Ontology-Based Query Answering: Expressivity and Extensions

Michaël Thomazo

▶ **To cite this version:**

Michaël Thomazo. Ontology-Based Query Answering: Expressivity and Extensions. Information Retrieval [cs.IR]. Ecole Normale Supérieure de Paris, 2023. tel-04275013

HAL Id: tel-04275013
https://hal.science/tel-04275013

Submitted on 8 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PSL UNIVERSITÉ PARIS

HABILITATION À DIRIGER DES RECHERCHES

DE L'UNIVERSITÉ PSL

Présentée à l'École normale supérieure

Ontology-Based Query Answering: Expressivity and Extensions

Présentation des travaux par

Michaël Thomazo

Discipline

Informatique

Soutenue le 04/10/2023

Composition du jury :

Carsten LUTZ PR, Leipzig Universität	*Rapporteur*
Marie-Christine, ROUSSET PR, UGA	*Rapporteuse*
Balder, TEN CATE Ass. PR, University of Amsterdam	*Rapporteur*
Diego, Figueira CR, CNRS/Univ. Bordeaux	*Examinateur*
Luc, SEGOUFIN DR, INRIA/ENS/PSL	*Examinateur*
Sophie, TISON PR, Université de Lille	*Examinatrice*

ENS | PSL

1 Introduction

This dissertation focuses on ontology-based query answering (OBQA), which has been studied over the last fifteen years both in the database community (which meets and publishes at conferences such as ICDT, PODS, SIGMOD, VLDB) and in the knowledge representation and reasoning community (which meets and publish at conferences such as KR, and IJCAI, or the DL workshop for the description logic subcommunity). The goal of OBQA is to facilitate access to data by using ontologies, which in this context are to be understood as logical formalizations of domains of interest. Ontologies allow users to seamlessly query data from several data sources, using a vocabulary they are familiar with, while reaping the benefits of reasoning to get more complete sets of answers.

Ontologies allow to state, for example, that "oxytocin is a hormone". When checking whether obesity can be induced by a hormone, the query can then be reformulated, among other, to check whether oxytocin can induce obesity, allowing to identify answers in sources that would not explicitly mention the word "hormone". Crucially, this reformulation is done in an explainable way, as the reasoning leading to a given answer can be followed and justified at every step. Such ontologies have been developed for domains such as governance and administration [75], digital media [16], e-commerce [51], but have been especially well adopted within health and life science applications, where practitioners have led the development of ontologies such as SNOMED CT [1].

My research focuses on the the theoretical underpinnings of ontology-based query answering, in which we abstract away from the implementation at a system level, and focus on the formal developments of such an approach. This problem takes as input a set of atoms I (an abstraction of a database), a set of logical formulas $\mathcal{R}$ (meant to represent knowledge regarding a given domain of application) and a query q (representing the information need of a user). The goal is to provide answers to the query. A main focus of the research pursued in the field has been to study the decidability and the complexity of the associated decision problem, in which one usually focuses on Boolean conjunctive queries. Formulated in a different manner, we wonder whether $I, \mathcal{R} \models q$, where $\models$ represent the entailment relation of first-order logic.

OBQA is *undecidable*, meaning that no algorithm can correctly solve it in finite time on all instances. The community has thus spent a lot of efforts on delimiting the limits of decidability, that is, to find which restrictions to put on the instances to ensure that there are (hopefully efficient) algorithms to solve OBQA. For instance, taking an empty set of formulas $\mathcal{R}$ reduces the problem to a problem that is routinely solved by relational database management systems. As undecidability proofs have been found with very simple queries and databases, the considered restrictions have thus focused on the logical layer part. A wealth of such restrictions have been defined, and I refer the interested reader to [64] for an overview, or Section 2 for a quicker introduction to the field.

While the core framework is now rather well understood from a decidability and complexity point of view, extensions of this framework are still open to exploration. In this dissertation, I present results concerning the expressivity of the framework, as well as relevant extensions:

- what is the expressivity made available by using an ontology as an intermediate layer in which to express queries over the data?
- what are adequate query languages to be used on top of ontologies, which are the computational challenges associated with these query languages, and how to overcome

them?

- how to gather additional information on the querying process, in order to annotate query answers with meta-information explaining their validity?

1.1 Overview of Contributions

1.1.1 Expressivity of Existential Rules as a Query Language

The first set of contributions that we illustrate relates to the field of *descriptive complexity* [54]. A lot of research in (theoretical) computer science has been motivated by the following high-level question: "how hard is the computational task at hand?". Let us consider for instance the problem of k-colorability of a graph, for a fixed k. This problems takes a graph $G = (V, E)$ as input, where V is a finite set and $E \subseteq V \times V$, and asks whether there exists a k-coloring, that is, a mapping c from V into $\{1, \ldots, k\}$ such that if $(u, v) \in E$, then $c(u) \neq c(v)$. A naive approach would be to try every mapping from V into $\{1, \ldots, k\}$, which results in an algorithm that runs in $\mathcal{O}(k^{|V|})$, which is unpractical on large graphs. A natural quest is to look for better algorithms, but all known algorithms remain of exponential complexity in the input [62, 20].

In a dual fashion, one can also look for lower bounds, and complexity theory [6] comes here to the rescue. Instead of describing the complexity of an algorithm, we here focus on the complexity of a (decision) problem, which is defined by the amount of resources that are required by a given computational model to solve that problem. For instance, the PTime class is the set of problems that can be solved by a deterministic Turing machine that runs in polynomial time. The NP class is the set of problems that can be solved by a non-deterministic Turing machine that runs in polynomial time. Among these problems, some are "as hard" as any other problem of the class, and called "complete" (under some type of reduction). For instance, a problem A is NP-complete (for polynomial time reduction, for instance), if A belongs to NP and any NP problem can be reduced in polynomial time to A. These classes have the advantages of being robust (as small modifications of the input do not modify these classes), but separating them is a notoriously hard problem.

Another way to look at the complexity of problems is to tell which resources are enough, not to solve the problem, but to describe the positive instances of that problem. Resources here is to be understood as a logical formalism. For instance, given a vocabulary for representing graphs, the positive instances of the 3-colorability problem are characterized by the following second order formula:

$$\exists C_1\ \exists C_2\ \exists C_3 \forall x[(C_1(x) \vee C_2(x) \vee C_3(x) \wedge \forall y(E(x,y) \rightarrow \\ \neg(C_1(x) \wedge C_1(y)) \wedge \neg(C_2(x) \wedge C_2(y)) \wedge \neg(C_3(x) \wedge C_3(y)))],$$

where C_1, C_2 and C_3 are unary second-order variables, while x is a first-order variable. An attentive reader will have noticed that all second-order quantifiers are existential, and this possibility is actually a consequence of Fagin's theorem (see [54] for a proof), stating that all NP-problems can be stated as an existential second-order formula. A major open question in the field of descriptive complexity is to define a logic that captures exactly the class PTime.

A result in that direction is the well-known Immerman-Vardi theorem [53, 77], which express that every polynomial query on an *ordered* database can be expressed by an order-invariant semipositive Datalog program. In the first paper presented in this section, we

study more precisely the impact of the two involved features, semipositivity and of order-invariance, on the expressivity of Datalog. We fully describe a Venn diagram that represent polynomial time queries, homomorphism-closed queries, and four variants of Datalog (with or without semipositivity, with or without access to an order). The most striking result is that semipositivity (and thus, access to some form of negation) is *necessary* to express some homomorphism-closed query, which is quite counter-intuitive.

In the second paper, we consider the other side of the spectrum regarding expressivity of existential rules: characterize which queries can be expressed with existential rules whose restricted chase[1] is terminating. We, quite surprisingly, show that it captures all the queries that can reasonably be expected to be captured, that is, all homomorphism-closed decidable queries (both upper bounds are quite direct, once the chase and its fundamental property is known).

1.1.2 A More Knowledge Representation View: Enhancing the Query Part

The previous results provide a clean view of what can be expressed with existential rules (or some fragments thereof), when existential rules are seen as a query language. This is however somewhat misleading, from a knowledge representation point of view, to consider a set of existential rules as a query. Intuitively, a query is user-defined, and it seems unreasonable to expect a user to define a whole set of existential rules as a query. Indeed, the definition of ontologies is a task that can easily span years, as witnessed for instance by the development of SNOMED CT. Much more reasonable are conjunctive queries, as introduced earlier, assuming that the ontology part has been independently defined. And indeed, most of the work in ontology-mediated query answering has focused on answering Boolean conjunctive queries over knowledge bases, where the ontology is chosen to be expressed over a variety of languages. In this setting, it becomes clear that extending the query language part is important to gain expressivity, and this motivated us to study other query languages to be evaluated in the presence of given ontology languages.

These topics have been at the core of my collaboration with Meghyn Bienvenu, starting from [19], in which we investigated how to adapt some of her previous work [18] to the existential rule setting. We first considered a rather "simple" case, that of regular path queries in the presence of linear rules, a well known fragment of existential rules famous for ensuring good computational properties with respect to conjunctive query answering. We then joined forces with Jean-François Baget and Marie-Laure Mugnier, to lift these results towards conjunctive regular path queries and guarded rules [11]. The collaboration with Meghyn then continued through the CQFD project, and in particular with the supervision of Quentin Manière, still aiming at extending the capabilities of the query part. The limitation to Boolean queries is a strong limitation from a practical point of view, as quite often aggregation operators are used in analytical queries. As a first step towards the support of such queries in ontology-based data access queries, we considered counting queries.

(Conjunctive) Regular Path Queries Some applications, such as the search for paths on a map, require the ability to check efficiently for neighbours in an underlying graph structure. This motivates the use of *graph databases*, which are structured in such a way that this operation is performed efficiently, and the definition of *regular path queries*, that asks for the pair of vertices (s, t) such that there is path from s to t that is labeled by a word from a given query language Λ. Such queries are both studied from a theoretical point of

[1] The details of the restricted chase will be recalled in Section 2.

view (see [41] for an introduction) and are at the core of multiple practical query languages (see [78] for a survey). Note that GQL, a yet unpublished ISO standard is currently under development [43] for graph query languages.

Several semantics for these queries have been considered in the literature, depending on which paths are considered: arbitrary paths, simple paths, and trails. Simple paths enforce that no nodes appear twice on a given path, while trails enforce that no arc is used twice on a given path. We focused on the arbitrary path semantics, because it is both efficiently evaluable and received significant attention in the literature. Extensions of regular path queries are also of interest, and notably the conjunctive regular path queries, which intuitively are conjunctive queries, where each atom can be a regular path query.

In the setting of incomplete data, conjunctive regular path queries have been studied for the lightweight description logics [18]. The technical development in such a setting is facilitated due to a strong tree-likeness of the structures involved in query answering (that is, of the chase, that will be defined formally later). We studied, first with Meghyn Bienvenu, then in collaboration with Jean-François Baget and Marie-Laure Mugnier, how to extend these results when the tree-likeness of the chase is (partially) relaxed, first for regular path queries, then for conjunctive regular path queries.

Counting Queries The queries introduced so far are all Boolean queries, that is, queries whose answer can be seen as a yes or a no. A first limitation of these queries is that it does not provide any answer tuple. This is not a strong limitation, as these problems are usually equivalent one to the other. A more worrying limitation is that it does not fit well what is actually used in practice, as some important features are missing. This is the case for aggregation, such as count, min, max, average,... We focused on the count operator, whose semantics over a relational database, which assumes complete information, is quite clear. However, in the setting of ontology-mediated query answering, their semantics is much less obvious to define: one of the goal of adding ontologies is to take into account data that is not explicitly stated, and as such, a closed-world assumption cannot be made anymore. This led to the definition of several alternative semantics for counting under incomplete information [30, 56, 3, 40].

The starting point of the investigation performed with Meghyn Bienvenu and Quentin Manière in the setting of Quentin's PhD has been a new definition for counting conjunctive queries, and the definition of their semantics by generalizing the semantics proposed by Kostylev and Reutter [56]. Starting from there, we undertook a systematic exploration of the sources of complexity for such queries under ontologies expressed in description logics that include those typically used in the setting of ontology based query answering. While the literature used to consider conjunctive queries in their whole generality, which we also did, we also tried to identify settings in which the complexity would be more reasonable. This happened to prove quite challenging, given that the classical restrictions were mostly ineffective at decreasing complexity. We finally decided to study in more details the case of atomic counting queries, coined cardinality queries (we compute the minimum cardinality of a given predicate over all models of the knowledge base), that are split in concept cardinality queries and role cardinality queries, depending on the arity of the considered predicate (in this section, we consider description logics, hence predicates are either unary or binary).

A wealth of techniques have been used, and we will give the gist of two of them:

- to provide upper bounds, we show that models that minimize the number of matches for a given query can be assumed to be of a certain shape, and we introduce for that

the notion of *existential extraction*;

- the equivalence between some of our problems and the problem of perfect matching, finally, which we use to obtain a classification of the data complexity of counting query answering at the level of the pair $(\mathcal{R}, q)$ (this is sometimes called "non-uniform data complexity" [60]).

1.1.3 Adding Provenance

Another way to have a look at extending the core framework of ontology-based query answering is to ask what can be provided on top of answers. Examples of such meta-data are explanations on why a tuple is an answer, the probability of a tuple to be an answer, the security clearance required to see a tuple as answer,... Each of these questions can be investigated independently, but there is also a nice theoretical framework developed in the database community that provides an abstraction of such questions: *semiring provenance*. It has been introduced in an influential paper by Green et al [50]. It spawned a large number of follow-ups, including in the ontology-based query answering setting [67, 31, 23]. However, whereas the definition of provenance for conjunctive queries has been quite consensual, the introduced definition for Datalog is more prone to discussion. Indeed, several definitions have been proposed when adapting provenance to other settings, and they happen not to coincide even in the setting of Datalog.

After hoping to study provenance for expressive sets of existential rules (for instance, guarded rules), with a focus on the complexity of computing it, we decided with Camille Bourgaux, Pierre Bourhis and Liat Peterfreund to focus on the actual definition of provenance for Datalog [21]. We investigated the space of definitions, based on three kind of definitions. The first kind is inspired from the original definition of Green et al. [50], based on the notion of derivation trees. But whereas their definition included every possible derivation tree, we considered also the possibility to include in the computation of provenance only a subset of these trees, as some of these trees are "unnatural". Interestingly, most of these alternative definitions can be understood as computing a provenance information based on a particular algorithm, be it the naive, optimized naive or semi-naive evaluation of Datalog. We also considered a third kind of semantics, based the notion of models of provenance annotated knowledge bases. After discovering such a variety of semantics, a very natural question is to *compare* them. We thus defined what can be thought of as rationality properties, that is, properties that can be expected to be fulfilled by a reasonable provenance definition. Perhaps surprisingly, none of the semantics proposed in the literature happen to fulfill all of these properties, and the one definition that we introduced having all of these properties is provably hard to compute.

Defining a consensual semantics is still, in my opinion, an open question. Recent work introduced yet another semantics [25]. Our paper both questions the "standard" status of the original semantics and provides valuable tools to compare provenance definitions, which may be a guide when trying to adopt one of the definitions for a given use case.

1.2 Structure of this Document

This dissertation is structured as follows. We first provide a quick overview of ontology-based query answering, mostly by introducing the *chase*, which is a core tool in the field and is also used for related problems, and review how constraints on the chase ensure decidability of the ontology-based query answering problem. Existential rules (also known as tuple-generating

dependencies), chase, derivations, tree decompositions, description logics, as well as Turing machines and perfect matchings will be introduced in this section. It can easily be skipped by someone already familiar with the field (though Example 2, taken from [34], illustrating the behavior of the restricted chase, might be worth having a look at if not already known). We then go over each of the contributions above mentioned. First, we consider existential rules as an extension of the query language Datalog, and we study the expressivity of such a query language in Section 3. We then adopt a more knowledge representation view, and see existential rules as a way to formalize a domain knowledge: it becomes crucial to allow users to use more than conjunctive queries, and we consider extensions of the core problem of ontology-based query answering in Section 4. We finally discuss the adaptation of the notion of provenance for Datalog in Section 5.

The approach adopted is to provide the reader with the minimal background necessary for understanding each of these sections, and to highlight a few technical tools or core ideas that have proven useful in each of these settings, and which can hopefully be reused in other settings as well. The reader interested in more technical details and/or detailed proofs of the claimed result can consult the associated papers, either in one of the five attached selected publications, or available on HAL, [2] the French repository of scientific publications.

2 Preliminaries

2.1 Ontology Based Query Answering

We consider first-order formulas over countably infinite sets $\mathbf{V}$ of variables and $\mathbf{P}$ of predicates, where each $\mathsf{p} \in \mathbf{P}$ has an arity $\mathsf{Ar}(\mathsf{p}) \in \mathbb{N}$. List of variables are denoted $\vec{x} = x_1, \ldots, x_k$ and will be treated like sets when order is not relevant. An *atom* is an expression $\mathsf{p}(\vec{x})$ with $\mathsf{p} \in \mathbf{P}$ and $|\vec{x}| = \mathsf{Ar}(\mathsf{p})$. An *instance* is a finite set of atoms.

An existential rule $\rho = (B, H)$ is a closed formula of the shape

$$\forall \vec{x} \forall \vec{y} (B(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} H(\vec{y}, \vec{z})),$$

where B and H are conjunctions of atoms, respectively called the body and the head of the rule, and $\vec{x}, \vec{y}$ and $\vec{z}$ are pairwise disjoint. The variables of $\vec{y}$ are called the frontier variables, denoted by $\mathsf{fr}(\rho)$, while the variables of $\vec{z}$ are called the existential variables. A rule is said to be *Datalog* if no variables are existentially quantified in its head.

The ontology based query answering (OBQA) problem asks, given a ruleset $\mathcal{R}$, a database I and a Boolean query q, where $I, \mathcal{R} \models q$, and $\models$ denotes the entailment relation of first-order logic. OBQA is undecidable, and major driving force of the research in the OBQA community has been to define restrictions on the set of rules $\mathcal{R}$ that ensures its decidability. Most of the defined decidable classes rely on properties that can be understood through the so-called *chase* [17], which is the notion we now introduce.

2.2 The Chase

The chase is a semi-decision procedure for OBQA. It is based on the very natural notion of rule application, and of sequences of such rule applications. Several criteria of applicability of a rule can be defined, and this leads to several variants of the chase. The "basic" version of the chase lead to the construction of a *universal model*, which is a model of the database

[2] `https://hal.science/`

I and of the set of existential rules $\mathcal{R}$ that homomorphically maps to any other model of I and $\mathcal{R}$. As such, it can be used to answer homomorphism-closed queries, among which the particular case of Boolean conjunctive queries has been the focus of most research in the field of ontology based query answering. Note that objects that are not universal models could also be used to answer homomorphism-closed queries, and such an object is what is generated by the equivalent chase that we will introduce.

We start the presentation of the chase with the notion of trigger, which is common to every variant.

Definition 1 (Trigger)**.** *Let $\mathcal{R}$ be a set of rules and I be an instance. An $\mathcal{R}$-trigger on I is a pair (ρ, π) such that $\rho \in \mathcal{R}$ and π is a homomorphism from the body of ρ into I. When $\mathcal{R}$ is clear from the context, we will just say a* trigger*.*

A trigger represents the possible applicability of a rule on an instance. The result of that rule application is defined by adding the least specific information to the considered instance in such a way that the head of the rule, instantiated by the image of the frontier, could be mapped to the resulting instance.

Definition 2 (Rule Application)**.** *Let (ρ, π) be a trigger on I. The result of the application of (ρ, π) on I is the instance $\alpha(I, \rho, \pi)$ defined by:*

$$\alpha(I, \rho, \pi) = I \cup \pi^s(H(\rho)),$$

where $\pi^s(x) = \pi(x)$ if $x \in \mathsf{fr}(\rho)$ and is a fresh term (often called a null*) otherwise.*

Applying one rule may not be enough to transform the instance in a model of the ruleset (because either other rules were already applicable from the start, or the new atoms added triggered the applicability of rules in a new way). This naturally leads to the definition of a derivation.

Definition 3 (Derivation)**.** *A derivation $\mathcal{S}$ from I_0 w.r.t. $\mathcal{R}$ is a sequence $(I_0, t_1, I_1, \ldots, I_n)$ such that for any $i \in \{1, \ldots, n\}$, t_i is a(n $\mathcal{R}$-)trigger on I_{i-1} and I_i is the result of applying t_i on I_{i-1}. We denote by $\mathcal{S}_i$ the prefix of $\mathcal{S}$ of length i, that is, the derivation $(I_0, t_1, \ldots, I_i)$.*

Note that this notion of derivation does not capture any kind of non-redundancy check. While this is not problematic with respect to the creation of a universal model, this prevents quite often the existence of finite sequence on the result of which no rule is applicable. For that reason, several notions of trigger activity have been defined, and to each of these notions will be associated a different variant of the chase: o-active triggers for the *oblivious* chase, so-active triggers for the *semi-oblivious* chase, r-active triggers for the *restricted* chase, and e-active triggers for the *equivalent* chase.

Definition 4 (Activity of a Trigger)**.** *A trigger $t = (\rho, \pi)$ is $\star$-active for $\star \in \{\mathsf{o}, \mathsf{so}, \mathsf{r}, \mathsf{e}\}$ w.r.t. to a derivation $\mathcal{S} = (I_0, t_1, I_1, \ldots, I_n)$ if:*

- *$\star = \mathsf{o}$ and there is no $i \in \{1, \ldots, n\}$ such that $\rho = \rho_i$ and $\pi = \pi_i$;*
- *$\star = \mathsf{so}$ and there is no $i \in \{1, \ldots, n\}$ such that $\rho = \rho_i$ and $\pi_{|\mathsf{fr}(\rho)} = \pi_{i|\mathsf{fr}(\rho)}$;*
- *$\star = \mathsf{r}$ and there is no homomorphism from $\pi^s(H(\rho))$ into I_n being the identity on $\pi(\mathsf{fr}(\rho))$;*
- *$\star = \mathsf{e}$, and there is no homomorphism from $\alpha(I_n, \rho, \pi)$ into I_n.*

Note that there is a hierarchy between trigger activities: any e-active trigger is also r-active, any r-active trigger is also so-active, and any so-active trigger is also o-active. Moreover, this hierarchy is strict.

We conclude by the definition of a $\star$-chase sequence. On top of enforcing that each trigger should be $\star$-active, it is also required that a chase sequence is fair, which means that a trigger cannot stay active infinitely long.

Definition 5 ($\star$-chase sequence)**.** *A $\star$-chase sequence is a* fair $\star$-derivation*, that is, a derivation such that for any i, (ρ_i, π_i) is a $\star$-active trigger w.r.t. $\mathcal{S}_{i-1}$, and for any j, for any trigger (ρ, π) that is $\star$-active w.r.t. $\mathcal{S}_j$, there exists $k > j$ such that (ρ, π) is not $\star$-active w.r.t. $\mathcal{S}_k$.*

A very basic theorem regarding ontology based query answering is the soundness and completeness of derivations w.r.t. query answering.

Theorem 1. *Let I be an instance, $\mathcal{R}$ be a set of existential rules and q be a conjunctive query. It holds that $I, \mathcal{R} \models q$ if and only if there exists a finite derivation whose result I^* is such that $I^* \models q$.*

As such, a specific case is very interesting: when a $\star$-chase sequence happens to be finite. Deciding the termination of the chase has been an important task, leading to numerous results on the frontier of the decidability of that problem (see [49] for an overview). We point out that the problem comes in several flavours:

- we may be interested in the termination of the chase on a given instance, or a universal criteria of termination of the chase.
- we may be interested in the termination of any chase-sequence, or in the existence of a terminating chase sequence.

This leads to the definition of several problems, whose naming convention is as follows:

$$\mathbb{CT}^{\star}_{\star_1\star_2},$$

where: $\star \in \{\mathsf{o}, \mathsf{so}, \mathsf{r}, \mathsf{e}\}$, $\star_1 \in \{I, \forall\}$ and $\star_2 \in \{\exists, \forall\}$. $\star_1$ describes if we are interested in an instance dependent termination, or a termination that holds for any instance, and $\star_2$ describes if we want the existence of a terminating chase sequence, or termination for every chase sequence. Note that $\mathbb{CT}^{\star}_{\star_1\forall}$ and $\mathbb{CT}^{\star}_{\star_1\exists}$ happen to coincide for every $\star \in \{\mathsf{o}, \mathsf{so}, \mathsf{e}\}$ and $\star_1 \in \{I, \forall\}$. However, the restricted chase exhibits slightly different behaviors, as is witnessed by Example 1.

Example 1. *Let us consider the following rules:*

- $r(x, y) \rightarrow \exists z\ r(y, z)$
- $r(x, y) \rightarrow p(y, y)$
- $p(x, x) \rightarrow r(x, x)$

Let $I = \{r(a, b)\}$. There exists a terminating r*-chase sequence, that starts by applying the second rule, then the third one. No more triggers are active. However, there exists as well an infinite* r*-chase sequence, which is obtained by first applying the first rule, then the second, then the third, and repeating the process on the atom generated by the first rule.*

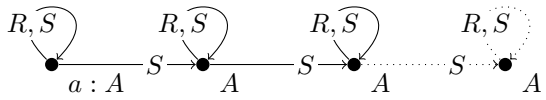

Figure 1: The only result of the Datalog-first restricted chase from the KB $(\mathcal{R}, \{A(a)\})$ introduced Example 2

As terminating restricted chase is a main focus of one of the contributions presented afterwards, let us point out a result obtained with David Carral, Lucas Larroque and Marie-Laure Mugnier [34], which illustrates well how counter-intuitively the restricted chase may behave. A very natural way of choosing which trigger to apply next is to apply first the triggers that do not generate new existentially quantified variables, as these triggers must be applied at some point in any restricted chase sequence. This leads to the notion of Datalog-first chase, and this has been the terminating strategy in Example 1.

Definition 6. *An* r*-derivation $\mathcal{S} = (I_0, t_1, I_1, \ldots, I_n)$ is said Datalog-first if for any $i \in \{1, \ldots, n\}$, if $t_i = (\rho_i, \pi_i)$ is such that ρ_i is not a Datalog rule, then no trigger (ρ, π) such that ρ is Datalog is active on $\mathcal{S}_{i-1}$.*

An intuition that has not been formalized but was behind some technical developments (such as restricted model faithful acyclicity [32]) is that if there exists a terminating restricted chase sequence, then there must exist a terminating Datalog-first restricted chase sequence. Quite surprisingly, this happens to be false, as witnessed by the following example, taken from [34].

Example 2. *Let us consider the instance $I = \{A(a)\}$, and the rules*

$$A(x) \to R(x, x) \tag{1}$$
$$R(x, y) \wedge S(y, z) \to S(x, x) \tag{2}$$
$$A(x) \wedge S(x, y) \to A(y) \tag{3}$$
$$A(x) \to \exists z\ R(x, z) \tag{4}$$
$$R(x, y) \to \exists z\ S(y, z) \tag{5}$$

2.2.1 Derivation of Bounded Treewidth

While the above presentation mostly focused on the case where the chase terminates, non-termination of the chase is not implying undecidability of the Boolean conjunctive query answering problem. Weaker properties may also ensure its decidability, and one such properties is that the instances generated by any derivation are of treewidth that is bounded by a function of the input database (and that of the set of rules). Let us first recall the notion of treewidth, which is a graph parameter that intuitively describes "how far" a graph is from being a tree. It is defined using the notion of tree decompositions.

Definition 7 (Tree decomposition)**.** *A tree decomposition of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with $\{X_i \mid i \in I\}$ a collection of subsets of V, called* bags*, and $T = (I, F)$ a tree such that:*

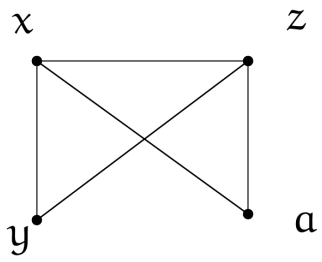

Figure 2: The Gaifman graph of $\exists x \exists y \exists z\ r(x, y) \wedge p(x, z, a) \wedge r(y, z)$

- *for all $v \in V$, there exists $i \in I$ with $v \in X_i$*
- *for all $\{v, w\} \in E$, there exists $i \in I$ such that $\{v, w\} \subseteq X_i$*
- *for all $v \in V$, the set $I_v = \{i \in I \mid v \in X_i\}$ forms a connected subtree of T.*

The width of a tree decomposition is defined by $\mathsf{max}_{i \in I} |X_i| - 1$, and the treewidth of a graph is the minimum width of one of its tree decomposition. To apply that graph notion to instances, we define the Gaifman (or primal) graph of an instance.

Definition 8 (Gaifman Graph of an Instance)**.** *Let I be an instance. The primal graph $G_I = (V_I, E_I)$ of I is defined as follows:*

- *V_I is a set of vertices in bijection with $\mathsf{term}(I)$*
- *E_I contains an edge between two vertices of V_I if and only if there is an atom of I in which the associated terms both appear.*

An example of Gaifman graph is presented in Figure 2.

Based on this notion of treewidth, it is possible to define a property of a ruleset: performing derivations starting from a given instance generates instances whose treewidth remains bounded.

Definition 9 (Bounded Treewidth Set)**.** *A ruleset $\mathcal{R}$ is a bounded treewidth set if for any instance I, there exists b such that for any derivation from I w.r.t. $\mathcal{R}$ resulting in I_n is such that $\mathsf{tw}(I_n) \leq b$.*

Other variants of the notion of bounded treewidth sets have been introduced, such as for instance this semantic bounded treewidth property.

Definition 10 (Semantic Bounded Teewidth)**.** *A ruleset $\mathcal{R}$ is a semantically bounded treewidth set if for any instance I, there exists b such that there exists a universal model I^* of I and $\mathcal{R}$ such that $\mathsf{tw}(I^*) \leq b$.*

The reader interested in the relationships between such abstract classes can consult [13]. In the literature have been defined several classes of rules that have the bounded treewidth property, among which guarded rules [26], inspired by the guarded fragment [4], are a prototypical example.

Definition 11 (Guarded Rules)**.** *An existential rule ρ is said guarded if there exists $\alpha \in B(\rho)$ such that $\mathsf{var}(B(\rho)) \subseteq \mathsf{var}(\alpha)$. A set of rules $\mathcal{R}$ is guarded if every rule of $\mathcal{R}$ is guarded.*

The class of frontier-guarded rules (where only frontier variables are guarded) has also this property. Both classes are fragments of the guarded negation fragment of first order logic (GNFO) [15], and shows that frontier-guarded rules are exactly the set of existential rules that can be expressed within GNFO [14]. Other class of rules ensuring the bounded treewidth property include generalizations of guarded rules, where not all body terms have to be guarded, but only subpart of them.

2.2.2 Bounded Derivation Depth Property

Another important property of rulesets guarantees the decidability of Boolean conjunctive query answering: the *bounded derivation depth property.*

Definition 12 (Bounded Derivation Depth Property)**.** *A set of rules has the bounded derivation depth property if for every query q, there exists k such that for any database I, it holds that*

$$\mathsf{chase}_k(I, \mathcal{R}) \models q \Leftrightarrow \mathsf{chase}(I, \mathcal{R}) \models q.$$

A class of rules known to have the bounded derivation depth property is that of *linear* rules [27].

Definition 13 (Linear Rules)**.** *A rule is* linear *if its body contains a single atom.*

Another way to look at the bounded derivation depth property is to notice that it is equivalent to the existence of a rewriting of q into a union of conjunctive queries q' such that

$$\forall I, I, \mathcal{R} \models q \Leftrightarrow I \models q'.$$

How such a rewriting can be obtained in practice is not necessary to understand the content of this manuscript, and we refer the interested reader to [64] for an introduction to ontology based query answering that present such a rewriting algorithm.

2.3 Description Logics

Description Logics are a major formalism for representing ontologies, and the interested reader can consult [8].

We use mutually disjoint sets N_C, N_R and $\mathbf{I}$ of respectively concept, role and individual names. A knowledge base is a pair $(\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An ABox is a finite set of concept assertions $A(b)$, with $A \in N_C$ and $b \in \mathbf{I}$, and role assertions $P(a, b)$, with $P \in N_R$ and $a, b \in \mathbf{I}$. We denote by $\mathsf{ind}(\mathcal{A})$ the set of individual names occurring in an ABox $\mathcal{A}$. A TBox is a finite set of axioms, whose from depends on the considered description logic. We introduce the axioms allowed in the $\mathcal{ALCHI}$ description logic, and say which subsets of these constructs are allowed for various sublogic of relevance in this dissertation. Axioms are either positive concept inclusions, negative concept inclusions, positive role inclusions and negative role inclusions. In this dissertation, role inclusions are always done with roles from the set $N_R^{\pm} = \{P, P^- \mid P \in N_R\}$, whereas concept inclusions may involve concepts B constructed by a subset of the following grammar:

		DL-Lite				Suffixes			Examples	
		pos	core	$\mathcal{EL}$	$\mathcal{ALC}$	$\mathcal{H}$	$\mathcal{I}$	$\bot$	$\mathcal{ALCI}$	$\mathcal{ELH}_\bot$
Concepts	$A \mid \exists R.\top$	✓	✓	✓	✓				✓	✓
	$\top \mid B_1 \sqcap B_2 \mid \exists R.B$			✓	✓				✓	✓
	$\bot$				✓			✓	✓	✓
	$\neg B \mid B_1 \sqcup B_2 \mid \forall R.B$				✓				✓	
Roles	$R \in \mathsf{N_R}$	✓	✓	✓	✓		✓		✓	✓
	$R \in \mathsf{N_R^\pm}$	✓	✓				✓		✓	
Axioms	Positive concept incl.	✓	✓	✓	✓				✓	✓
	Positive role incl.					✓				✓
	Corresp. neg. incl.		✓		✓			✓	✓	✓

Figure 3: The naming convention of some Description Logics

$$B := \top \mid \bot \mid A \mid \neg B \mid B_1 \sqcap B_2 \mid B_1 \sqcup B_2 \mid \exists R.B \mid \forall R.B, \text{ with } A \in N_C, R \in N_R^\pm$$

The naming convention of a specific description logic is summarized in Figure 3.

The semantics of of constructors is classically defined, and recalled in Figure 4. Classically, an interpretation $\mathcal{I}$ is a model of a TBox axiom $G \sqsubseteq H$ if $G^\mathcal{I} \subseteq H^\mathcal{I}$. It is a model of an ABox assertion $A(a)$ (resp. $P(a,b)$) if $a \in A^\mathcal{I}$ (resp. $(a,b) \in P^\mathcal{I}$). It is a model of a TBox if it is a model of all its axioms, and a model of an ABox if it is a model of all its assertions.

Note that there is huge variety in the description logics above detailed: the members of the DL-Lite family are relatively inexpressive description logics that have been specifically tailored for allowing conjunctive query answering using a rewriting approach [29]. The $\mathcal{EL}$ is another Horn description logic, introduced in [7]. It benefits from the bounded treewidth property.

$\mathcal{ALC}$ is however a much more expressive description logics, which has the particularity (among those introduced here) to not admit a universal model. This makes very basic problems, such as the satisfiability of a concept, more complex (at least from a conceptual point of view) to solve [8].

2.4 Some Folklore Notions

The reader is assumed to be familiar with Turing machines, in deterministic, non-deterministic or alternating fashion. A good reference for these notions (or for the complexity class that will be mentioned in this manuscript) is Sipser's book [73].

Constructor	Syntax	Interpretation
Inverse role	P^-	$\{(y,x) \mid (x,y) \in \mathrm{P}^{\mathcal{I}}\}$
Bottom	$\bot$	$\emptyset$
Top	$\top$	$\Delta^{\mathcal{I}}$
Negation	$\neg\mathrm{B}$	$\Delta^{\mathcal{I}} \setminus \mathrm{B}^{\mathcal{I}}$
Conjunction	$\mathrm{B}_1 \sqcap \mathrm{B}_2$	$\mathrm{B}_1^{\mathcal{I}} \cap \mathrm{B}_2^{\mathcal{I}}$
Disjunction	$\mathrm{B}_1 \sqcup \mathrm{B}_2$	$\mathrm{B}_1^{\mathcal{I}} \cup \mathrm{B}_2^{\mathcal{I}}$
Existential restriction	$\exists\mathrm{R}.\mathrm{B}$	$\{d \mid \exists e \in \Delta^{\mathcal{I}}, (d,e) \in \mathrm{R}^{\mathcal{I}} \wedge e \in \mathrm{B}^{\mathcal{I}}\}$
Universal restriction	$\forall\mathrm{R}.\mathrm{B}$	$\{d \mid \forall e \in \Delta^{\mathcal{I}}, (d,e) \in \mathrm{R}^{\mathcal{I}} \rightarrow e \in \mathrm{B}^{\mathcal{I}}\}$

Figure 4: The semantics of the introduced constructors

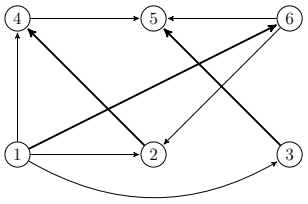

Figure 5: A graph G_1 and a perfect matching, in bold.

Another graph notion that is used at several places in this manuscript is the notion of perfect matchings. A *directed finite graph* (in the following, just *graph*) is a pair $G = (V, E)$ where V is a finite set of *vertices* and $E \subseteq V \times V$ is the set of edges. For any $(v, v') = e \in E$, the vertices v and v' are called the *ends* of e. Two edges are said adjacent if they share an end. A *matching* for G is a set M of pairwise non-adjacent edges. A *perfect matching* for G is a matching M such that every vertex of V belongs to an edge of M. It is well-known (but not trivial) that the existence of a perfect matching in a graph can be checked in polynomial time [37]. Matchings can also (and may be more classically) defined on non-directed graphs.

Figures 5 and 6 illustrate the notion of perfect matching.

3 Expressivity of Existential Rules as a Query Language

In this section, we study the expressivity of existential rules as a query language. As is classical in this setting, we fix a set of extensional predicates $\mathbf{P}_{\mathcal{E}}$ which are used to build the instances we query, and a set of intensional predicates $\mathbf{P}_{\mathcal{I}}$, which may not appear in queried instances and are the only appearing in rule heads. We study which sets of databases can be characterized by an existential rule query, defined as follows.

Definition 14 (Existential Rule Query)**.** *An* existential rule *query on databases on vocabulary $\mathbf{P}_{\mathcal{E}}$ is a set $\mathcal{R}$ of existential rules, whose body predicates belong to $\mathbf{P}_{\mathcal{E}} \cup \mathbf{P}_{\mathcal{I}}$ and*

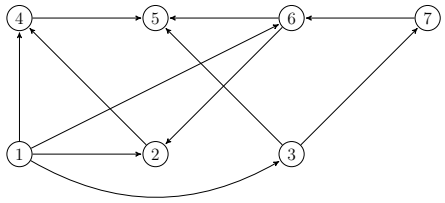

Figure 6: A graph G_2 without perfect matching. Note that G_1 homomorphically maps to G_2.

head predicates belong to $\mathbf{P}_\mathcal{I}$*, with a distinguished nullary predicate* $\mathbf{goal} \in \mathbf{P}_\mathcal{I}$ *and with* $\mathbf{P}_\mathcal{I}$ *disjoint from* $\mathbf{P}_\mathcal{E}$*. The query defined by an existential rule query* $\mathcal{R}$ *is the set of databases over* $\mathbf{P}_\mathcal{E}$ *defined by:*

$$\{I \mid I, \mathcal{R} \models \mathbf{goal}\}.$$

Note that a similar definition can be provided for Datalog (or extensions thereof, as explained later). Note also that this view is also rather databasy, whereas an alternative definition could be specified by mapping, for each set of existential rules $\mathcal{R}$, a database I to the set of queries q such that $I, \mathcal{R} \models q$. Such a notion has been studied by Zhang et al. [79].

In this section, we will discuss the expressivity of Datalog and of several of its extensions. Two important features that have been considered is the ability to use negation for the database predicates (also referred to as input negation, resulting in so-called semipositive Datalog) and the availability of a linear order on the domain elements (giving rise to order-invariant Datalog). Expressivity of a logical formalism can be studied in a relative fashion or an absolute fashion. The relative way is to compare the expressivity of one query language $\mathcal{Q}_1$ with respect to anther query language $\mathcal{Q}_2$, and to prove that any query of $\mathcal{Q}_1$ can be equivalently expressed by a query of $\mathcal{Q}_2$. Another way of studying the expressivity of a query language is to relate the queries in a given query language to external measures of complexity and expressivity. This helps in turn to establish non-expressibility results. A simple example of such corollaries is to notice that queries expressible in Datalog have polynomial complexity. Hence, an existential rule query whose complexity is exponential cannot be expressed in Datalog (which shows for instance, that some queries expressible with weakly-acyclic [38] rulesets are not expressible in Datalog, because it is ExpTime-complete in data complexity to answer conjunctive queries under such rules).

The expressivity of the extensions of Datalog have attracted a lot of investigations. After formalizing the extensions of Datalog that we will consider in this section, we recall the main results on the expressivity of these query languages.

3.1 Background

Definition 15 (Semipositive Datalog)**.** *A semipositive Datalog program is a set of first-order logic formulae of the shape*

$$\forall \vec{x}\ \forall \vec{y} B[\vec{x}, \vec{y}] \rightarrow p(\vec{y}),$$

where B *is a conjunction of:*

- *extensional atoms of the form* $\mathtt{p}(\vec{z})$ *with* $\mathtt{p} \in \mathbf{P}_\mathcal{E}$*, and* $\vec{z} \subseteq \vec{x} \cup \vec{y}$

- *negated extensional atoms of the form* $\neg \mathtt{p}(\vec{z})$ *with* $\mathtt{p} \in \mathbf{P}_{\mathcal{E}}$ *and* $\vec{z} \subseteq \vec{x} \cup \vec{y}$ *and*
- *intensional atoms of the from* $\mathtt{p}(\vec{z})$ *with* $\mathtt{p} \in \mathbf{P}_{\mathcal{I}}$.

A semipositive Datalog query is a semipositive Datalog program containing a special nullary predicate **goal**. If $\mathcal{R}$ is a semipositive Datalog query, the $\mathbf{P}_{\mathcal{E}}$-instance I with domain Δ belongs to $q_{\mathcal{R}}$ if and only if $\mathsf{comp}(I) \cup \mathcal{R} \models \mathbf{goal}$ according to the first-order logic semantics, where $\mathsf{comp}(I) = I \cup \{\neg \mathtt{p}(\vec{a}) \mid \mathtt{p}(\vec{a}) \notin I, \mathtt{p} \in \mathbf{P}_{\mathcal{E}}, \vec{a} \in \Delta^n\}$. Semipositive Datalog programs (queries) without negated atoms are called Datalog programs (queries).

Give a $\mathbf{P}_{\mathcal{E}}$-database I over a domain Δ and a linear order $\leq$ over Δ, we define the $\mathbf{P}_{\mathcal{E}} \cup \{\mathsf{first}, \mathsf{last}, \mathsf{succ}\}$-database $I^{<}$ as I extended by the atoms:

- $\mathsf{first}(a)$ for the $<$-minimal element a of Δ
- $\mathsf{last}(b)$ for the $<$-maximal element b of Δ
- $\mathsf{succ}(c, d)$ for any two $<$-consecutive elements c and d of Δ, that is, $c < d$ and for no $e \in \Delta \setminus \{c, d\}$, it holds that $c < e < d$.

An order invariant Datalog query making use of $\mathsf{first}, \mathsf{last}, \mathsf{succ}$ (besides predicates from $\mathbf{P}_{\mathcal{E}}$ and $\mathbf{P}_{\mathcal{I}}$) whose result is independent from the particular choice of the linear order $\leq$. We then let D belong to the defined query if $D^{\leq}$ belongs to it, for some (or equivalently, every) linear order $\leq$. These two features leads to four distinct query languages, denoted by Datalog, Datalog$^{\neg}$, Datalog$^{<}$ and Datalog$^{\neg,<}$.

A way of understanding the relationships between these query languages is to study their closure properties, and especially revelant is that of homomorphism-closedness.

Definition 16 (Homomorphism-Closedness)**.** *A query q is said to be homomorphism-closed, if for any instance I, I', if $I \in q$ and there is a homomorphism from I to I', then $I' \in q$.*

A natural question is to establish links (and hopefully characterizations) of the set of queries that can be expressed through a syntactic restriction of the query language. Semipositive Datalog can obviously express queries that are not homomorphism-closed (checking if a graph is not symmetric, that is, if there exists two individuals a and b such that $E(a, b)$ holds but not $E(b, a)$), but does not extend enough the power of Datalog to express all polynomial time queries. Feder and Vardi [39] showed than when restricted to homomorphism-closed queries, both Datalog$^{\neg}$ and Datalog have the same expressivity. However, Dawar and Kreutzer [36] showed that there are some homomorphism-closed queries that are not expressible in Datalog. With Sebastian Rudolph, we completed the systematic exploration of the space of extensions of Datalog with respect to input negation and linear order [71]. In particular is answered the question of whether input negation may be necessary to express homomorphism-closed queries, that is, whether all PTime homomorphism-closed queries are expressible in Datalog$^{<}$. While this paper proposes a completed Venn diagram (Figure 7) of the class defined by homomorphism-closedness, polynomiality, and the four considered variants of Datalog, the expressivity of Datalog remains not fully understood. Seemingly very basic questions remain unsolved, such as the possibility to express the query checking the existence of path of squared length from a source to a target:[3] the best known upper bound comes from the paper of Dawar and Kreuzer, relying on an intricate pumping lemma, and that provides a double exponential lower bound.

The rest of this section is devoted to a high-level presentation of the technical tools used in these results.

[3]Thanks to Jerzy Marcinkowski for raising this open problem.

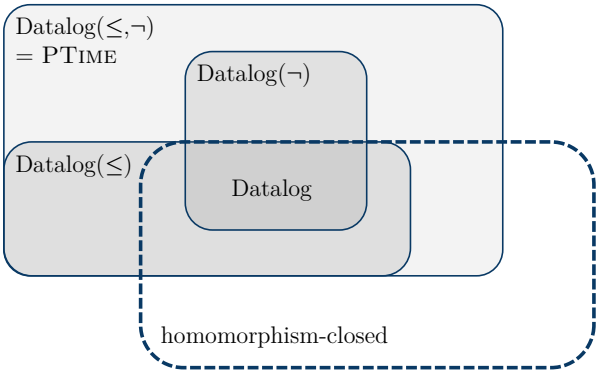

Figure 7: Relationships between the query classes.

3.2 Input Negation is Necessary even for Homomorphism-Closed Queries

The main technical result of [71] is to prove the existence of a query that is both polynomial and homomorphism-closed, but that it is not expressible in Datalog$^<$.

We exhibit such a query, taking inspiration from the perfect matching problem. As mentioned in the preliminaries, this problem is known to be polynomially solvable. It is however not homomorphism closed, as can be seen through the examples of Figures 5 and 6. We thus define a query that is slightly different (the intuition behind the construction will be outlined below), but whose expressivity in Datalog$^<$ would yield a polynomial *monotone circuit* solving the perfect matching problem, which is known to be impossible by a result from Razborov [68]. Let us first recall some notions from circuit complexity.

Circuit Complexity A *k-ary Boolean function* (for $k \geq 1$) is a function $f : \{0,1\}^k \to \{0,1\}$. A *k-input Boolean circuit* $\mathcal{C}$ (for $k \geq 1$) is a directed acyclic graph with k distinguished nodes (labeled from 0 to $k-1$), called *sources (inputs)* which have no incoming edges, and with one distinguished node called *sink (output)* which has no outgoing edges. Every non-source node of $\mathcal{C}$ is called a *gate*; it is labelled with either one of $\wedge, \vee$, in which case is has two incoming edges, or with $\neg$, in which case there is one incoming edge. Given a vector $\mathbf{x} \in \{0,1\}^k$, the *value of source* i on $\mathbf{x}$ is the i^{th} bit of $\mathbf{x}$. The *value of a non-source gate* x labeled by $\wedge$ (resp. $\vee$, resp. $\neg$) is the conjunction of the values of its incoming gates (resp. disjunction, resp. negation). The value of the sink on $\mathbf{x}$ is denoted by $\mathcal{C}(\mathbf{x})$.The number of nodes in $\mathcal{C}$ is its size, denoted by $|\mathcal{C}|$. $\mathcal{C}$ is said to compute a Boolean function f if for any tuple $\mathbf{x} \in \{0,1\}^k$, $\mathcal{C}(\mathbf{x}) = f(\mathbf{x})$. A Boolean circuit is *monotone* if no gate is labeled with $\neg$. Let us last note that we will consider circuit with arbitrary many ingoing edges for $\wedge$ and $\vee$ gates (straightforwardly generalizing the definition). As long as we are interested in circuits of polynomial size, this is not a problem as for every such circuit there is a circuit with gates $\wedge$ and $\vee$ having two ingoing edges and of polynomial size with respect to the original circuit.

A family of circuits for the perfect matching problem is a sequence of circuits $(\mathcal{C}_i)$ where $\mathcal{C}_i$ has i^2 inputs, $0, \ldots, i^2 - 1$, such that the output of $\mathcal{C}_i$ is 1 if and only if its input is the representation of a graph with i vertices that contains a perfect matching, and 0 otherwise. It is known [68] that there exists no family of monotone Boolean circuits of polynomial size

computing the perfect matching function.

Theorem 2 ([68]). *There exists a constant $c > 0$ such that the monotone circuit complexity of the perfect matching function is greater than $n^{c \cdot \log(n)}$.*

We build a query on extensional predicates $\{\mathsf{first}, \mathsf{next}, \mathsf{last}, \mathsf{edge}\}$. Intuitively, a database instance is "well-behaved" if the atoms of predicate first, next, last encode a linear order of the domain elements of that database. On well-behaved databases, the built query $\mathtt{q}$ accepts exactly the databases encoding a graph admitting a perfect matching. We must define the behavior of $\mathtt{q}$ on non well-behaved databases, and we proceed as follows:

- $\mathtt{q}$ does not match a database I that does not contain a next-connected component containing a first-individual and a last-individual;
- the query matches databases which (even after removal of all elements not contained in any first and last containing next-connected component) cannot be mapped into any structure where first, last and next constitute a linear order;
- any non-well behaved database I not falling in any of the two previous categories can be turned into a well-behaved database I' in a deterministic fashion (by merging domain elements and removing unconnected elements) with the following two properties: *(i)* any well-behaved database receiving a homomorphism from I also receives a homomorphism from I'; *(ii)* any well-behaved database having a homomorphism into I also has a homomorphism into I'. Therefore, we can safely define $\mathtt{q}$ as matching I if and only if $\mathtt{q}$ matches I', that is, if and only if I' encodes a graph that contains a perfect matching.

The contradiction with Theorem 2 is obtained as follows: assume that we are given an order-invariant Datalog program $\mathbb{P}$ computing $\mathtt{q}$. Given this program and a natural number n, we build a polynomial-size monotone Boolean circuit over n^2 input variables deciding the existence of a perfect matching in a graph with n vertices. Hence, $\mathtt{q}$ is not expressible in order-invariant Datalog. Note that a very similar proof technique has been used by Ketsman and Koch [55] while studying monotone queries expressible in extensions of Datalog.

3.3 Terminating Restricted-Chase

We now turn to the other extremity of the spectrum, considering existential rules. Some upper bounds for the expressivity of existential rules as a query languages are quite easy to obtain. Indeed, the chase provides us with a semi-decidability procedure, ensuring that any existential rule query is recursively enumerable. Moreover, it is quite clear that such queries are closed under homomorphism, again by analyzing the chase. Proving that indeed any recursively enumerable homomorphism-closed query is expressible by an existential rule query has been the topic of a collaboration with Sebastian Rudolph [70]. An unfortunate (from a theoretical point of view) property of the construction built in this setting is that, even in the case where the considered query is known to be decidable, the set of rules that we build to express it does not fall in any known class for which we have a decidability algorithm. This raises the question of whether we can obtain better behaved construction in this (slightly) restricted setting. This has been studied with Camille Bourgaux, David Carral, Markus Krötzsch and Sebastian Rudolph, with whom we show that for any decidable homomorphism-closed query, there exists an existential rule query expressing it for

which the restricted chase is universally terminating. This is much better, but there is still a short-coming here, as membership of a ruleset to the class of rules whose restricted chase is universally terminating is undecidable. However, we also show that this is unavoidable, as any query language having the same expressivity is doomed to have undecidable membership.

On a technical point of view, the proof follows the lines of the Immerman-Vardi theorem, but major adaptations are required in both cases, making use of a variety of techniques to overcome the absence of linear order and input negation. The rest of this section is structured as follows: Section 3.3.1 provides a high-level view of the proof of the Immerman-Vardi theorem, as both results presented in this section follow the same approach; Section 3.3.2 proposes two ways to simulate input negation and the availability of a linear order, one through the use of existentials (which has been used in [70]) and one through the use of disjunction (the approach used in [22]); Section 3.3.3 introduce the use of the chasing set technique, introduced by Carral et al. [33], in order to simulate disjunction with existential rules while keeping universal restricted chase termination.

3.3.1 A Recall on Immerman-Vardi Theorem

Let us first recall in a high-level fashion the proof of the Immerman-Vardi theorem. The assumption is that there exists a polynomial Turing machine deciding the considered query, and such a Turing machine is thus the starting point of the proof. The key idea is the following: define and build (through Datalog rules) a relational representation of the tape representation of the input database, then simulate the polynomial Turing machine through other Datalog rules. We first introduce a relational representation of the tape, then explain how to build such a representation. The actual simulation of the Turing machine through Datalog rules being not relevant for the remaining of this section, we skip this somewhat technical part.

Tape Representation Our assumption is that the query of interest is expressible in PTime. This means that there exists a polynomial Turing machine that takes as input a representation of an input database I on $\mathbf{P}_{\mathcal{E}}$ and decides whether I belongs to the query or not. We take as encoding of a database an enumeration of the tuples present in the database. For instance, let us consider a database having two individuals a and b, and two atoms, $r(a, b)$ and $p(a, a)$. A representation of that database would be:

$$\#p1r0p0r1p0r0p0r0$$

The first atom $p1$ encodes the fact that $p(a, a)$ belongs to the encoded database, while the first $r0$ encodes the fact that $r(a, a)$ does not belong to the database.

However, Datalog works on relational databases, and what is manipulated by the Datalog program is a relational representation of that tape (and of the subsequent configurations that are reached by the Turing machine). To that purpose, predicates of polynomial arity are used: a set of k positions will be used to represent timestamps (up to the size of the database to the power of k), a set of k positions will be used to represent cell addresses (up to the size of the database to the poser of k), and a constant number of additional positions will be used to represent the actual content of the tape (position of the head, content of the cell).

Building the Relation Encoding of the Tape To build a Datalog program simulating the run of the Turing machine on the tape encoding the database, two challenges are to be overcome: building, from the database, the above described relational representation of the tape of the encoding of that database, and simulating the Turing machine on that relational representation. The second step is mainly routine, and we introduce in a bit more details the first step, which is the one which requires significant work to be adapted in the settings we consider in this dissertation.

As explained above, we have to add tuples representing, for each atom that can be built using extensional predicates and terms from the database, whether that atom is present in the database. Intuitively, the Datalog program is going to iterate over the tuples, and to check for the presence or absence of the corresponding atom. The iteration is made by creating a linear order on k-uples from the linear order on database individuals. For each of these atoms, one add the corresponding representation by considering the case where the tuple is present (using a Datalog rule that does not contain any negated atom) and the case where the tuple is not present (using a Datalog rule containing a negated atom). In both cases, it is noted that a given tuple for a given predicate has been processed, which allows to process the next tuple.

3.3.2 Replacing Ordering and Input Negation

For the cases we are considering (that is, recursively enumerable or decidable queries that are closed under homomorphism), the following challenges should be tackled to adapt the previous reasoning:

- the Turing machines we consider are not polynomial anymore, and no bound on their running time may be assumed. This imply that we cannot use a representation that is based on a timestamp and a cell address; this, however, is a difficulty that has been solved in the literature numerous times (see for instance [12]), and we will not present it here;
- input negation and linear order are not available anymore; we propose two ways to overcome this limitation, one through the use of existential variables, the other through the use of disjunction.

As recalled in the previous section, both linear order and input negation are used in order to create the relational representation of the tape encoding the input database. None of these ingredients are available in the setting of existential rules (even without terminating restricted chase). An idea that has been introduced by Sebastian in [70] is to use existentials to create "faulty" orders (called D-lists), possibly with missing elements and with repetitions. This has been done with the following set of rules, in which the ACDom predicate is to be understood as a unary predicate holding for any individual in the active domain.

- $\mathsf{ACDom}(x) \rightarrow \exists y\ \mathsf{link}(x, y) \wedge \mathsf{first}(y) \wedge \mathsf{last}(y)$
- $\mathsf{ACDom}(x) \rightarrow \exists y\ \mathsf{link}(x, y) \wedge \mathsf{first}(y) \wedge \mathsf{partial}(y)$
- $\mathsf{ACDom}(x) \wedge \mathsf{partial}(y) \rightarrow \exists z\ \mathsf{succ}(y, z) \wedge \mathsf{link}(x, z) \wedge \mathsf{partial}(z)$
- $\mathsf{ACDom}(x) \wedge \mathsf{partial}(y) \rightarrow \exists z\ \mathsf{succ}(y, z) \wedge \mathsf{link}(x, z) \wedge \mathsf{last}(z)$

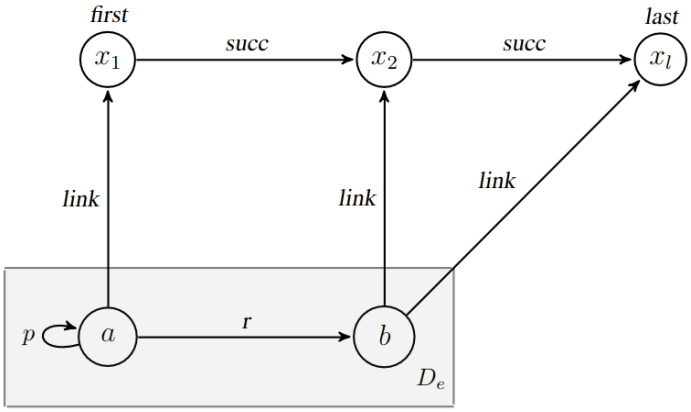

Figure 8: (Partial) Effect of the list-annotator on the input database

The first rule states that for every element of the database x, there is a D list whose first element is x and last element is x. The second rule starts the creation of a longer D-list, which starts with x, but is not complete yet (hence the atom $\mathsf{partial}(y)$). The third and fourth rule allow to add elements to a partial D-list, either by adding another non final element (third rule), or by adding a last element to the current D-list (fourth rule). An example of D-list obtainable from our sample database is presented Figure 8: it is the D-list (a, b, b).

Obviously, many incorrect orderings are created by such an approach: some elements may not be present in a D-list, while some other elements may be repeated (as is the case Figure 8). Most of the technical contribution of [70] is to show that these incorrect orderings are not problematic, that is, they cannot lead to an unwanted acceptation of a database by the built existential rule query. This relies on homomorphism-closedness of the computed query.

Another problem with such an approach occurs when the termination of the chase matters, as is the case in [22], where we aim at expressing decidable queries through rulesets having universally terminating restricted chase. Indeed, the above way of creating D-lists clearly does not terminate, even for an e-chase. A workaround has been to use *disjunction*, temporarily getting out of the scope of existential rules to delve into disjunctive existential rules. Disjunction is used both at the level of the linear order creation and for simulating input negation. For the linear order simulation, it is "guessed" by the following set of disjunctive Datalog rules (and two existential rules of empty frontier).

$$\rightarrow \exists y\ \mathsf{first}(y) \wedge \mathsf{ACDom}(y) \tag{6}$$
$$\rightarrow \exists z\ \mathsf{last}(z) \wedge \mathsf{ACDom}(z) \tag{7}$$
$$\mathsf{ACDom}(x) \rightarrow \mathtt{Eq}(x, x) \tag{8}$$
$$\mathtt{Eq}(x, y) \rightarrow \mathtt{Eq}(y, w) \tag{9}$$
$$\mathtt{NEq}(x, y) \rightarrow \mathtt{NEq}(y, x) \tag{10}$$
$$\mathsf{ACDom}(x) \wedge \mathsf{ACDom}(y) \rightarrow \mathtt{Eq}(x, y) \vee \mathtt{NEq}(x, y) \tag{11}$$
$$\mathtt{NEq}(x, y) \rightarrow \mathtt{LT}(x, y) \vee \mathtt{LT}(y, x) \tag{12}$$
$$\mathtt{LT}(x, y) \wedge \mathtt{LT}(y, z) \rightarrow \mathtt{LT}(x, z) \tag{13}$$
$$\mathsf{first}(x) \wedge \mathtt{NEq}(x, y) \rightarrow \mathtt{LT}(x, y) \tag{14}$$
$$\mathsf{last}(y) \wedge \mathtt{NEq}(x, y) \rightarrow \mathtt{LT}(x, y) \tag{15}$$

Here again, all such generated orders are not correct ones. Using homomorphism-closedness, it is argued that other "faulty" orderings are not problematic.

Similar techniques (introduction of fresh existentials in one approach [70], use of disjunction in the other approach [22]) are used for simulating input negation, with the same behavior with respect to the termination of the chase. This detour through disjunction allowed for an easier formal treatment of the construction of the tape. Avoiding this detour is possible, by removing disjunction through the use of the chasing set technique, as presented in Section 3.3.3.

3.3.3 Disjunction Removal and Chasing Sets

Another interesting technical ingredient that has been used in this construction is that of chasing set, in order to remove disjunction in a terminating way. We illustrate it by showing how to reduce disjunctive Datalog towards restricted chase terminating existential rule. A naive way to perform this reduction would go as follows: we associate a fresh null with the set of atoms that is valid in the initial database. We say that this fresh null represent the initial "world". Then, for each world, if the body of a disjunctive rule is mappable into its atom, two fresh nulls are added, representing the two possible ways of completing the data to make the head rule satisfied. A query is satisfied in a world, either if it maps directly to the atoms associated with that world, or if there is a rule applicable on that world so that both generated worlds by this rule application satisfy the query.

Unfortunately, some restricted chase sequences will terminate, but not all of them. To avoid this unfortunate behavior, we lift the *chasing set technique* [57]. We call a set of ground atoms a "world" and they will be denoted by variables w (possibly sub- or superscripted) in the figures. Facts $\mathtt{Ins}_{\mathtt{p}}(\vec{x}, w, w')$ express that world w' is obtained by adding $\mathtt{p}(\vec{x})$ to world w. In particular, the fact $\mathtt{Ins}_{\mathtt{p}}(\vec{x}, w, w)$ states that $\mathtt{p}(\vec{x})$ is in w. Rule (16) creates an empty initial world, which is populated by facts from the original database by successive applications of Rules (17), (20) and (21). Note that the presence of `Done` in the body of Rule (17), together with the other two rules, ensure that a world is "complete", meaning that it contains all the atoms that stems from the world from which it as been created, before having the possibility to trigger the creation of a new world. Rules (18) and (19) allow to simulate the disjunction, again imposing through the presence of $\mathtt{Done}(w)$ in the body that the world is complete before triggering the creation of new worlds.

$$\rightarrow \exists w.\texttt{Init}(w) \wedge \texttt{Done}(w) \wedge \texttt{Empty}(w) \tag{16}$$

$$\texttt{Done}(w) \wedge \texttt{Init}(w) \wedge \texttt{p}(\vec{x}) \rightarrow \exists w'.\texttt{Ins}_{\texttt{p}}(\vec{x}, w, w') \wedge \texttt{Subs}(w', w') \wedge \texttt{Init}(w') \tag{17}$$

$$\texttt{Done}(w) \wedge \bigwedge_{\texttt{p}(\vec{x}) \in B} \texttt{Ins}_{\texttt{p}}(\vec{x}, w, w) \rightarrow \exists w_1.\texttt{Ins}_{\texttt{p}_1}(\vec{x_1}, w, w_1) \wedge \texttt{Subs}(w_1, w_1) \tag{18}$$

$$\texttt{Done}(w) \wedge \bigwedge_{\texttt{p}(\vec{x}) \in B} \texttt{Ins}_{\texttt{p}}(\vec{x}, w, w) \rightarrow \exists w_2.\texttt{Ins}_{\texttt{p}_2}(\vec{x_2}, w, w_2) \wedge \texttt{Subs}(w_2, w_2) \tag{19}$$

$$\texttt{Ins}_{\texttt{p}}(\vec{x}, w_0, w_1) \wedge \texttt{Subs}(w_1, w_2) \rightarrow \texttt{Ins}_{\texttt{p}}(\vec{x}, w_2, w_2) \wedge \texttt{p}'(\vec{x}, w_2) \wedge \texttt{Subs}(w_0, w_2) \tag{20}$$

$$\texttt{Empty}(w) \wedge \texttt{Subs}(w, w') \rightarrow \texttt{Done}(w') \tag{21}$$

Figure 9: The rule set Σ'_1, where we instantiate (17) and (20) for all $\texttt{p} \in \mathbf{P}_{\mathcal{E}}$, and (18) and (19) for all $B \rightarrow \texttt{p}_1(\vec{x_1}) \vee \texttt{p}_2(\vec{x_2}) \in \Sigma_1$.

$$\texttt{Goal}'(w) \rightarrow \texttt{Acc}(w) \tag{22}$$

$$\begin{aligned} &\texttt{Ins}_{\texttt{p}_1}(\vec{x}_1, w, w_1) \wedge \texttt{Acc}(w_1) \wedge \\ &\texttt{Ins}_{\texttt{p}_2}(\vec{x}_2, w, w_2) \wedge \texttt{Acc}(w_2) \wedge \\ &\qquad \bigwedge_{\texttt{p}(\vec{x}) \in B} \texttt{Ins}_{\texttt{p}}(\vec{x}, w, w) \rightarrow \texttt{Acc}(w) \end{aligned} \tag{23}$$

$$\texttt{Init}(w) \wedge \texttt{Acc}(w) \rightarrow \texttt{Goal} \tag{24}$$

Figure 10: The rule set Σ'_3, where (23) is instantiated for all rules $B \rightarrow \texttt{p}_1(\vec{x_1}) \vee \texttt{p}_2(\vec{x_2}) \in \Sigma_1$

The rules from Figure 10 allow to "propagate back" acceptance from "completed worlds" towards the initial world.

3.4 A Word on Singe Headedness

Let us quickly discuss an issue regarding the allowed syntax of existential rules. We used rules that do not have any restriction on the number of atoms that appear in the head of rules (as long as it remains finite), and indeed, the rules presented in Section 3.3.3 have several atoms in their heads. Quite often, a stronger assumption is made, namely that rule heads are atomic [28, 74, 44, 58]. It is often considered to be done without loss of generality, because there are normalization procedures that allow to reduce the problem of conjunctive query answering under rules with arbitrary heads to conjunctive query answering under rules with atomic head. The most classical such normalization procedure is the following.

Definition 17 (One Way Atomic Decomposition)**.** *The one-way atomic decomposition of a rule $\rho = B(\vec{x}, \vec{y}) \rightarrow H(\vec{y}, \vec{z})$ is a ruleset containing the rule $B \rightarrow \exists \vec{z} X_\rho(\vec{y}, \vec{z})$ and for each atom $\texttt{p}(\vec{t}) \in H$, the rule $X_\rho(\vec{y}, \vec{z}) \rightarrow \texttt{p}(\vec{t})$, where X_ρ is a fresh predicate unique for ρ, of arity $|\vec{x}| + |\vec{z}|$. The one-way atomic decomposition of a rule set is the union of the one-way atomic decomposition of its rules.*

In [34], we showed that this normalization procedure actually behaves quite badly with respect to the termination of the chase, and we introduced another normalization procedure, that behaves in a better way, the two-way atomic decomposition, meaning that it preserves the termination of all the chases except for the restricted chase. This latter case appears to be much more intricate, as witnessed by the following theorem.

Theorem 3. *No computable function f exists that maps rule sets to rule sets having atomic-head rules such that $\mathcal{R} \in \mathbb{CT}^{\mathsf{r}}_{\forall\forall}$ if and only if $f(\mathcal{R}) \in \mathbb{CT}^{\mathsf{r}}_{\forall\forall}$.*

This latter proof relies on the use of a technique called the emergency brake [57], which crucially use rules having multiple atom in the head. This feature has a strong impact on the termination of the restricted chase. For instance, the *fairness theorem* [45], which holds for single head existential rules, is known to fail for multi-head rules [45, 58].

4 Ontologies as Knowledge Representation: Beyond Conjunctive Queries

The previous chapter focuses on a setting with two important features:

- we only considered Boolean queries, that is, functions from the set of instances to $\{0, 1\}$. This is a quite natural choice of a topic of investigation, in particular because there are usually reductions from Boolean query entailment to and from query answering. But this remains nonetheless a restricted setting, and many non-Boolean queries are used very routinely, for instance in data analytical pipelines.
- the set of existential rules has been considered as part of the query. While this is a totally valid abstract view, and is actually the classical standpoint in the database community, it is somewhat odd in a knowledge representation view. Indeed, ontologies are usually considered fixed and part of knowledge base, which is then queried through a given query language. In this setting, it is clear that conjunctive queries do not provide the same expressivity as, for instance, navigational query languages, such as regular path queries. Actually, the historical focus of the KR community either do not consider any query (and only focus on the satisfiability of an ontology), or on very simple queries, namely instance queries.

These two points motivate that we investigate extensions of the "conjunctive query part" of an ontology mediated query, and we present in this section a couple of papers whose aim is to extend the query language part in a KR view. We consider two different kinds of extensions:

- staying in the realm of Boolean queries, we wonder how complex it is to use regular path queries (and even conjunctive regular path queries), study the complexity of such queries under classical fragments of existential rules, and propose algorithms for answering such queries. This line of work extends related work that has focused on such query languages for much weaker ontology languages, namely members of the DL-Lite family [18];
- we also go beyond the Boolean setting, and make a first step towards the use of aggregate queries in ontology-mediated query answering. To that aim, we consider counting queries, define their semantics, and study the complexity of answering such queries under a variety of description logics, ranging from the very restricted DL-Lite$_{pos}$ to the much more expressive $\mathcal{ALCHI}$.

The rest of this chapter focuses on each of these contributions.

4.1 Conjunctive Regular Path Queries

The so-called navigational queries have gained in popularity due to their adequation in the setting of graph databases, which makes natural the querying of neighbors. The easiest form

of such queries is that of regular path queries, which asks for pair of nodes (a, b) for which there is a path from a to b that is labeled by a word that belongs to a regular language Λ. These queries have been studied both in a context without any ontology [63, 42], or with ontologies expressed in lightweight description logics [18]. In this line of work, we aimed at generalizing the setting by considering more expressive ontology languages, such as linear rules and guarded rules. We recall that a non-deterministic automaton (NFA) is a tuple $\mathbb{A} = (S, \Sigma, \delta, s_0, F)$, where S is a finite set of states $\delta \subseteq S \times \Sigma \times S$, $s_0 \in S$ and $F \subseteq S$. We denote by $\mathbf{P}_2$ the subset of $\mathbf{P}$ consisting of binary predicates. Given a binary predicate $\mathtt{p}$, we also consider its inverse predicate $\mathtt{p}^-$. We set $\mathbf{P}_2^{\pm} = \mathbf{P}_2 \cup \{\mathtt{p}^- \mid \mathtt{p} \in \mathbf{P}_2\}$.

A *path predicate* Λ is a binary predicate described by an NFA over $\mathbf{P}_2^{\pm}$. A *path atom* is an atom built on a path predicate, *i.e.*, it takes the form $\Lambda(t_1, t_2)$, where Λ is a path predicate and t_1, t_2 are terms. Note that standard binary predicates are special cases of path predicates. We assume w.l.o.g. that automata associated with different path predicates have disjoint sets of states. A conjunctive (two-way) regular path query (CRPQ) has the form $q(\vec{x}) = \exists \vec{y} B$, where $\vec{x}$ and $\vec{y}$ are disjoint tuples of variables, and B is a conjunction of standard and path atoms, with $\mathsf{term}(B) = \vec{x} \cup \vec{y}$.

4.1.1 Lower Bound for RPQs and Linear Rules

It is well known that linear rules allow to simulate polynomial space Turing machines [47]. The idea is to encode the tape of a Turing machine through a single atom of polynomially large arity, allowing to simulate a transition of a given Turing machine by a single application of a linear rule. The accepting condition is then checked by an atomic query that checks for the accepting state (where the head can safely be assumed to be on the first position of the tape).

A technique introduced in [19] has been to use regular path queries to be able to encode not only polynomial space Turing machine, but also polynomial space *alternating* Turing machines. This is done by modifying the construction proposed by Gottlob and Papadimitriou [48] for encoding Turing machines configurations by atoms, adding "input" and "output" positions to that encoding. The existence of a path from the input position tot the output position of the atom encoding a configuration will represent the fact that this configuration is accepting. A noteworthy fact is that the regular language that is used for that reduction is actually one of the simplest one: p^*.

More formally, the construction of the ruleset works as follows. Given a PSpace Turing machine and an input x, we represent a configuration c reached during the computation by storing the content of the first $p(|x|)$ cells, as well as the head of the tape and the current state of the Turing machine. Adding the so-called input and the output positions, this can be encoded by a predicate conf of arity $2 \times p(|x|) + 3$:

$$\mathsf{conf}(i_x, state, cell_1, cur_1, cell_2, cur_2, \ldots, cell_{p(|x|)}, cur_{p(|x|)}, o_c),$$

where $state$ contains the state identifier, $cell_i$ represent the content of the i^{th} cell, cur_i is equal to 1 if the head of the Turing machine is on cell i and 0 otherwise, and i_c (resp. o_c) is the input (resp. output) term of this atom. We say that the above atom *represents* configuration c.

For every accepting state q_f, we create the following rule:

$$\mathsf{conf}(i_c, q_f, \ldots, o_c) \to p(i_c, o_c),$$

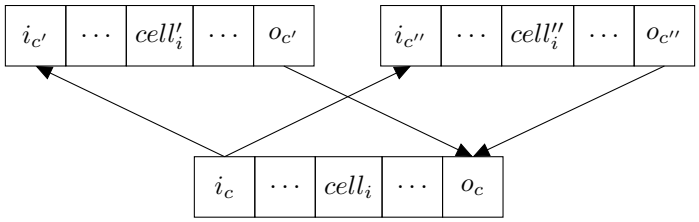

Figure 11: The existential gadget

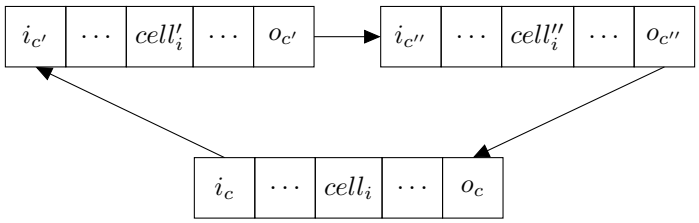

Figure 12: The universal gadget

hereby creating a path from the input position to the output position of any atom representing an accepting configuration. Existential states are treated as represented in Figure 11: for each transition $\delta(q,\gamma) = \{(q',\gamma',L),(q'',\gamma'',L)\}$ such that q is existential, we create the rule:

$$\begin{aligned}\mathsf{conf}(i_c, q, cell_1, cur_1, \ldots, cell_{i-1}, 0, \gamma, 1, \ldots, o_c) \rightarrow \\ \exists i_{c'}, o_{c'}, i_{c''}, o_{c''}\ \mathsf{conf}(i_{c'}, q', cell_1, cur_1, \ldots, cell_{i-1}, 1, \gamma', 0, \ldots, o_{c'}), \\ \mathsf{conf}(i_{c''}, q'', cell_1, cur_1, \ldots, cell_{i-1}, 1, \gamma'', 0, \ldots, o_{c''}), \\ p(i_c, i_{c'}), p(o_{c'}, o_c), p(i_c, i_{c''}), p(o_{c''}, o_c). \quad (25)\end{aligned}$$

and similar rules for cases where the Turing machine head moves in different directions. As for universal states, we associate with each transition $\delta(q,\gamma) = \{(q',\gamma',L),(q'',\gamma'',L)\}$ the following rule:

$$\begin{aligned}\mathsf{conf}(i_c, q, cell_1, cur_1, \ldots, cell_i, 0, \gamma, 1, \ldots, o_c) \rightarrow \\ \exists i_{c'}, o_{c'}\ , i_{c''}, o_{c''}\ \mathsf{conf}(i_{c'}, q', cell_1, cur_1, \ldots, cell_i, 1, \gamma', 0, \ldots, o_{c'}), \\ \mathsf{conf}(i_{c''}, q'', cell_1, cur_1, \ldots, cell_i, 1, \gamma'', 0, \ldots, o_{c''}), \\ p(i_c, i_{c'}), p(o_{c'}, i_{c''}), p(o_{c''}, o_c). \quad (26)\end{aligned}$$

4.1.2 Linearization of a Set of Guarded Rules

Another technique of interest is that of *linearization* of guarded rules. This linearization allows to reduce CRPQ answering under guarded rules towards CRPQ answering under linear rules. Two main ideas underly this reduction. First, we define an alternative notion

of derivation, called *locally complete*. A locally complete derivation is such that at each step k, the current instance $\hat{I}_k$ contains exactly the subset of the result of the chase to the terms of $\hat{I}_k$. Of course, such derivations are not computable for arbitrary sets of rules, because that would imply the decidability of atomic entailment. However, they are computable for guarded rules. Second, applications of guarded rules in locally complete derivations are simulated through applications of linear rules in a classical derivation expressed on an extended vocabulary, in which each predicate represents a guarded set.

Definition 18 (Locally Complete Derivation)**.** *Let I be an instance and $\mathcal{R}$ be a set of existential rules. A* locally complete $\mathcal{R}$-derivation *from I resulting in $\hat{I}_n$ is a sequence $\hat{I}_0, \hat{I}_1, \ldots, \hat{I}_n$ with $\hat{I}_0 = \mathsf{chase}(I, \mathcal{R})_{|\mathsf{term}(I)}$, and for any $i \geq 1$, there is a trigger (ρ_i, π_i) on $\hat{I}_{i-1}$ such that $\hat{I}_i = \mathsf{chase}(\hat{I}'_{i-1}, \mathcal{R})_{|\mathsf{term}(\hat{I}'_{i-1})}$ where $\hat{I}'_{i-1} = \alpha(\hat{I}_{i-1}, \rho_i, \pi_i)$.*

The equivalence with classical derivation stems from the following two propositions.

Proposition 1. *For any classical $\mathcal{R}$-derivation from I resulting in I', there exists a locally complete $\mathcal{R}$-derivation from I resulting in $\hat{I}$ such that $\hat{I} \models I'$.*

Proposition 2. *For any locally complete $\mathcal{R}$-derivation from I resulting in $\hat{I}$, there exists a classical $\mathcal{R}$-derivation from I resulting in I' such that $I' \models \hat{I}$.*

The linearization technique can be seen as a simulation of locally complete derivations. One starts from a set of atoms (on an extended vocabulary) that encodes the saturation of the chase restricted to the terms of the initial instance. Then linear rules are used (still on the extended vocabulary) to simulate locally complete derivations. Let us have a look at this construction on an example.

Example 3. *Let us consider $I = \{p(a), r(a,b), r(b,c), q(c)\}$ and $\mathcal{R} = r(x,y) \wedge q(y) \rightarrow q(x); p(x) \wedge q(x) \rightarrow \exists y s(x,y)\}$.*

We consider a set of atoms G such that each atom α of the database has a set of terms that is a subset of the terms of g_α for some $g_\alpha \in G$. Here, we could consider $r(a,b)$ and $r(b,c)$. I restricted to the terms of $r(a,b)$ is $\{r(a,b), p(a)\}$, and we could represent that set by the atom $p_{\{r(X_1,X_2),p(X_1)\}}(a,b)$. Similarly, we could encode the set of atoms $r(b,c), q(c)$ through the atom $p_{\{r(X_1,X_2),q(X_2)\}}(b,c)$.

The effect of the rule $p(x) \wedge q(x) \rightarrow \exists y\ s(x,y)$ would be represented (among others: there would be one such rule per type!) by rules of the shape:

$$p_{\{r(X_1,X_2),p(X_2),q(X_2)\}}(x,y) \rightarrow \exists z\ p_{\{p(X_1),q(X_1),s(X_1,X_2)\}}(y,z)$$

$$p_{\{s(X_1,X_2),p(X_1),q(X_1)\}}(x,y) \rightarrow \exists z\ p_{\{p(X_1),q(X_1),s(X_1,X_2)\}}(x,z).$$

Note that this technique indeed reduces reasoning under guarded rules to reasoning under linear rules, but it comes with a major drawback: the generated set of rules is of double exponential size, as we introduce one "complex" predicate per type of the initial ruleset. Given the complexity of conjunctive query answering under linear rules lies in PSpace, whereas it is 2-ExpTime-complete for guarded rules, this is not surprising. Making such a transformation practical may not be totally unreasonable, assuming that not necessarily arbitrary types may appear in real-life setting (and that one of the exponential level is due to unbounded arities, whereas as the arity can be considered small in some practical setting). Constraints (which are usually not exploited for query answering algorithms

for ontology-mediated query answering) could prove very useful as a means to reduce the involved combinatorics.

Note that a very similar technique has been independently defined by Gottlob et al. ([46], Section 4.2), with the aim of defining polynomial combined rewritings for specific sets of guarded rules. These kind of rewritings allows for modification of both the data and the query: is is originally inspired from [59] for $\mathcal{ELH}_{\bot}^{dr}$, an extension of $\mathcal{EL}$ allowing for role hierarchies, domain and range restrictions as well as inconsistencies. Linearization has also recently been reused to study termination of the semi-oblivious chase [24].

4.1.3 Perspectives

The approach used here has been a lifting of the techniques proposed for description logics. This is done possible by the tree-like behavior that is ensured by guardedness: investigating what may happen when guardedness is not enforced anymore is an interesting challenge. Moreover, I am currently investigating with an intern, Gaia Carenini, how to use CRPQ answering as a black box to solve conjunctive query answering under linear rules and transitivity rules [10], which is an important open problem as transitivity naturally appears in some modeling.

4.2 Counting Queries

We now present some technical results regarding another type of queries, that we called counting queries, in the setting of ontology-mediated query answering. Contrarily to the rest of this dissertation, the formalism used in this section is that of Description Logics (DLs, see [8, 9]). This is motivated by several factors: *(i)* description logics are a prominent formalism to represent ontologies, that is both studied theoretically and at the core of numerous applications, notably by underpinning the Semantic Web standard of OWL 2 EL, OWL 2 QL and OWL 2 EL; *(ii)* related work on adding counting features with ontological knowledge focused on description logics, which made the comparison easier; *(iii)* description logics have only unary and binary predicates, which remove part of the difficulty that arise when considering existential rules. An example of these additional difficulties has been illustrated in Section 4.1 when lifting results from DL-Lite to linear existential rules. Avoiding these difficulties allowed us, with Meghyn and Quentin, to tackle perhaps more interesting questions, such as extending our algorithm originally developed for DL-Lite in a (relatively) direct way to the non-Horn description logics $\mathcal{ALCHI}$.

After introducing the semantics of counting conjunctive queries, we'll briefly describe some specific cases to illustrate the diversity of behaviors that can happen with respect to the data complexity of counting queries with atomic queries and very simple ontologies, before introducing a tool called *interlacing*, which allows to provide tight upper bounds for the description logics $\mathcal{ALCHI}$.

4.2.1 Semantics

We now turn to defining the syntax and semantics of counting queries. Syntactically speaking, a counting query is very similar to a conjunctive query: the only difference is that non-answer variables are split between existential variables and counting variables.

Definition 19. *A* counting conjunctive query (CCQ) *takes the form*

$$q(\vec{x}) = \exists \vec{y}\ \exists \vec{z}\ \psi(\vec{x}, \vec{y}, \vec{z}),$$

where $\vec{x}, \vec{y}, \vec{z}$ are tuples of answer, existential, *and* counting variables, *respectively, and ψ is a conjunction of concept and role atoms with terms from $\mathbf{I} \cup \vec{x} \cup \vec{y} \cup \vec{z}$. A CCQ q is* Boolean *if $\vec{x} = \emptyset$.*

We rely on the notion of matches and of counting matches to define the semantics of counting queries.

Definition 20. *A match for a CCQ q in an interpretation $\mathcal{I}$ is a function π that maps each term in q to an element of $\Delta^{\mathcal{I}}$ such that $\pi(t) = t$ if $t \in \mathbf{I}$, $\pi(t) \in A^{\mathcal{I}}$ for every $A(t) \in q$ and $(\pi(t), \pi(t')) \in P^{\mathcal{I}}$ for every $P(t, t') \in q$. If a match π maps $\vec{x}$ to $\vec{a}$, then the restriction of π to $\vec{z}$ is called a* counting match *of $q(\vec{a})$ in $\mathcal{I}$.*

We first define the notion of answer for a counting query in an interpretation.

Definition 21 (Answer for a Counting Query in an Interpretation)**.** *The set of* answers *to q in $\mathcal{I}$, denoted by $q^{\mathcal{I}}$, contains all pairs $(\vec{a}, [m, M])$ with $m, M \in \mathbb{N} \cup \{+\infty\}$, such that the number of distinct counting matches of $q(\vec{a})$ in $\mathcal{I}$ belongs to the interval $[m, M]$.*

Note that we have defined it as a pair $(\vec{a}, [m, M])$, where m and M are respectively upper and lower bounds on the number of matches in the considered interpretation rather than a pair $(\vec{a}, m^*)$, with m^* being the exact number of matches. This allows to define the notion of certain answer in a very classical way, that is, the intersection of answers over every model of the knowledge base, while still getting answer whenever there exists a match mapping the answer variables to $\vec{a}$.

Definition 22 (Certain Answer)**.** *A* certain answer *to q w.r.t. $(I, \mathcal{R})$ is an answer in every model of $(I, \mathcal{R})$.*

Note that for the ontology languages that we are considering, the upper bound is not relevant (as one can almost always ensure that whenever there are matches, there are arbitrarily many), and we thus focus on finding k such that $[k, +\infty]$ is a certain answer.

This notion of answer generalizes and unifies the notions introduced by Kostylev and Reutter [56]. Their complexity results can be lifted to our case, but large gaps remained. A major goal has been to close these gaps, both in the restricted setting of DL-Lite ontologies, as well as in more expressive ontology languages, such as $\mathcal{EL}$ or $\mathcal{ALCHI}$.

Given the high complexity of answering counting conjunctive queries, it has also been a focus to find restrictions ensuring a tractable problem in data complexity. All known tractable cases were relying on restrictions ensuring that the universal model is optimal, which we will see is very restrictive. We found ways to find restrictions ensuring that an optimal model can be found by merging the universal model according to some strategy, to be found among a set of strategies depending (mostly) on the TBox, ensuring small data complexity.

4.2.2 A Variety of Data Complexities

In this section, we consider role cardinality queries, that is, queries of the shape

$$q_S = \exists z_1, z_2 S(z_1, z_2),$$

where both z_1 and z_2 are counting variables, and are interested in the data complexity of checking whether $[m, +\infty]$ is a certain answer. Note that under the empty ontology, this can be done within TC^0. However, adding an ontology can lead to a variety of other behaviors, that we illustrate through the following examples.

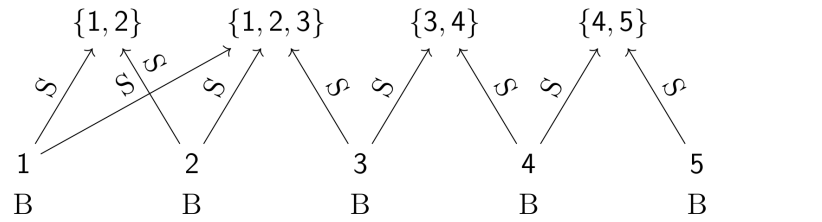

Figure 13: Illustration of the reduction from Example 4: there exists a k-cover if and only if $9 + k + 1$ is not a certain answer for q_S.

A co-NP completeness proof The paper from Kostylev and Reutter already showed that answering counting queries can be co-NP-hard in data complexity, but used rather involved queries and ontologies to do so. The following example is a much simpler case, which we will leverage soon after.

Example 4. *Let us consider* $\mathcal{T} = \{B \sqsubseteq \exists R_1, R_1 \sqsubseteq S, \exists R_1^- \sqsubseteq \exists R_2, R_2 \sqsubseteq S\}$. *There exists a non-trivial propagation of* S. *We reduce the complementary of the problem* Set Cover *to the problem of answering the role cardinality query for* S. *For an instance of* Set Cover, *we build the ABox* $\mathcal{A} = \{B(u) \mid u \in \mathcal{U}\} \cup \{S(u, s) \mid u \in s, s \in \mathcal{S}\}$. *It can be shown that there exists a* k*-cover if and only if* $[\Sigma_{s \in \mathcal{S}}|s| + k + 1, +\infty]$ *is* not *a certain answer for* q_S *over* $(\mathcal{T}, \mathcal{A})$. *This reduction is illustrated in Figure 4.2.2.*

A case L-equivalent to the complement of Perfect Matching Let us consider the following TBox $\mathcal{T}$:

$$\{B \sqsubseteq \exists R, R \sqsubseteq S, R \sqsubseteq S^-\}.$$

Under this ontology, a model of an instance minimizing the number of S atoms is obtained by pairing elements for which B holds, but do not have any ingoing or outgoing R atom. This allows to perform a reduction to and from the complement of Perfect Matching in logarithmic space.

A trichotomy for DL-Lite$^{\mathcal{H}}_{\mathsf{pos}}$... but not beyond Quite surprisingly, it happens that the cases described so far are at the basis of a trichotomy for role cardinality queries under DL-Lite$^{\mathcal{H}}_{\mathsf{pos}}$ ontologies.

Theorem 4. *Let* $\mathcal{T}$ *be a DL-Lite*$^{\mathcal{H}}_{\mathsf{pos}}$ *TBox. Answering the role cardinality query* $q_S = \exists z_1 \exists z_2 S(z_1, z_2)$ *over* $\mathcal{T}$ *is either* co-NP*-complete,* L*-equivalent to the complement of* Perfect Matching, *or within* TC^0.

The proof of that theorem relies on delimiting the cases where the above reductions are adaptable, and showing that whenever none of the above constructions are applicable, then

the problem can be solved in TC^0. Interestingly, these tractable cases do not rely on the universal model being an optimal model, as was the case in the literature so far [30]. Trying to generalize the trichotomy to the the more expressive DL-Lite$^{\mathcal{H}}_{\mathsf{core}}$, we uncovered not only more sources of NP-hardness, but also an L-complete case, as witnessed by the following example.

Theorem 5. *Answering the role cardinality query q_S over the DL-Lite$^{\mathcal{H}}_{\mathsf{core}}$ TBox $\mathcal{T} = \{B \sqsubseteq \exists R, R \sqsubseteq S, R \sqsubseteq \neg R^-\}$ is* L*-complete.*

The hardness proof is done by reduction from the Undirected Forest Accessibility problem, known to be L-complete [35], while the upper bounds rely on the fact that reachability in undirected graphs is solvable in logarithmic space [69].

4.2.3 A Tool for Providing Upper-Bounds: Interlacings

The above examples give a gist of which behaviors may arise when trying to answer counting queries. It is quite clear that even in cases where the universal model exist, it may not be optimal with respect to the minimization of the number of counting matches. A major contribution of Quentin's PhD has been to significantly generalize an approach for "structuring" models having a given number of counting matches. We illustrate the construction with the example represented in Figures 14 and 15. Starting from a model $\mathcal{I}$ of a given ABox and TBox, the model is "unravelled" by choosing for each element and each concept of the shape $\exists R.A$ appearing in a right handside of a TBox axiom a successor, freshly instantiated in a new model called *the existential extraction* of $\mathcal{I}$. Such an existential extraction (for a given choice of successors) is represented Figure 15. We then consider models that are obtained by merging in a sufficiently regular way this existential extraction in order to obtain other models of the considered $\mathcal{A}_e$ and $\mathcal{T}_e$, without increasing the number of matches of the query under consideration. This allows us to get information about the structure of a model minimizing the number of match of a given query, and to adopt an approach of enumeration and checks.

4.2.4 Perspectives

The problem of answering counting conjunctive queries appeared to be much more complex than initially thought. As possible follow-ups, the following lines of research are quite natural: extending the expressivity of the ontologies that are supported by our approach, in particular with respect to the counting features, such as functionality; using SAT solvers to allow for an efficient computation of the combinatorially hard part in the cases that are tractable in data complexity; better understanding the links with other problems such as reasoning under closed predicates [65]; defining the semantics of other operators under incomplete knowledge such as those introduced in practical query languages [43]; investigating the approximability of the semantics we proposed in [61].

5 Not Only Answers: Adding Provenance

Much of the work on ontology-mediated query answering has focused on providing answers to Boolean queries, with the possible extension to queries with answer variables, in which case answer tuples are provided. But a user may be interested in more than the plain answers [72]: why is a tuple an answer? where values come from? how is it generated? how

$$\begin{array}{llll} \mathrm{A} \sqsubseteq \exists \mathrm{R}.\mathrm{A}' & \mathrm{B} \sqsubseteq \mathrm{B}' \sqcup \mathrm{D} & \mathrm{D} \sqsubseteq \exists \mathrm{S}.\mathrm{D} & \mathrm{A} \sqcap \mathrm{B} \sqsubseteq \bot \\ \mathrm{A}' \sqsubseteq \exists \mathrm{R}.\mathrm{A} & \mathrm{B}' \sqsubseteq \exists \mathrm{R}.\mathrm{C} & \mathrm{C} \sqsubseteq \exists \mathrm{S}.\mathrm{A} & \mathrm{R} \sqcap \mathrm{R}^- \sqsubseteq \bot \\ \mathrm{A}' \sqsubseteq \exists \mathrm{R}.\mathrm{B} & \mathrm{B}' \sqsubseteq \exists \mathrm{T}.\mathrm{D} & \mathrm{C} \sqsubseteq \exists \mathrm{S}.\mathrm{B} & \mathrm{D} \sqcap \exists \mathrm{R}^-.\mathrm{A}' \sqsubseteq \bot \end{array}$$

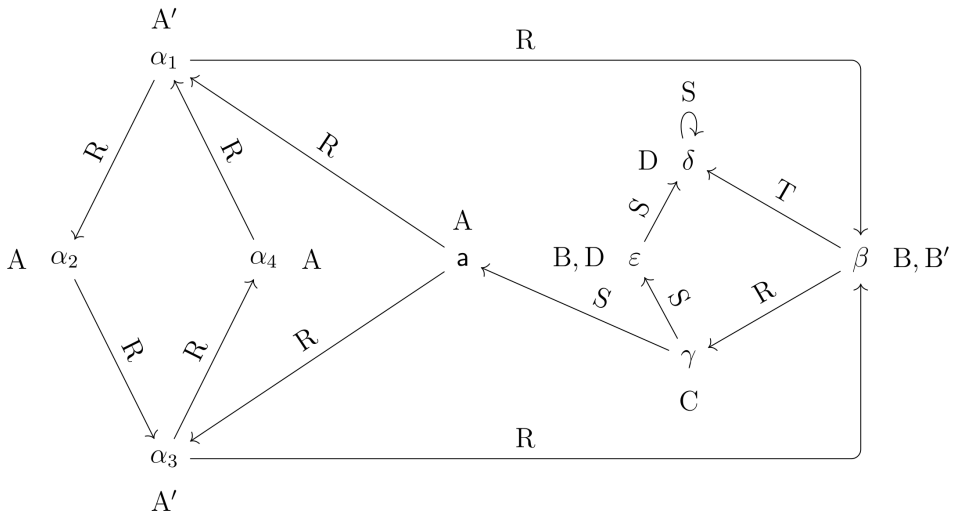

Figure 14: A TBox $\mathcal{T}_e$, and a model $\mathcal{I}_e$ of $\mathcal{T}_e$ and of the ABox $\{A(a)\}$.

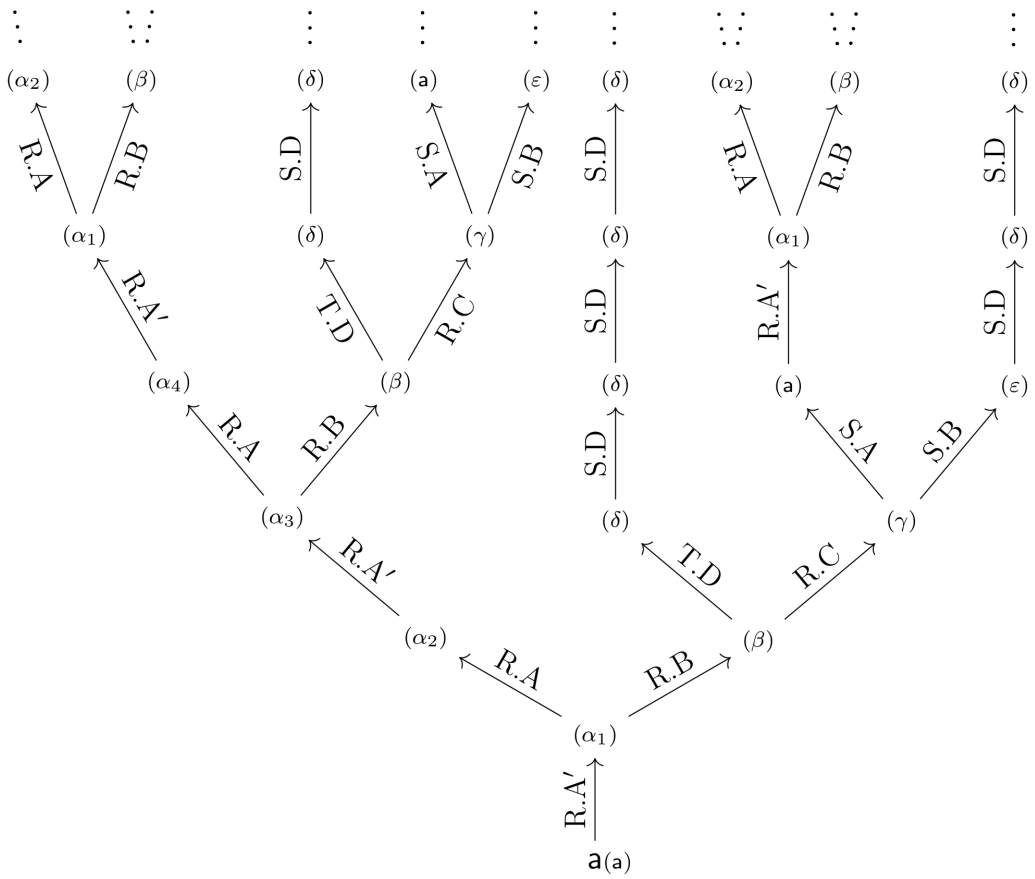

Figure 15: The existential extraction of $\mathcal{I}_e$

many times is it obtained? with which probability does the answer hold, given a probability distribution on the the input data? which security clearance is needed to see the result?

All these questions can be seen as a problem of *provenance*. Prominent in the setting of data provenance, *semiring provenance* have been introduced in a paper by Green et al. [50], neatly generalizing previous provenance notions. We first recall the setting of semiring provenance, show how different semantics proposed in the literature do not agree in settings where all are applicable, and discuss which properties could reasonably be expected from a provenance semantics. It appears that defining a good notion of provenance in the ontology-mediated query answering setting is not a closed topic. This line of thought has been pursued in a joint work with Camille Bourgaux, Pierre Bourhis and Liat Peterfreund [21].

5.1 Semiring Provenance

Let us recall the definition of a semiring.

Definition 23 (Semiring)**.** *A semiring* $\mathbb{K} = (K, +_\mathbb{K}, \times_\mathbb{K}, 0_\mathbb{K}, 1_\mathbb{K})$ *is a set* K *with distinguished elements* $0_\mathbb{K}$ *and* $1_\mathbb{K}$*, equipped with two binary operations* $+_\mathbb{K}$*, called the addition, which is an associative and commutative operator with identity* $0_\mathbb{K}$ *and* $\times_\mathbb{K}$*, called the the multiplication, which is an associated operator with identity* $1_\mathbb{K}$*. When multiplication is commutative, the semiring is said to be commutative.*

Classical semirings in this setting include for instance the *counting* semiring $(\mathbb{N}, 0, 1, +, \times)$, the *Boolean* semiring $(\{\bot, \top\}, \bot, \top, \vee, \wedge)$, the *tropical* semiring $(\mathbb{N} \cup \{\infty\}, \infty, 0, \mathsf{min}, +)$, the semiring of *positive Boolean functions over variables* X, the semiring of *integer polynomials* with variables in X. Semirings elements can be used to annotate a database, henceforth allowing to derive the provenance of an answer tuple.

Definition 24 (Annotated Database)**.** *An* annotated database *is a triple* $(I, \mathbb{K}, \lambda)$ *where* I *is an instance,* $\mathbb{K}$ *is a semiring and* $\lambda : I \to K \setminus \{0_\mathbb{K}\}$ *maps facts into semiring elements different from* $0_\mathbb{K}$*.*

Annotations on instances can be lifted to annotate query answers. A rather consensual case is the way that answers to conjunctive queries are annotated, where, at an intuitive level, joint use of data (conjunction) corresponds to the multiplication of their annotations and alternative use (such as union of projection) corresponds to the addition of their annotations.

The seminal paper on provenance semirings also define the provenance for a tuple generated by a Datalog program. That definition is based on the notion of derivation tree, which we first recall.

Definition 25 (Derivation Tree)**.** *A* derivation tree t *of a fact* α *w.r.t. a database* I *and a program* $\mathcal{R}$ *is a finite tree whose leaves are labeled by facts from* I *and non-leaf nodes are labeled by triples* $(p(t_1, \ldots, t_m), \alpha, \pi)$ *where*

- $p(t_1, \ldots, t_m)$ *is an atom with* $p \in \mathbf{P}$*;*
- α *is a rule from* $\mathcal{R}$ *of the form* $\phi(\vec{x}, \vec{y}) \to p(\vec{x})$*;*
- h *is a homomorphism from* $\phi(\vec{x}, \vec{y})$ *to the atoms of the labels of the node children, such that* $\pi(p(\vec{x})) = p(t_1, \ldots, t_m)$*;*
- *there is a bijection* f *between the node children and the atoms of* $\phi(\vec{x}, \vec{y})$*, such that for every* $q(\vec{z}) \in \phi(\vec{x}, \vec{y})$*,* $f(q(\vec{z}))$ *is of the form* $(\pi(q(\vec{z})), \alpha', \pi')$ *or is a leaf labeled by* $\pi(q(\vec{z}))$*.*

Moreover, if $(p(t_1, \cdots, t_m), \alpha, \pi)$ *or* $p(t_1, \cdots, t_m)$ *is the root of* t*, then* $p(t_1, \cdots, t_m) = \alpha$*.*

The annotation of a derivation tree is the product of the annotations of its leaves. This is well-defined on commutative $\mathbb{K}$, which are the ones on which semiring provenance is defined. The annotation of an atom is defined as the sum of the annotations of the derivation trees having that atom as root. This value, when defined, is denoted by $\mathcal{P}^{\mathtt{AT}}(\mathcal{R}, I, \mathbb{K}, \lambda, \alpha)$: the provenance of α w.r.t. to $\mathcal{R}$ and I, which has been annotated through λ by values from $\mathbb{K}$.

5.2 Several Notion of Provenance in Recursive Cases

The above definition presents the unsettling property of relying on a possibly infinite sum, which may very well not be defined,[4] even though the computation associated with such queries is finite. This is the case as *all* derivation trees are considered, including derivation trees which are "unnatural", as for instance derivation trees that use an atom to derive itself. It is thus natural to consider only specific subsets of trees, for instance trees that are non-recursive (no two nodes in an ancestor relationship are labeled by the same atom), of minimal depth (among trees having the same root atom), or of hereditary minimal depth (where the condition is also enforced for internal trees). This respectively leads to the provenance $\mathcal{P}^{\mathtt{NRT}}, \mathcal{P}^{\mathtt{MDT}}, \mathcal{P}^{\mathtt{HMDT}}$.

Note also that quite interestingly the all tree, minimal depth tree and hereditary minimal depth tree semantics can alternatively defined based on execution-based semantics. The all tree semantics can be seen as the semantics associated with a naive evaluation of Datalog, the minimal depth trees can be seen as an optimized naive evaluation (where the computation stops once the relevant atom is computed) and the hereditary minimal tree semantics corresponds to a seminaive evaluation of Datalog. See [2] for a presentation of Datalog evaluations.

Another quite natural way to define semantics of provenance is to define annotated models, in similar ways as what is done in [52, 66].

5.3 Rationality Properties

As explained in the previous section, there is no shortage of ways to define a notion of provenance for Datalog queries. How can be compared such notions? Is there one that is better than the others? To address this question in a principled manner, we aimed at defining *rationality properties*, in a way similar to what is done in the belief merging community, where properties that should be fulfilled by a well-behaved operator are introduced, and representation theorems are provided [76]. Such a theorem intuitively states that a given operator is the only one that fulfills a given set of rationality properties.

What would be rationality properties of a provenance notion? In [21], we propose some such properties. This happened to be a task much more intricate than expected, and I believe we are still quite far from understanding which properties should a "reasonable" semantics for Datalog provenance fulfill. To give an idea of which kind of properties we proposed, I present below some of them.

A first property is the consistency between a provenance notion for Datalog and the notion for the rather consensual union of conjunctive queries, that we call the relational provenance.

[4] A common restriction is then to define the provenance only on suitable semirings, for instance, ω-continuous semirings.

Property 1 (Algebra Consistency). *Let $\mathcal{R}$ be a set of Datalog rules of the shape $\phi(\vec{x},\vec{y}) \to H(\vec{x})$, where H is a predicate that does not appear in the body of any rule, and H does not appear in I. Then $\mathcal{P}(\mathcal{R}, I, \mathbb{K}, \lambda, H(\vec{a}))$ is equal to the relational provenance of the query $\cup_{\phi(\vec{x},\vec{y})\to H(\vec{x})\in\mathcal{R}} \exists\vec{y}\phi(\vec{x},\vec{y})$.*

A second property is the possibility to commute the computation of provenance and the application of a semiring homomorphism. This kind of property is very useful on a practical level, as it allows to compute (and possibly store) provenance in a general semiring, before possibly applying a homomorphism to get the provenance in the semiring required by a user.

Property 2 (Commutation with Homomorphism). *If there is a semiring homomorphism h from $\mathbb{K}_1$ to $\mathbb{K}_2$, then $h(\mathcal{P}(\mathcal{R}, I, \mathbb{K}_1, \lambda, \alpha)) = \mathcal{P}(\mathcal{R}, I, \mathbb{K}_1, h \circ \lambda, \alpha)$.*

We also defined properties that exhibit some links between the annotations of the atoms used for deriving an atom and the annotation of the atom. That link is formalized through the binary relation $\sqsubseteq$ over semiring elements, which is defined by $a \sqsubseteq b$ if and only if there exists c such that $a + c = b$.

Property 3 (Self). *If $\alpha \in D$, then $\lambda(\alpha) \sqsubseteq \mathcal{P}(\mathcal{R}, I, \mathbb{K}, \lambda, \alpha)$*

Another kind of property that we consider are intuitively some kind of refinement of algebra consistency. Instead of only agreeing with the original semantics for UCQs, we would like the $+$ and $\times$ operators to correspond to respectively alternative use or joint use of the data. The formalization for $\times$ and joint use of the data is done as follows:

Property 4 (Joint Use). *For all facts $\alpha_1, \ldots, \alpha_n$,*

$$\mathcal{P}(\mathcal{R}', I, \mathbb{K}, \lambda, \mathsf{goal}) = \Pi_{j=1}^{n} \mathcal{P}(\mathcal{R}, I, \mathbb{K}, \lambda, \alpha_j),$$

where $\mathcal{R}' = \mathcal{R} \cup \{\bigwedge_{j=1}^{n} \alpha_j \to \mathsf{goal}.$

Last, we investigate how data modifications impact the provenance, by considering insertion or deletion of atoms. For instance, we could expect provenance to behave as follows with respect to data insertion.

Property 5 (Insertion). *For every $(I', \mathbb{K}, \lambda')$ such that $I \cap I' = \emptyset$,*

$$\mathcal{P}(\mathcal{R}, I, \mathbb{K}, \lambda, \alpha) + \mathcal{P}(\mathcal{R}, I', \mathbb{K}, \lambda', \alpha) \sqsubseteq \mathcal{P}(\mathcal{R}, I \cup I', \mathbb{K}, \lambda \cup \lambda', \alpha)$$

	$\mathcal{P}^{\mathtt{AT}}$	$\mathcal{P}^{\mathtt{NRT}}$	$\mathcal{P}^{\mathtt{MDT}}$	$\mathcal{P}^{\mathtt{HMDT}}$
Algebra Consistency	✓	✓	✓	✓
Com. with Hom.		✓	✓	✓
Joint Use	✓	✓		✓
Self	✓	✓	✓	✓
Insertion	✓	✓		

Table 1: Does a property hold for a provenance semantics?

Table 1 summarizes which properties are fulfilled by which semantics. Note that $\mathcal{P}^{\mathtt{NRT}}$ is the only semantics fulfilling all the properties that we designed (and it actually also holds when considering properties not included in this dissertation, or the model-based semantics).

A word on computational complexity The goal of the above work was to focus on the semantics of provenance, but also is interesting is they actual computability. Results regarding $\mathcal{P}^{\mathtt{AT}}$ semantics are quite positive, with polynomial representation of such semantics (under certain condition) by circuits. The model-based semantics have similarly good properties, but unfortunately, this is not true for $\mathcal{P}^{\mathtt{NRT}}$, for which we prove that there is no polynomially computable circuit representation computing it on $\mathbb{N}^\infty[[X]]$, by reduction from a result by Arenas et al. [5]. Note that very recently Calautti et al. studied the complexity of why-provenance for Datalog queries [25], with yet another semantics.

5.3.1 Perspectives

The quest for deciding what a good semantics for provenance within Datalog is, in my opinion, far from being over. Finding a "nicer" set of properties that a property should fulfill would be an approach; the dual approach of looking at what can be efficiently computed is also natural, and is what has recently been started in [25], for a given semantics based on different kind of trees. Another natural line of research is to go beyond Datalog, and to consider existential rules: the fragment of guarded existential rules seems a natural target for such an extension, due to its nice combinatorial properties and its practical relevance.

References

[1] Snomed webside. `https://www.snomed.org`. Accessed: 2023-05-10.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] F. N. Afrati and P. G. Kolaitis. Answering aggregate queries in data exchange. In M. Lenzerini and D. Lembo, editors, *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 129–138. ACM, 2008.

[4] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. Philos. Log.*, 27(3):217–274, 1998.

[5] M. Arenas, S. Conca, and J. Pérez. Counting beyond a yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In A. Mille, F. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 629–638. ACM, 2012.

[6] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[7] F. Baader. Terminological cycles in a description logic with existential restrictions. In G. Gottlob and T. Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 325–330. Morgan Kaufmann, 2003.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[9] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[10] J. Baget, M. Bienvenu, M. Mugnier, and S. Rocher. Combining existential rules and transitivity: Next steps. In Q. Yang and M. J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2720–2726. AAAI Press, 2015.

[11] J. Baget, M. Bienvenu, M. Mugnier, and M. Thomazo. Answering conjunctive regular path queries over guarded existential rules. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 793–799. ijcai.org, 2017.

[12] J. Baget, M. Leclère, M. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

[13] J. Baget, M. Mugnier, and S. Rudolph. Bounded treewidth and the infinite core chase: Complications and workarounds toward decidable querying. In F. Geerts, H. Q. Ngo, and S. Sintos, editors, *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 291–302. ACM, 2023.

[14] V. Bárány, M. Benedikt, and B. ten Cate. Some model theory of guarded negation. *CoRR*, abs/2005.06299, 2020.

[15] V. Bárány, B. ten Cate, and L. Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, 2015.

[16] BBC. Business news ontology.

[17] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[18] M. Bienvenu, M. Ortiz, and M. Simkus. Regular path queries in lightweight description logics: Complexity and algorithms. *J. Artif. Intell. Res.*, 53:315–374, 2015.

[19] M. Bienvenu and M. Thomazo. On the complexity of evaluating regular path queries over linear existential rules. In M. Ortiz and S. Schlobach, editors, *Web Reasoning and Rule Systems - 10th International Conference, RR 2016, Aberdeen, UK, September 9-11, 2016, Proceedings*, volume 9898 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2016.

[20] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009.

[21] C. Bourgaux, P. Bourhis, L. Peterfreund, and M. Thomazo. Revisiting semiring provenance for datalog. In G. Kern-Isberner, G. Lakemeyer, and T. Meyer, editors, *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022*, 2022.

[22] C. Bourgaux, D. Carral, M. Krötzsch, S. Rudolph, and M. Thomazo. Capturing homomorphism-closed decidable queries with existential rules. In M. Bienvenu, G. Lakemeyer, and E. Erdem, editors, *Proceedings of the 18th International Conference*

on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021, pages 141–150, 2021.

[23] C. Bourgaux, A. Ozaki, R. Peñaloza, and L. Predoiu. Provenance for the description logic elhr. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1862–1869. ijcai.org, 2020.

[24] M. Calautti, G. Gottlob, and A. Pieris. Non-uniformly terminating chase: Size and complexity. In L. Libkin and P. Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 369–378. ACM, 2022.

[25] M. Calautti, E. Livshits, A. Pieris, and M. Schneider. The complexity of why-provenance for datalog queries. *CoRR*, abs/2303.12773, 2023.

[26] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.

[27] A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Semant.*, 14:57–83, 2012.

[28] A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[29] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.*, 39(3):385–429, 2007.

[30] D. Calvanese, E. Kharlamov, W. Nutt, and C. Thorne. Aggregate queries over ontologies. In R. Elmasri, M. Doerr, M. Brochhausen, and H. Han, editors, *Proceedings of the 2nd International Workshop on Ontologies and Information Systems for the Semantic Web, ONISW 2008, Napa Valley, California, USA, October 30, 2008*, pages 97–104. ACM, 2008.

[31] D. Calvanese, D. Lanti, A. Ozaki, R. Peñaloza, and G. Xiao. Enriching ontology-based data access with provenance. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1616–1623. ijcai.org, 2019.

[32] D. Carral, I. Dragoste, and M. Krötzsch. Restricted chase (non)termination for existential rules with disjunctions. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 922–928. ijcai.org, 2017.

[33] D. Carral, I. Dragoste, M. Krötzsch, and C. Lewe. Chasing sets: How to use existential rules for expressive reasoning (extended abstract). In M. Simkus and G. E. Weddell, editors, *Proceedings of the 32nd International Workshop on Description Logics, Oslo, Norway, June 18-21, 2019*, volume 2373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[34] D. Carral, L. Larroque, M. Mugnier, and M. Thomazo. Normalisations of existential rules: Not so innocuous! In G. Kern-Isberner, G. Lakemeyer, and T. Meyer, editors,

Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022, 2022.

[35] S. A. Cook and P. McKenzie. Problems complete for deterministic logarithmic space. *J. Algorithms*, 8(3):385–394, 1987.

[36] A. Dawar and S. Kreutzer. On datalog vs. LFP. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 2008.

[37] J. Edmonds. Paths, trees and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[38] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[39] T. Feder and M. Y. Vardi. Homomorphism closed vs. existential positive. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 311–320, 2003.

[40] C. Feier, C. Lutz, and M. Przybylko. Answer counting under guarded tgds. In K. Yi and Z. Wei, editors, *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, volume 186 of *LIPIcs*, pages 11:1–11:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[41] D. Figueira. Foundations of graph path query languages - course notes for the reasoning web summer school 2021. In M. Simkus and I. Varzinczak, editors, *Reasoning Web. Declarative Artificial Intelligence - 17th International Summer School 2021, Leuven, Belgium, September 8-15, 2021, Tutorial Lectures*, volume 13100 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2021.

[42] D. Florescu, A. Y. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In A. O. Mendelzon and J. Paredaens, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, pages 139–148. ACM Press, 1998.

[43] N. Francis, A. Gheerbrant, P. Guagliardo, L. Libkin, V. Marsault, W. Martens, F. Murlak, L. Peterfreund, A. Rogova, and D. Vrgoc. A researcher's digest of GQL (invited talk). In F. Geerts and B. Vandevoort, editors, *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, volume 255 of *LIPIcs*, pages 1:1–1:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[44] T. Gogacz, J. Marcinkowski, and A. Pieris. All-instances restricted chase termination. In D. Suciu, Y. Tao, and Z. Wei, editors, *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, pages 245–258. ACM, 2020.

[45] T. Gogacz, J. Marcinkowski, and A. Pieris. All-instances restricted chase termination for linear tgds. *Künstliche Intell.*, 34(4):465–473, 2020.

[46] G. Gottlob, M. Manna, and A. Pieris. Polynomial combined rewritings for existential rules. In C. Baral, G. D. Giacomo, and T. Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press, 2014.

[47] G. Gottlob and C. H. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 183(1):104–122, 2003.

[48] G. Gottlob and C. H. Papadimitriou. On the complexity of single-rule datalog queries. *Inf. Comput.*, 183(1):104–122, 2003.

[49] G. Grahne and A. Onet. Anatomy of the chase. *Fundam. Informaticae*, 157(3):221–270, 2018.

[50] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In L. Libkin, editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM, 2007.

[51] M. Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 329–346. Springer, 2008.

[52] A. Hernich and P. G. Kolaitis. Foundations of information integration under bag semantics. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

[53] N. Immerman. Relational queries computable in polynomial time. *Inf. Control.*, 68(1-3):86–104, 1986.

[54] N. Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[55] B. Ketsman and C. Koch. Datalog with negation and monotonicity. In C. Lutz and J. C. Jung, editors, *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, volume 155 of *LIPIcs*, pages 19:1–19:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[56] E. V. Kostylev and J. L. Reutter. Complexity of answering counting aggregate queries over dl-lite. *J. Web Semant.*, 33:94–111, 2015.

[57] M. Krötzsch, M. Marx, and S. Rudolph. The power of the terminating chase (invited talk). In P. Barceló and M. Calautti, editors, *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[58] M. Leclère, M. Mugnier, M. Thomazo, and F. Ulliana. A single approach to decide chase termination on linear existential rules. In P. Barceló and M. Calautti, editors, *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*, pages 18:1–18:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[59] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic EL using a relational database system. In C. Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2070–2075, 2009.

[60] C. Lutz and F. Wolter. Non-uniform data complexity of query answering in description logics. In G. Brewka, T. Eiter, and S. A. McIlraith, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press, 2012.

[61] Q. Manière. *Counting queries in ontology-based data access. (Requêtes de comptage pour l'accès aux données en présence d'ontologies)*. PhD thesis, University of Bordeaux, France, 2022.

[62] L. Meijer. 3-coloring in time o(1.3217ˆn). *CoRR*, abs/2302.13644, 2023.

[63] A. O. Mendelzon and P. T. Wood. Finding regular simple paths in graph databases. *SIAM J. Comput.*, 24(6):1235–1258, 1995.

[64] M. Mugnier and M. Thomazo. An introduction to ontology-based query answering with existential rules. In M. Koubarakis, G. B. Stamou, G. Stoilos, I. Horrocks, P. G. Kolaitis, G. Lausen, and G. Weikum, editors, *Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, volume 8714 of *Lecture Notes in Computer Science*, pages 245–278. Springer, 2014.

[65] N. Ngo, M. Ortiz, and M. Šimkus. Closed predicates in description logics: results on combined complexity. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 237–246, 2016.

[66] C. Nikolaou, E. V. Kostylev, G. Konstantinidis, M. Kaminski, B. C. Grau, and I. Horrocks. Foundations of ontology-based data access under bag semantics. *Artif. Intell.*, 274:91–132, 2019.

[67] B. Pérez, J. Rubio, and C. Sáenz-Adán. A systematic review of provenance systems. *Knowledge and Information Systems*, 57:495–543, 2018.

[68] A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.

[69] O. Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, 2008.

[70] S. Rudolph and M. Thomazo. Characterization of the expressivity of existential rule queries. In Q. Yang and M. J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3193–3199. AAAI Press, 2015.

[71] S. Rudolph and M. Thomazo. Expressivity of datalog variants - completing the picture. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1230–1236. IJCAI/AAAI Press, 2016.

[72] P. Senellart. Provenance and probabilities in relational databases. *SIGMOD Rec.*, 46(4):5–15, 2017.

[73] M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[74] E. Tsamoura, D. Carral, E. Malizia, and J. Urbani. Materializing knowledge bases via trigger graphs. *Proc. VLDB Endow.*, 14(6):943–956, 2021.

[75] UKGov. Cgov: Central government ontology. an ontology of the uk central government.

[76] F. van Harmelen, V. Lifschitz, and B. W. Porter, editors. *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier, 2008.

[77] M. Y. Vardi. The complexity of relational query languages (extended abstract). In H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 137–146. ACM, 1982.

[78] P. T. Wood. Query languages for graph databases. *SIGMOD Rec.*, 41(1):50–60, 2012.

[79] H. Zhang, Y. Zhang, and J. You. Existential rule languages with finite chase: Complexity and expressiveness. *CoRR*, abs/1411.5220, 2014.

RÉSUMÉ

Répondre à des requêtes en présence d'une ontologie est une approche étudiée depuis une quinzaine d'années pour faciliter l'intégration de données. L'intérêt est de permettre aux utilisateurs d'exprimer leurs requêtes en utilisant un vocabulaire familier, tout en faciliant l'intégration de données hétérogènes et en complétant les réponses à l'aide de la prise en compte de la sémantique des termes utilisés. Les frontières de la décidabilité, les études de complexité et les algorithmes développés se sont concentrés sur le cadre où des requêtes conjonctives booléennes sont utilisées pour interroger une base de connaissance.

Dans ce manuscrit, nous explorons les limites de l'expressivité de ce cadre, et proposons quelques pistes d'extension pertinentes d'un point de vue applicatif. L'expressivité est étudiée sous le prisme de la complexité descriptive, alors que les extensions proposent de sortir du cadre des requêtes conjonctives booléennes, soit en permettant des requêtes à base de chemins, soit en sortant du cadre booléen, en apportant des informations de comptage ou des méta-données sur la provenance des réponses.

ABSTRACT

Answering queries while taking an ontology into account has been studied over the last fifteen years to ease data integration. The goal is to allow users to express their queries in a vocabulary they are familiar with, while facilitating data integration and providing more complete sets of answers thanks to the exploitation of the semantics of the query terms. Decidability, complexity and algorithms developed so far have focused on the case where Boolean conjunctive queries are used to query a so-called knowledge base.

In this manuscript, we explore the limitations of this setting, and suggest a few extensions that are relevant from an applicative point of view. Expressivity is studied through the prism of descriptive complexity, while the proposed extensions escape from the Boolean conjunctive query framework, by either allowing path queries, or by going beyond the Boolean case, by supporting counting features or providing meta-data about the provenance of an answer.