On the Expressive Power of Ontology-Mediated Queries: Capturing coNP

Sanja Lukumbuzya[1], Magdalena Ortiz[2] and Mantas Šimkus[2]

[1]TU Wien, Austria
[2]Umeå University, Sweden

Abstract

The complexity and relative expressiveness of *Ontology-mediated Queries (OMQs)* is quite well understood by now. In this paper, we study the *expressive power* of OMQs from a *descriptive complexity* perspective, where the central question is to understand whether a given OMQ language is powerful enough to express *all* queries that can be computed within some bound on time or space. We show that the OMQ language that pairs instance queries with ontologies in the very expressive DL $\mathcal{ALCHOI}$ with closed predicates cannot express all coNP-computable Boolean queries, despite being coNP-complete in data complexity. We, then, propose an extension of this OMQ language that is expressive enough to precisely capture the class of all Boolean queries computable in coNP. This involves adding functionality as well as path expressions and nominal schemata, which are restricted in a way that allows us to carefully incorporate them into the existing mosaic technique for the DL $\mathcal{ALCHOIF}$ with closed predicates without affecting the coNP upper bound in data complexity.

Keywords

Description Logics, Ontology-mediated Query Answering, Expressive Power, Descriptive Complexity

1. Introduction

Ontology-mediated Queries (OMQs) have received significant attention in the *Description Logic (DL)* community as a powerful tool to answer database-like queries, while taking into account domain knowledge captured in ontologies. The computational complexity of this problem is now well-understood both in terms of *combined complexity* and *data complexity*. Specifically, for the standard OMQs based on expressive DLs of the $\mathcal{ALC}$ family, we have coNP-completeness in data complexity: this is the complexity of checking if a given tuple of individuals belongs to the answer to an OMQ Q over an ABox $\mathcal{A}$, assuming that Q is fixed and only $\mathcal{A}$ varies (see, e.g., [1]). *Relative expressiveness* of OMQs has also been investigated, e.g., via the established translations of OMQs into variants of Datalog (see, e.g., [2, 3, 4]).

The *expressive power* of OMQs from a descriptive complexity [5] perspective has thus far received limited attention. In this context, we ask whether a given OMQ language is powerful enough to express all queries computable within some time or space resources, i.e., belonging to a certain complexity class. It was shown in [2] that there are OMQ languages that capture certain subclasses of coNP related to *constraint satisfaction problems* (CSPs). More recently, a

DL 2023: 36th International Workshop on Description Logics, September 2–4, 2023, Rhodes, Greece
lukumbuzya@kr.tuwien.ac.at (S. Lukumbuzya)

close connection between OMQ languages with the so-called *closed predicates* and *surjective CSPs* was shown in [6]. In this paper, we continue this line of work and we focus on finding an OMQ language that precisely captures the complexity class coNP. It is easy to see that OMQs based on standard expressive DLs and conjunctive queries—while being coNP-complete in data complexity—are not powerful enough to express all queries computable in coNP. This follows directly from the *monotonicity* of standard OMQ languages (e.g., in all cases where the OMQ language is based on first-order logic). Specifically, if Q is a Boolean OMQ in such a language, then for all ABox pairs $\mathcal{A}_1 \subseteq \mathcal{A}_2$ we have that $Q(\mathcal{A}_1) = 1$ implies $Q(\mathcal{A}_2) = 1$. These OMQ languages cannot capture coNP since there exist (rather trivial) *non-monotonic* queries computable in coNP (or even polynomial time). Here is a simple example of such a (non-monotonic) query: "Does the input ABox have an odd number of individuals?"

We, therefore, pose the following question: "Which features make an OMQ language sufficiently expressive to capture the class of all queries computable in coNP, while still maintaining coNP-completeness in data complexity?" The summary of our contributions is as follows:

- We first present an inexpressibility result for *non-monotonic* OMQs based on $\mathcal{ALCHOI}$ with closed predicates. For such expressive OMQs the aforementioned monotonicity-based argument does not apply, and we need a more sophisticated approach. Specifically, by analyzing an existing algorithm in [3] and invoking the *Non-deterministic Time Hierarchy Theorem*, we show that instance queries mediated by $\mathcal{ALCHOI}$ TBoxes with closed predicates are not expressive enough to capture coNP.
- We present an OMQ language that is powerful enough to express all coNP computable queries. As our base DL we choose $\mathcal{ALCHOIF}$ (with closed predicates) and we add to it *nominal schemas* [7] of very restricted shapes. We argue that these additions do not cause an increase in combined complexity and data complexity, i.e. answering ontology-mediated instance queries remains complete for NExpTime and coNP, respectively. This is done by suitably modifying the mosaic-based algorithm in [4].
- We prove that our enriched OMQ language is powerful enough to express all (so-called *generic*) Boolean queries computable in coNP. Each such generic query q is associated to a signature Σ as well as to a set of ABoxes over Σ (also called Σ-ABoxes) in which the answer to the query is "true". By saying that "q is computable in coNP" we mean that there is a *Nondeterministic Turing Machine (NTM)* M_q that recognizes the language of strings representing Σ-ABoxes in which the answer to the query is "false" and runs in polynomial time in the size (of the string representation) of the input ABox. We show that the enriched OMQ language can properly express the computations of M_q. As a consequence of this and the previous point, we obtain a language that captures precisely coNP.

2. Preliminaries

Ontology-Mediated Queries (OMQs) We recall here the necessary notions related to OMQs based on $\mathcal{ALCHOIF}$ with closed predicates.

We use N_C, N_R, and N_I to denote countably infinite, mutually disjoints sets of *concept names*, *role names*, and *individuals*, respectively. A *signature* is any finite set $\Sigma \subseteq \mathrm{N}_C \cup \mathrm{N}_R$. An *assertion* (or, *fact*) is an expression of the form $r(a, b)$ or $A(b)$, where $r \in \mathrm{N}_R$, $A \in \mathrm{N}_C$, and $a, b \in \mathrm{N}_I$.

An *ABox* $\mathcal{A}$ is any finite set of assertions. If all concept and role names that appear in an ABox $\mathcal{A}$ belong to a signature Σ, then $\mathcal{A}$ is a Σ*-ABox*.

We define the set of *roles* N_R^+ as follows: $\mathrm{N}_R^+ = \{p, p^- \mid p \in \mathrm{N}_R\}$. Furthermore, we define the set N_C^+ of *basic concepts* as $\mathrm{N}_C^+ = \mathrm{N}_C \cup \{\{o\} \mid o \in \mathrm{N}_I\} \cup \{\top, \bot\}$. *(Complex) concepts* are defined according to the following syntax $C := A \mid C \sqcap C \mid C \sqcup C \mid \neg C \mid \{o\} \mid \exists r.C \mid \forall r.C$, where $A \in \mathrm{N}_C$, $o \in \mathrm{N}_I$, and $r \in \mathrm{N}_R^+$. We call the concepts of the form $\{o\}$, where $o \in \mathrm{N}_I$, *nominals*. *Axioms* are expressions of the form $C \sqsubseteq D$ (*concept inclusions*), $r \sqsubseteq p$ (*role inclusions*), and $(\mathsf{func}\ r)$ (*functionality assertion*), where C, D are concepts and r, p are roles. A *TBox* is a finite set of axioms. A *knowledge base (with closed predicates)* (KB) is a tuple $(\mathcal{T}, \Sigma, \mathcal{A})$, where $\mathcal{T}$ is a TBox, $\Sigma \subseteq \mathrm{N}_C \cup \mathrm{N}_R$ is a set of *closed predicates* and $\mathcal{A}$ is an ABox. We use $\mathsf{Nom}(\mathcal{K})$ to denote the set $\{\{o\} \mid o \in \mathrm{N}_I,$ and o occurs in $\mathcal{K}\}$. The semantics of TBoxes and ABoxes as given above is defined in the standard way using *interpretations* of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. Additionally, we make the *Standard Name Assumption (SNA)*, which is common when dealing with closed predicates and which forces us to interpret every constant occurring in the KB as itself. We say that $\mathcal{I}$ satisfies a KB $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$, in symbols $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathcal{A}$ and (i) for each $A \in \Sigma$, $A^{\mathcal{I}} = \{a \mid A(a) \in \mathcal{A}, A \in \Sigma \cap \mathrm{N}_C\}$, and (ii) for each $r \in \Sigma$, $r^{\mathcal{I}} = \{(a, b) \mid r(a, b) \in \mathcal{A}, r \in \Sigma \cap \mathrm{N}_R\}$.

In the literature, an OMQ is often given as a pair $(\mathcal{T}, q)$, where $\mathcal{T}$ is a TBox and q is a *Conjunctive Query (CQs)*. In this paper, we do not deal with general CQs: here q is just an *atomic query* or an *inconsistency query* (corresponding to a Boolean CQ $\exists x.\bot(x)$). On the other hand, our OMQs include closed predicates. Thus, an *ontology-mediated atomic query* is a triple $Q = (\mathcal{T}, \Sigma, P)$, where $\mathcal{T}$ is a TBox and $\Sigma \cup \{P\} \subseteq \mathrm{N}_C \cup \mathrm{N}_R$. If P is a concept name, then the *answer* to Q over an ABox $\mathcal{A}$ is the set of all individuals a such that $a^{\mathcal{I}} \in P^{\mathcal{I}}$ for all models $\mathcal{I}$ of $(\mathcal{T}, \Sigma, \mathcal{A})$. The definition of an answer in case P is a role name is analogous. An *inconsistency query* given as a pair $Q = (\mathcal{T}, \Sigma)$. For such Q and any ABox $\mathcal{A}$, we let $Q(\mathcal{A}) = 1$ if $(\mathcal{T}, \Sigma, \mathcal{A})$ has no model, and $Q(\mathcal{A}) = 0$ if $(\mathcal{T}, \Sigma, \mathcal{A})$ has a model.

Turing Machines A *Nondeterministic Turing Machine (NTM)* is a tuple $M = (\Gamma, Q, \delta, q_0, q_{acc}, q_{rej})$, where Γ is an *alphabet*, Q is a set of *states*, $\delta \subseteq (\Gamma \cup \{B\}) \times Q \times (\Gamma \cup \{B\}) \times Q \times \{-1, +1\}$ is a *transition relation*, and $q_0, q_{acc}, q_{rej} \in Q$ are the *initial state*, the *accepting state*, and the *rejecting state*, respectively. The symbol B is the *blank* symbol. An NTM M takes as input a finite word w over Γ, which is written on an infinite tape: the blank symbol B is written in every cell that is not occupied by w. Initially, the read-write head of M is over the first symbol of w and the machine is in state q_0. The computation of M is defined in the usual way. If there is a run of M on w that reaches q_{acc}, then M *accepts* w. The *language* of M is defined as the set of all words over Γ that M accepts.

3. Inexpressbility Result

We present here our inexpressibility results. For this, we first need to formalise *generic queries* over ABoxes, and clarify the notion of membership of such queries in a complexity class.

Definition 1 (Generic Boolean Queries)**.** *For an ABox $\mathcal{A}$, we use* $\mathsf{Adom}(\mathcal{A})$ *to denote the set of individuals that appear in $\mathcal{A}$. We say ABoxes $\mathcal{A}_1, \mathcal{A}_2$ are* isomorphic*, if they are equal up* *to renaming of individuals, i.e. there is a bijection* $f : \mathsf{Adom}(\mathcal{A}_1) \to \mathsf{Adom}(\mathcal{A}_2)$ *such that* $\mathcal{A}_2 = \{A(f(c) \mid A(c) \in \mathcal{A}_1)\} \cup \{r(f(c), f(d)) \mid r(c,d) \in \mathcal{A}_1)\}$.

A Generic Boolean Query (GBQ) Q over a signature Σ *is a function that maps each* Σ*-ABox* $\mathcal{A}$ *to a value* $Q(\mathcal{A}) \in \{0,1\}$*, and is such that* $Q(\mathcal{A}_1) = Q(\mathcal{A}_2)$ *holds for any pair* $\mathcal{A}_1, \mathcal{A}_2$ *of isomorphic* Σ*-ABoxes.*

The assumption that answers GBQs are invariant under isomorphic ABoxes is natural: we are interested in queries about the structure of ABoxes, and they should not depend on the concrete names of individuals. Dropping this assumption would render the expressivness analysis virtually meaningless: because an OMQ (or any standard database query) can only use a finite number of constants in the query expression, many computationaly trivial queries could not be expressed even in very powerful query languages.

Turing Machines operate on strings. This means that in order to compute an answer to a query over an ABox $\mathcal{A}$, we need to suitably encode $\mathcal{A}$ as a string. We choose a simple encoding, where we first enumerate all pairs c_i, c_j of individuals in $\mathcal{A}$. Then, for each such pair, we store in a *single symbol* all the concept names asserted for those individuals along with the roles that link them. Note that different types of encodings are possible (cf. [5], Chapter 2.2).

Definition 2 (Encoding ABoxes as words)**.** *Consider a fixed signature* Σ*. A* 2-type over Σ *is a tuple* (T, R, T')*, where* $T, T' \subseteq \Sigma \cap \mathsf{N}_C$ *and* $R \subseteq \Sigma \cap \mathsf{N}_R^+$*. We let* Γ^Σ *denote the set of all* 2*-types over* Σ*. We next define an* encoding function enc^Σ *that maps* Σ*-ABoxes to words over* Γ^Σ*. Assume a* Σ*-ABox* $\mathcal{A}$ *with* ℓ *individuals and take an arbitrary enumeration* $c_1, \ldots, c_\ell$ *of the individuals in* $\mathcal{A}$*. Then we define* $enc^\Sigma(\mathcal{A}) = \sigma_{1,1} \ldots \sigma_{1,\ell}\, \sigma_{2,1} \ldots \sigma_{2,\ell} \ldots \sigma_{\ell,1} \ldots \sigma_{\ell,\ell}$*, where each* $\sigma_{i,j} := (\{A \mid A(c_i) \in \mathcal{A}\}, \{r \mid r(c_i, c_j) \in \mathcal{A}\}, \{A \mid A(c_j) \in \mathcal{A}\})$.

Based on the encoding above, we can now define membership of a GBQ in a complexity class.

Definition 3. *Let* Σ *be a signature and* Q *a GBQ over* Σ*. We say* Q belongs to *a complexity class* $\mathcal{C}$*, if the language* $\{enc^\Sigma(\mathcal{A}) \mid \mathcal{A}$ *is a* Σ*-ABox with* $Q(\mathcal{A}) = 1\}$ *over the alphabet* Γ^Σ *belongs to* $\mathcal{C}$.

Proposition 1. *Assume a GBQ* Q *over* Σ*. The following are equivalent:*

1. Q *belongs to coNP.*
2. *There is an integer* k *and a NTM* M *with alphabet* Γ^Σ *such that, for any* Σ*-ABox* $\mathcal{A}$ *we have:*
   a) $Q(\mathcal{A}) = 0$ *iff* M *accepts* $enc^\Sigma(\mathcal{A})$*;*
   b) M *terminates within* $|enc^\Sigma(\mathcal{A})|^k$ *computation steps.*

As we have already seen in the introduction, OMQs based on classical first-order logic do not capture coNP due to their monotonicity. We can also show that the same applies to inconsistency queries based on $\mathcal{ALCHOI}$ with closed predicates, which are non-monotonic in general.

Theorem 1. *There exists a GBQ* Q_1 *over* Σ *such that:*

(a) Q_1 *belongs to* coNP*, and*

(b) *there is no inconsistency query* $Q_2 = (\mathcal{T}, \Sigma')$*, with* $\mathcal{T}$ *in* $\mathcal{ALCHOI}$*, such that* $Q_1(\mathcal{A}) = Q_2(\mathcal{A})$ *holds for all* Σ*-ABoxes* $\mathcal{A}$*.*

Proof. To see this, we can analyze the running time of the algorithm in [3] for checking satisfiability of an $\mathcal{ALCHOI}$ knowledge base $\mathcal{K} = (\mathcal{T}, \Sigma', \mathcal{A})$, where Σ' is a set of closed predicates. We assume that $\mathcal{T}, \Sigma'$ are fixed, and we want an algorithm that takes an ABox $\mathcal{A}$ as input and checks if $\mathcal{K} = (\mathcal{T}, \Sigma', \mathcal{A})$ is consistent. Specifically, based on [3], we have an algorithm that runs in time bounded by $|\mathcal{A}|^k \times \ell + v$, where ℓ and v are constants that depend on $\mathcal{T}$ and Σ', while k is a constant that does not depend on $\mathcal{T}$ or Σ'. In other words, there is a constant k, such that for any $\mathcal{T}$ and Σ', we can build a non-deterministic algorithm that checks the consistency of an input ABox $\mathcal{A}$ in the KB $\mathcal{K} = (\mathcal{T}, \Sigma', \mathcal{A})$, and that runs in time $\mathcal{O}(|\mathcal{A}|^k)$.

The key here is that the constant k does not depend on $\mathcal{T}$ or Σ'. Using an $\mathcal{ALCHOI}$ TBox with closed predicates one can capture decision problems that can be solved via a non-deterministic TM in time that is polynomial with degree k. However using the *Non-deterministic Time Hierarchy Theorem*, we know that there are problems that can be solved in non-deterministic polynomial time, but not in polynomial time with polynomial degree k. Specifically, there a problems solvable in time $\mathcal{O}(n^{k+1})$ but not $\mathcal{O}(n^k)$. □

We note that the above result can be formulated also for OMQs with atomic queries and a certain class of conjunctive queries, but it is unclear if it generalizes to OMQs with CQs. This is because the proof of Theorem 1 relies on an upper bound on the running time of a known algorithm for $\mathcal{ALCHOI}$ with closed predicates. To the best of our knowledge, no suitable upper bounds on data complexity are known in the case of CQs over $\mathcal{ALCHOI}$ KBs. Furthermore, at this point, it is unfortunately unclear whether one can prove the same inexpressibility result for $\mathcal{ALCHOIF}$ with closed predicates. This is left as future work.

4. Language Extension

As stated above, it is unclear whether OMQs based on plain $\mathcal{ALCHOIF}$ with closed predicates are capable of capturing coNP. However, we present an extension of this OMQ language that we prove is powerful enough to do so. To this end, we assume a countably infinite set N_V of variables. We refer to the expressions of the form $\{x\}$, where $x \in \mathrm{N}_V$, as *nominal variables.*

Complex roles are expressions of the form $A? \circ P$ s.t. $A \in \mathrm{N}_C^+$ and P is of the form $r_1 \circ A_1? \circ \ldots r_n \circ A_n$, where $n \geq 1$ and $r_i \in \mathrm{N}_R^+$, $A_i \in \mathrm{N}_C^+$, for all $1 \leq i \leq n$. We call an expression of type $A?$, where $A \in \mathrm{N}_C^+$ a *test role.*

Now, in addition to standard $\mathcal{ALCHOIF}$ axioms, we allow axioms of the following shape:

$$(\mathsf{trans}\ s), \text{ for a restricted role } s \in \mathrm{N}_R \text{ (\textit{transitivity axiom})} \tag{6}$$

$$\exists P.(\{x\} \sqcap \exists s.\{y\}) \sqcap \exists R.\{y\} \sqsubseteq B, \text{ where } B \in \mathrm{N}_C, s \text{ is a restricted role, and} P, R \text{ are complex roles consisting only of tests and functional roles and } x, y \in \mathrm{N}_V \tag{7}$$

$$\exists P.(\{x\} \sqcap \neg\exists s.\{y\}) \sqcap \exists R.\{y\} \sqsubseteq B, \text{ where } B \in \mathrm{N}_C, s \text{ is a restricted role, and } P, R \text{ are complex roles consisting only of tests and functional roles and } x, y \in \mathrm{N}_V \tag{8}$$

$$\{x\} \sqsubseteq \forall P.\{x\}, \text{ where } P \text{ is a complex role} \tag{9}$$

$$\{x\} \sqcap A \sqsubseteq \forall s.\neg\{x\}, \text{ where } A \in \mathrm{N}_C, s \text{ is a restricted role, and } x \in \mathrm{N}_V \tag{10}$$

Note that in the previous definition, we refer to *functional* and *restricted* roles. We say that a role p is functional if $(\mathsf{func}\ p) \in \mathcal{T}$. Intuitively, a role p is restricted if we can guarantee that, in any model $\mathcal{I}$ of the KB, $(e, e') \in p^{\mathcal{I}}$ implies that e and e' are constants occurring in the KB. We next give a syntactic definition that, albeit incompletely, characterizes such roles.

Definition 4. *A concept $A \in \mathrm{N}_C^+$ is* restricted w.r.t. a TBox $\mathcal{T}$ and a set Σ of closed predicates *if one of the following conditions hold: (i) $A \in \Sigma \cup \mathsf{Nom}(\mathcal{K}) \cup \{\bot\}$, or (ii) $A \sqsubseteq B_1 \sqcup \cdots \sqcup B_n \in \mathcal{T}$, where B_i is a restricted concept or an expression of the form $\exists r$ or $\exists r^-$, for a* restricted role r.

A role $r \in \mathrm{N}_R^+$ is called restricted w.r.t. a TBox $\mathcal{T}$ and a set Σ of closed predicates *if one of the following holds: (i) $r \in \Sigma$, (ii) $\{\exists r \sqsubseteq A, \exists r^- \sqsubseteq B\} \subseteq \mathcal{T}$, where A and B are restricted concepts, (iii) r^- is a restricted role, or (iv) $r \sqsubseteq s$, where s is a restricted role.*

Extended semantics Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation and $\mathcal{K}$ be a KB. The extension of the interpretation function $\cdot^{\mathcal{I}}$ to complex roles P of the form $A? \circ r_1 \circ A_1? \circ \cdots \circ r_n \circ A_n?$ is defined as $P^{\mathcal{I}} := \{e_0 \in \Delta^{\mathcal{I}} \cap A^{\mathcal{I}} \mid \exists e_1, \ldots, e_n \in \Delta^{\mathcal{I}}$ s.t. $(e_{i-1}, e_i) \in r_i^{\mathcal{I}}, e_i \in A_i^{\mathcal{I}}$, for all $1 \leq i \leq n\}$. The semantics of transitivity axioms is standard: $\mathcal{I}$ satisfies $(\mathsf{trans}\ s)$ if $(e_1, e_2) \in s^{\mathcal{I}}$ and $(e_2, e_3) \in r^{\mathcal{I}}$ implies $(e_1, e_3) \in r^{\mathcal{I}}$. The axioms of the form (7)-(10) are reminiscent of nominal schemas introduced in [8, 7]. In these works, the semantics of such nominal schemas is given by grounding the knowledge base with respect to the set of all individuals N_I, where N_I is assumed to be finite. For our purposes, we ground $\mathcal{K}$ with respect to $\mathsf{Nom}(\mathcal{K})$ by uniformly replacing all nominal variables with nominals in $\mathsf{Nom}(\mathcal{K})$ in all possible ways. We use $\mathsf{ground}(\mathcal{K})$ to denote such grounding of $\mathcal{K}$ and we say that $\mathcal{I}$ satisfies $\mathcal{K}$ if it satisfies $\mathsf{ground}(\mathcal{K})$.

Theorem 2. *The KB satisfiability problem in $\mathcal{ALCHOIF}$ with closed predicates extended with axioms of the form (6)-(10) is NP-complete in data, and NExpTime-complete in combined complexity.*

Proof sketch. Due to space restrictions, we only offer a brief sketch of the decision procedure that runs in nondeterministic exponential time in the size of the given knowledge base, and in nondeterministic polynomial time, in if the TBox and the set of closed predicates are considered fixed. The first step in this procedure is to guess the extensions of restricted concepts and roles over the individuals occurring in the given knowledge base. These concepts and role names are now considered closed. Since all transitivity axioms only involve restricted roles, we can right away check whether they are satisfied and eliminate them. Thus, what is left to do is devise a procedure that can decide satisfiability of KBs with closed predicates whose TBox contains no transitivity axioms, and where all restricted concepts and role names are now considered closed. We do this by modifying the mosaic approach introduced in [4] for $\mathcal{ALCHOIF}$ with closed predicates to support complex roles and nominal schemas.

5. The Encoding

Theorem 3 (Main result)**.** *Assume a signature Σ and a GBQ Q over Σ that belongs to coNP. Then there is a TBox $\mathcal{T}$ in extended $\mathcal{ALCHOIF}$ such that the Boolean inconsistency query OMQ $q = (\mathcal{T}, \Sigma)$ has the following property: for all Σ-ABoxes $\mathcal{A}$, $Q(\mathcal{A}) = q(\mathcal{A})$.*

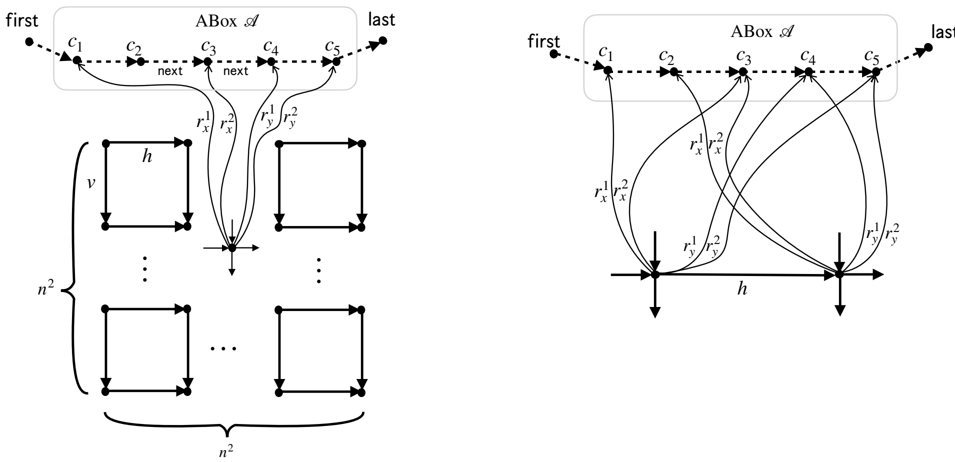

Figure 1: Construction of the $n^k \times n^k$ grid, for $k = 2$. Left: Assigning coordinates to grid nodes. Right: Propagation of coordinates along horizontal successors.

The rest of this section serves as a proof sketch for the theorem above. Let Q be a GBQ over some signature Σ that is in coNP. According to Proposition 1, there is a nondeterministic Turing machine M that decides the language $\{enc^{\Sigma}(\mathcal{A}) \mid Q(\mathcal{A}) = 0, \mathcal{A} \text{ is a } \Sigma\text{-ABox}\}$ and its running time is bounded by n^k, where k is a constant and n is the size of the input word. We show that we can come up with a TBox $\mathcal{T}_M$ such that for the ontology-mediated inconsistency query $Q_M = (\mathcal{T}_M, \Sigma)$, $Q_M(\mathcal{A}) = Q(\mathcal{A})$, for all Σ-ABoxes $\mathcal{A}$. The basic idea is to craft $\mathcal{T}_M$ in a way that ensures that all models of $(\mathcal{T}, \Sigma, \mathcal{A})$ contain a grid structure of size $n^k \times n^k$. We then use this grid to simulate the given Turing machine M as follows. The first row of the grid stores the initial configuration of M while each subsequent row stores the next configuration in some computation of M. Finally, we eliminate those computations that do not end in acceptance of the word. As a result, we have that each model of $(\mathcal{T}_M, \Sigma, \mathcal{A})$ corresponds to a computation of M that accepts $enc^{\Sigma}(\mathcal{A})$, and vice versa: every computation of M that ends in acceptance of $enc^{\Sigma}(\mathcal{A})$ corresponds to some model of $(\mathcal{T}_M, \Sigma, \mathcal{A})$, for all Σ-ABoxes $\mathcal{A}$. Thus, checking whether M accepts $enc^{\Sigma}(\mathcal{A})$ boils down to checking whether $(\mathcal{T}_M, \Sigma, \mathcal{A})$ is unsatisfiable, which is equivalent answering the inconsistency query Q_M.

We now begin with our construction. In the rest of this section we assume we are given a GBQ in the form of a nondeterministic Turing machine $M = (\Gamma^{\Sigma}, Q, \delta, q_0, q_{acc}, q_{rej})$ and an integer constant k.

5.1. Constructing the $n^k \times n^k$ Grid

Consider an arbitrary ABox $\mathcal{A}$ over some signature Σ. We next show how to build a KB $\mathcal{K} = (\mathcal{T}, \Sigma, \mathcal{A})$ s.t. that each model $\mathcal{I}$ of $\mathcal{K}$ contains a $n^k \times n^k$ grid formed by the domain elements, where n is the number of known individuals (i.e., the number of individuals occurring in $\mathcal{A}$ plus two special constants first and last).

We begin by generating different domain elements that serve as grid nodes. Each such grid node is associated two words of length k over the known individuals that serve as its x-

and y-coordinate in the grid. This is accomplished using roles two sets of functional roles: $r_x^1, \ldots r_x^k$ and $r_y^1, \ldots r_y^k$. We say that a domain element e has an x-coordinate $c_1 c_2 \cdots c_k$, if (e, c_i) participates in r_x^i, for each $i, 1 \leq i \leq k$. The y-coordinate of e is defined analogously. This is illustrated in Figure 1, left. In the first step of the construction, we let the special individual first be the origin of the grid and set its x- and y- coordinates to $\mathsf{first} \cdots \mathsf{first}$. To generate the remainder of the grid nodes, we add axioms that create a binary tree rooted in first using two roles h and v, denoting horizontal and vertical successors, respectively. We next assign the x- and y-coordinates to each grid node in the tree making sure that they respect a certain pattern. To do this, we use a linear order over the known individuals that we can can easily generate by using first and last as the designated first and last elements and guessing the remaining part of the successor relation, encoded using the role . We then lift this linear order to words of length k over the available individuals and add axioms that require that for each grid node e with the horizontal successor e', the x-coordinate of e' is the successor of the x-coordinate of e with respect to the generalized linear order, while the y-coordinate remains unchanged. We then do a similar thing with the vertical successor e'' of e. Namely, the y-coordinate of e'' is the successor of the y-coordinate of e with respect to the generalized linear order, while the x-coordinate stays the same. Figure 1, on the right illustrates this. It is not hard to see that all possible pairs of x- and y-coordinates occur within this tree. Now, the only thing that is left to do is to merge nodes with same coordinates. This is easy: we simply let the special individual last be the only grid node with $\mathsf{last} \cdots \mathsf{last}$ as its x- and y-coordinate. Propagating backwards from last while relying on the fact that each grid node has at most one h- and at most one v-predecessor, we can easily see that each different combination of the coordinates occurs exactly one time – thus we have n^{2k} different grid nodes. Moreover, the way we assigned their coordinates ensures that they form a proper grid. We next detail the construction by providing all the relevant axioms.

Collect constants from $\mathcal{A}$. We first collect in Adom all the individuals occurring in $\mathcal{A}$:

$$\mathsf{Adom} \equiv \bigsqcup_{A \in \mathsf{N}_C^+ \cap \Sigma} A \sqcup \bigsqcup_{r \in \mathsf{N}_R^+ \cap \Sigma} (\exists r \sqcup \exists^- r)$$

Guess a linear order. We next add the axioms that guess a linear order over the known individuals, stored using the concept name Node. We use two individuals First and Last as designated first and last elements in this linear order. The role stores the successor relation, and lessThan is a role that stores the induced "less than" relation.

$$\begin{array}{ll}
\mathsf{Node} \equiv \mathsf{Adom} \sqcup \{\mathsf{First}\} \sqcup \{\mathsf{Last}\} & \{x\} \sqcap \mathsf{Node} \sqsubseteq \forall \mathsf{lessThan}.\neg\{x\} \\
\mathsf{Node} \sqsubseteq \exists \mathsf{next}.\mathsf{Node} \sqcup \{\mathsf{Last}\} & \mathsf{next} \sqsubseteq \mathsf{lessThan} \\
\mathsf{Node} \sqsubseteq \exists \mathsf{next}^-.\mathsf{Node} \sqcup \{\mathsf{First}\} & (\mathsf{trans}\ \mathsf{lessThan}) \\
(\mathsf{func}\ \mathsf{next}) & \exists \mathsf{lessThan} \sqsubseteq \mathsf{Node} \\
(\mathsf{func}\ \mathsf{next}^-) & \exists \mathsf{lessThan}^- \sqsubseteq \mathsf{Node}
\end{array}$$

Axioms on the left-hand side are responsible for guessing the successor relation of the linear order that is being generated. They ensure that all individuals except for the last one have a successor, and all individuals except for the first one have a predecessor. Moreover, successors and predecessors must be unique. Axioms on the right-hand side says that the transitive closure of contains no cycles, meaning that we have a proper linear order. The last two axioms serve as guards to ensure that a transitivity assertion is made over a restricted role.

Creating the grid structure. To create a $n^k \times n^k$ grid, we take the approach above and add, for all $1 \leq i \leq k$, the following axioms that create a binary tree routed in first using h and v:

$$
\begin{array}{ll}
\mathsf{GridNode} \sqsubseteq \bigsqcap_{1\leq i\leq k}(\exists r_x^i.\mathsf{Node} \sqcap \exists r_y^i.\mathsf{Node}) & (\mathsf{func}\ h) \\
\{\mathsf{First}\} \equiv \mathsf{GridNode} \sqcap \bigsqcap_{1\leq i\leq k}(\exists r_x^i.\{\mathsf{First}\} \sqcap \exists r_y^i.\{\mathsf{First}\}) & (\mathsf{func}\ h^-) \\
\mathsf{GridNode} \sqsubseteq \exists h.\mathsf{GridNode} \sqcup (\bigsqcap_{1\leq i\leq k} \exists r_x^i.\{\mathsf{Last}\}) & (\mathsf{func}\ v) \\
\mathsf{GridNode} \sqsubseteq \exists v.\mathsf{GridNode} \sqcup (\bigsqcap_{1\leq i\leq k} \exists r_y^i.\{\mathsf{Last}\}) & (\mathsf{func}\ v^-) \\
\bigsqcap_{1\leq i\leq k} \exists r_x^i.\{\mathsf{Last}\} \sqsubseteq \neg\exists h.\top & (\mathsf{func}\ r_x^i) \\
\bigsqcap_{1\leq i\leq k} \exists r_y^i.\{\mathsf{Last}\} \sqsubseteq \neg\exists v.\top & (\mathsf{func}\ r_y^i)
\end{array}
$$

The first axiom on the left-hand side states that every grid node has 2k pointers to the known individuals using functional roles r_x^i and r_y^i, $1 \leq i \leq k$ that encode its x- and y-coordinates. The second axiom on the left-hand side sets first as a designated origin point with x- and y-coordinates first $\cdot$ first. The rest of the axioms simply create the tree.

We next make sure that the coordinates align, i.e., if e' is an h-successor of e, then the y-coordinates of e and e' coincide, while the x-coordinate of e' is a successor of the x-coordinate of e w.r.t. to the linear order in extended to words of length k. For example, if the x-coordinate of e is $c_k \cdots c_i \cdot \mathsf{last} \cdots \mathsf{last}$, where $c_i \neq 1$, then the x-coordinate of d is $c_k \cdots c_i' \cdot \mathsf{first} \cdots \mathsf{first}$, where c_i' is the successor of c_i according to the given linear order. We now define the axioms that do this and add for all i, $1 \leq i \leq k$:

$$
\begin{array}{l}
\mathsf{GridNode} \sqcap \neg\exists r_x^i.\{\mathsf{Last}\} \sqcap \bigsqcap_{1\leq j\leq i} \exists r_x^j.\{\mathsf{Last}\} \sqsubseteq \mathsf{IncrX}_i \\
\mathsf{IncrX}_i \sqsubseteq \forall h.(\bigsqcap_{1\leq j\leq i} \exists r_x^j.\{\mathsf{First}\}) \\
\{\mathsf{First}\} \sqsubseteq \forall (r_x^i)^-.\forall h^-.\forall r_x^i.\{\mathsf{Last}\}) \\
\{x\} \sqsubseteq \forall \mathsf{Node}? \circ (r_x^i)^- \circ \mathsf{IncrX}_i? \circ h \circ r_x^i \circ \mathsf{next}^-.\{x\}) \\
\{x\} \sqsubseteq \forall \mathsf{Node}? \circ (r_x^j)^- \circ \mathsf{IncrX}_i? \circ h \circ r_x^j.\{x\} \\
\{x\} \sqsubseteq \forall \mathsf{Node}? \circ (r_y^i)^- \circ h \circ r_y^i.\{x\}
\end{array}
$$

We only show how to handle the x-coordinate, since the y-coordinate is treated analogously. Finally, we add the axiom that triggers the merging of the nodes with same coordinates:

$$
\{\mathsf{Last}\} \equiv \mathsf{GridNode} \sqcap \bigsqcap_{1\leq i\leq k} (\exists r_x^i.\{\mathsf{Last}\} \sqcap \exists r_y^i.\{\mathsf{Last}\})
$$

5.2. Encoding the Turing Machine

We next simulate the computation of M using the grid we just created. We assume we have the following concept names available: (i) $A_1, \bar{A}_1, A_2, \bar{A}_2, A_s, \bar{A}_s$, for all $A \in \Sigma \cap \mathrm{N}_C$ and all $r \in \Sigma \cap \mathrm{N}_R$, (ii) L_γ, for all symbols $\gamma \in \Gamma^{\Sigma'} \cup \{B\}$, (iii) S_q, for all $q \in Q$, and (iv) $H_<$ and $H_>$.

Copying $\mathcal{A}$ onto the input tape. The first row of the grid, referred to as the *input tape*, represents the initial configuration of M. Recall that we encode Σ-ABoxes over the signature as words of length n^2 where each position in the word represents a pair of individuals in $\mathcal{A}$ and each pair of individuals occurring in $\mathcal{A}$ is represented by one position in the word. We now add axioms that make sure that each one of the first n^2 cells on the input tape corresponds to a single pair of individuals occurring in the KB, while the remainder of the cells on the input tape are filled out with the blank symbol B. This is done by assuring that every input cell, i.e., a node in the first row, has two pointers to known individuals: hasFst and hasSnd. If for some input cell e there are two known individuals a, b s.t. (e, a) participates in hasFst and (e, b) participates in hasSnd, then e represents the pair (a, b). To ensure that all pairs are represented on the input tape, we follow the same approach as for the grid construction. Namely, the available linear order is lifted to pairs of known individuals, and we require that a horizontal successor of some input cell also represents the next pair w.r.t. to this linear order. Once all pairs are represented, the remaining input cells are set to blank, i.e., they participate in the concept L_B. We defer the exact axioms to the appendix.

We next need put the correct symbols in each cell on the input tape. Recall that, if position i in the encoding of the ABox represents the pair (a, b), we have the following symbol at position i: $\gamma = (\{A \mid A(a) \in \mathcal{A}\}, \{r \mid r(a, b) \in \mathcal{A}\}, \{A \mid A(b) \in \mathcal{A}\}) \in \Gamma^{\Sigma'}$. We next add axioms that ensure exactly that. Namely, if a cell on the input tape represents the pair (a, b), then it participates in the concept L_γ. We first copy the information about which concept and role names a and b participate in. To this end, for $A \in \Sigma \cap \mathrm{N}_C$ and every $r \in \Sigma \cap \mathrm{N}_R$ we add:

$$
\begin{array}{lll}
A_1 \sqcap \bar{A}_1 \sqsubseteq \bot & \exists \mathsf{hasFst}.A \sqsubseteq A_1 & \exists \neg \mathsf{hasFst}.A \sqsubseteq \bar{A}_1 \\
A_2 \sqcap \bar{A}_2 \sqsubseteq \bot & \exists \mathsf{hasSnd}.A \sqsubseteq A_2 & \exists \neg \mathsf{hasSnd}.A \sqsubseteq \bar{A}_2 \\
A_s \sqcap \bar{A}_s \sqsubseteq \bot & \multicolumn{2}{l}{\exists \mathsf{hasFst}.(\{x\} \sqcap \exists s.\{y\}) \sqcap \exists \mathsf{hasSnd}.\{y\} \sqsubseteq A_s} \\
 & \multicolumn{2}{l}{\exists \mathsf{hasFst}.(\{x\} \sqcap \neg \exists s.\{y\}) \sqcap \exists \mathsf{hasSnd}.\{y\} \sqsubseteq \bar{A}_s}
\end{array}
$$

Finally, for each $\gamma = (T, R, T') \in \Gamma^{\Sigma}$, we add:

$$
\bigsqcap_{\substack{A \in T, \\ B \in (\Sigma \cap \mathrm{N}_C) \setminus T}} (A_1 \sqcap \bar{B}_1) \sqcap \bigsqcap_{\substack{A \in T', \\ B \in (\Sigma \cap \mathrm{N}_C) \setminus T'}} (A_2 \sqcap \bar{B}_2) \sqcap \bigsqcap_{\substack{s \in R, \\ (r \in \Sigma \cap \mathrm{N}_R) \setminus R}} (A_s \sqcap \bar{A}_r) \sqsubseteq L_\gamma,
$$

We now use the rest of the grid to simulate the computation of the TM M. Recall that a row in the grid stores a configuration that M is currently in, while v corresponds to time. To this end, we need to ensure that for each row ϱ in the grid satisfies two conditions. Firstly, there is exactly one element e in ϱ where S_q holds for some and at most one $q \in Q$. For other elements $e' \neq e$ in ϱ, S_q does not hold for any q. Secondly, for all elements of e in ϱ, L_γ holds for exactly one $\gamma \in \Gamma \cup \{B\}$. It is then clear that each row is indeed a valid encoding of some configuration of M. If S_q for a node e in ϱ, then M is in state q and the read-write head is in the position e.

We next add the axioms that ensure that at the beginning, M is in the state q_0 and the read-write head is above the first symbol:

$$
\mathsf{InputCell} \sqcap \bigsqcap_{1 \leq i \leq k} r_x^i.\{\mathsf{First}\} \sqsubseteq S_{q_0} \qquad S_q \sqsubseteq \bigsqcap_{q' \in (Q \setminus \{q\})} S_{q'}, \text{ for all } q \in Q
$$

Further, for all $(q,\gamma) \in Q \times \Gamma \cup \{B\}$, we add the following axiom that selects one configuration among possible next configurations, and overwrites the current symbol, changes the state and moves the read-write head accordingly:

$$S_q \sqcap L_\gamma \sqsubseteq \big(\bigsqcup_{(q',\gamma',+1)\in\delta(q,\gamma)} \forall v.(L_{\gamma'} \sqcap \forall h.S_{q'})\big) \sqcup \big(\bigsqcup_{(q',\gamma',-1)\in\delta(q,\gamma)} \forall v.(L_{\gamma'} \sqcap \forall h^-.S_{q'})\big)$$

For all states $q \in Q$, we mark the positions that are not under the read-write head:

$$S_q \sqsubseteq (\forall h.H_<) \sqcap (\forall h^-.H_>) \qquad H_< \sqsubseteq \bigsqcap_{q\in Q} S_{q'} \sqcap \forall h.H_< \qquad H_> \sqsubseteq \bigsqcap_{q\in Q} S_{q'} \sqcap \forall h^-.H_<$$

The intuition of the above is as follows. If in some position e we have S_q, then all the position to the right from e are marked with $H_<$. Intuitively, $H_<$ (resp. $H_>$) says that the read-write head is behind (resp. ahead) and thus these positions do not participate in S_q, for any q.

As one of the last steps, we need to add an axiom that copies the content of the tape that is not overwritten. For all $\gamma \in \Gamma \cup \{B\}$ we add: $\quad L_\gamma \sqcap (H_> \sqcup H_<) \sqsubseteq \forall v.L_\gamma$

We are now almost done with our construction of $\mathcal{T}_M$: for any Σ-ABox $\mathcal{A}$, each model of $(\mathcal{T}, \Sigma, \mathcal{A})$, where $\mathcal{T}$ is the TBox we have constructed so far, corresponds to one possible computation of M on the encoding of $\mathcal{A}$. By assumption, M always terminates, which means that in each model of the theory we will either have some object for which q_{acc} holds or some object for which q_{rej} holds. Finally, to obtain $\mathcal{T}_M$, we add the axiom $q_{rej} \sqsubseteq \bot$ to $\mathcal{T}$. Now, every model of $(\mathcal{T}_M, \Sigma, \mathcal{A})$ corresponds to computation of M accepting the encoding of $\mathcal{A}$. Thus, for $Q_M = (\mathcal{T}_M, \Sigma)$, $Q_M(\mathcal{A}) = 1$ if and only if there are no accepting computations of M ran on enc^Σ, that is, $Q(A) = 1$.

6. Discussion

In this paper, we have discussed some of the expressiveness limitation of very expressive OMQ languages, and then proposed an extension of $\mathcal{ALCHOIF}$ equipped with closed predicates as OMQ language that captures precisely the class of generic Boolean queries over ABoxes that are computable in coNP.

The arguments presented in the paper can also be applied to standard Horn-DLs (with no closed predicates). For instance, the OMQ language that couples inconsistency and instance queries with $\mathcal{ELHIF}$ ontologies is PTime-hard, but it cannot express all queries computable in PTime. It is not difficult see that extending $\mathcal{ELHIF}$ with a built-in linear order is not sufficient to capture PTime, but the further addition of the features described in Section 4 leads to a DL that allows to precisely capture PTime.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. It was also partially supported by the Austrian Science Fund (FWF) project P30873.

References

[1] M. Ortiz, D. Calvanese, T. Eiter, Data complexity of query answering in expressive description logics via tableaux, J. Autom. Reason. 41 (2008) 61–98. URL: https://doi.org/10.1007/s10817-008-9102-9. doi:10.1007/s10817-008-9102-9.

[2] M. Bienvenu, B. ten Cate, C. Lutz, F. Wolter, Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP, ACM Trans. on Database Systems 39 (2014) 33:1–33:44. URL: http://doi.acm.org/10.1145/2661643. doi:10.1145/2661643.

[3] S. Ahmetaj, M. Ortiz, M. Simkus, Polynomial rewritings from expressive description logics with closed predicates to variants of datalog, Artificial Intelligence 280 (2020) 103220. URL: https://doi.org/10.1016/j.artint.2019.103220. doi:10.1016/j.artint.2019.103220.

[4] T. Gogacz, S. Lukumbuzya, M. Ortiz, M. Simkus, Datalog rewritability and data complexity of ALCHOIF with closed predicates, in: D. Calvanese, E. Erdem, M. Thielscher (Eds.), Proc. of KR 2020, 2020, pp. 434–444. URL: https://doi.org/10.24963/kr.2020/44. doi:10.24963/kr.2020/44.

[5] N. Immerman, Descriptive complexity, Graduate texts in computer science, Springer, 1999. URL: https://doi.org/10.1007/978-1-4612-0539-5. doi:10.1007/978-1-4612-0539-5.

[6] C. Lutz, I. Seylan, F. Wolter, The data complexity of ontology-mediated queries with closed predicates, Logical Methods in Computer Science 15 (2019). URL: https://doi.org/10.23638/LMCS-15(3:23)2019. doi:10.23638/LMCS-15(3:23)2019.

[7] M. Krötzsch, F. Maier, A. Krisnadhi, P. Hitzler, A better uncle for OWL: nominal schemas for integrating rules and ontologies, in: S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, R. Kumar (Eds.), Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011, ACM, 2011, pp. 645–654. URL: https://doi.org/10.1145/1963405.1963496. doi:10.1145/1963405.1963496.

[8] M. Krötzsch, S. Rudolph, Nominal schemas in description logics: Complexities clarified, in: C. Baral, G. D. Giacomo, T. Eiter (Eds.), Proc. of KR 2014, AAAI Press, 2014. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/8027.