Sigma, q)$ and (P, q) coincide for any given ABox $\mathcal{A}$ over the signature of $\mathcal{T}$.

(ii) If $\Sigma = \emptyset$, then P is a positive program.

(iii) If $\Sigma = \emptyset$ and $\mathcal{T}$ is an $\mathcal{ALCHI}$ TBox, then P is a positive program with no occurrences of the $\neq$ predicate.

References

1. S. Ahmetaj, M. Ortiz, and M. Simkus. Polynomial datalog rewritings for expressive description logics with closed predicates. In *Proc. of IJCAI 2016*. AAAI Press, 2016.
2. S. Ahmetaj, M. Ortiz, and M. Simkus. Polynomial disjunctive datalog rewritings of instance queries in expressive description logics. In *Proc. of DL 2016*. CEUR-WS.org, 2016.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.
4. M. Bienvenu and M. Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web*, volume 9203 of *Lecture Notes in Computer Science*, pages 218–307. Springer, 2015.
5. M. Bienvenu, B. ten Cate, C. Lutz, and F. Wolter. Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
7. T. Eiter, M. Ortiz, M. Šimkus, T. Tran, and G. Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of AAAI 2012*. AAAI Press, 2012.
8. E. Franconi, Y. A. Ibáñez-García, and I. Seylan. Query answering with DBoxes is hard. *Electr. Notes Theor. Comput. Sci.*, 278:71–84, 2011.
9. G. Gottlob, S. Kikot, R. Kontchakov, V. V. Podolskii, T. Schwentick, and M. Zakharyaschev. The price of query rewriting in ontology-based data access. *Artif. Intell.*, 213:42–59, 2014.
10. G. Gottlob, M. Manna, and A. Pieris. Polynomial rewritings for linear existential rules. In *Proc. of IJCAI 2015*. AAAI Press, 2015.
11. G. Gottlob and T. Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of KR 2012*. AAAI Press, 2012.
12. U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.
13. C. Lutz, I. Seylan, and F. Wolter. Ontology-based data access with closed predicates is inherently intractable(sometimes). In *Proc. of IJCAI 2013*. IJCAI/AAAI, 2013.
14. C. Lutz, I. Seylan, and F. Wolter. Ontology-mediated queries with closed predicates. In *Proc. of IJCAI 2015*. IJCAI/AAAI, 2015.
15. N. Ngo, M. Ortiz, and M. Šimkus. The combined complexity of reasoning with closed predicates in description logics. In *Proc. of DL 2015*. CEUR-WS.org, 2015.
16. M. Ortiz, S. Rudolph, and M. Šimkus. Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In *Proc. of KR 2010*. AAAI Press, 2010.
17. H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. *J. Applied Logic*, 8(2):186–209, 2010.
18. I. Seylan, E. Franconi, and J. de Bruijn. Effective query rewriting with ontologies over DBoxes. In *Proc. of IJCAI 2009*, 2009.
19. D. Trivela, G. Stoilos, A. Chortaras, and G. B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Sem.*, 33:30–49, 2015.