# Data Access With Horn Ontologies: Where Description Logics Meet Existential Rules

Marie-Laure Mugnier

**Noname manuscript No.**
(will be inserted by the editor)

# Data Access With Horn Ontologies:

## Where Description Logics Meet Existential Rules

**Marie-Laure Mugnier**

Accepted to KI - Künstliche Intelligenz (special issue on Ontologies and Data Management), June 2020.
https://doi.org/10.1007/s13218-020-00678-3

**Abstract** Two main families of ontology languages are considered in the context of data access, namely Horn description logics and existential rules. In this paper, we review the semantic relationships between these families in the light of the ontology-mediated query answering problem. To this end, we rely on the standard translation of description logics in first-order logic and on the notion of semantic emulation. We focus on description logics and classes of existential rules for which the conjunctive query answering problem has polynomial data complexity.

**Keywords** Data Access · Ontology-Mediated Query Answering · Existential Rules · Horn Description Logics

### 1 Introduction

Intelligent methods to manage large, complex or heterogeneous datasets require domain knowledge, typically encoded in ontologies. In the last years, there has been a growing interest in the problem known as *ontology-mediated query answering*, which consists in exploiting ontological knowledge when querying data. In this framework, an ontology is added on top of data, which yields a knowledge base (see e.g., [**?**] for a high-level overview). By contrast to classical query evaluation against a database, query answering against a knowledge base takes into account inferences enabled by the ontology, as illustrated by Example **??**.

Marie-Laure Mugnier
University of Montpellier, LIRMM, Inria
E-mail: mugnier@lirmm.fr

*Example 1* As a simple illustration, let us consider a database about literature. Assume a specific individual $a$ is known in this database as a poet, which we note $Poet(a)$, without any further information about his work. A query on this database asking for authors of poems, like $q(x) = \exists y\ (author(x, y) \wedge Poem(y))$, will certainly not retrieve $a$. Adding the knowledge that poets are authors of poems, which can be expressed as the logical formula $\forall x\ (Poet(x) \rightarrow \exists y\ (author(x, y) \wedge Poem(y)))$, allows inferring that $a$ is an answer to the query, even if no specific poem of $a$ is known, therefore palliating data incompleteness.

Two main families of ontology languages are considered in this context, namely description logics and existential rules. *Description Logics* (DLs) have been specially designed to represent and reason with ontologies [**?**]. They notably underlie the Semantic Web ontological language OWL [**?**,**?**]. A DL ontology contains axioms that express inclusions between concepts and between binary relations (called roles), and possibly other properties of roles (like symmetry or transitivity). Complex concepts and roles can be built using a set of constructors. E.g. the knowledge in Example **??** can be seen as the logical translation of the concept inclusion $Poet \sqsubseteq \exists author.Poem$, where $Poet$ is simply a concept name and $\exists author.Poem$ is a complex concept.

The expressiveness of a specific DL depends on the allowed set of constructors and shape of axioms. Historically, description logics focused on so-called standard reasoning tasks, like checking the satisfiability of a knowledge base or classifying concepts. When DL-based data access began to be investigated, it turned out that answering basic database queries, i.e., conjunctive queries, had a very high complexity with classical DLs (e.g., it is 2ExpTime-complete for $\mathcal{ALCI}$ and its

popular extension $\mathcal{SHIQ}$ [?,?]). For the DL $\mathcal{SROIQ}$, which underlies OWL 2, the lastest version of OWL [?], it is even not known if conjunctive query answering is decidable (see e.g., [?]). Hence, research on ontology-mediated query answering studied less expressive DLs. This includes new *lightweight DLs*, like the DL-Lite family, specially tailored for efficient query answering [?] and the $\mathcal{EL}$ family introduced to deal with large ontologies [?,?], as well as fragments of classical DLs, like Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SROIQ}$ [?,?,?]. All these DLs are often referred to as *Horn description logics* because they have the common property of being expressible in the Horn fragment of first-order logic.

On the other hand, *existential rules* have emerged in the last ten years as a new ontological language directed towards data access. Existential rules are a fragment of first-order logic that can be seen as an extension of datalog, the query language of deductive databases, to which they bring the capability of inferring the existence of unknown individuals [?,?]. For instance, the knowledge in Example ?? is an existential rule, which expresses that every poet is author of a poem, even if this poem may be unknown. Such a feature, which is offered by description logics, is considered as crucial for reasoning in an open-world perspective, where it cannot be assumed that all existing individuals are known in advance and stored in the database. Another line of work introduced existential rules as the logical translation of graph-based rules [?,?].

Generally speaking, existential rules and description logics are incomparable in terms of expressivity. However, the existential rule framework generalizes *Horn* description logics. More specifically, it extends these latter by two main features. First, predicates in rules have unrestricted arity, while DLs are usually restricted to unary and binary predicates. Unrestricted arity allows for a natural coupling with relational database schemas in which relations may have any arity, an important feature in the context of data access. It also provides greater flexibility; in particular, contextual information, such as data provenance, can be easily taken into account by adding new predicate arguments. Second, arbitrarily complex relationships between objects can be expressed, whereas DL axioms essentially express tree-like relationships. Unsurprisingly, greater expressivity comes with undecidability of query answering, even for the simplest queries (see e.g., [?] in a database context). However, a wide range of decidable classes of existential rules have been defined, which provide different tradeoffs between expressivity and tractability of query answering.

The purpose of this paper is to take a closer look at the relationships between decidable fragments of existential rules and Horn description logics. Our aim here is not to be exhaustive but rather to exhibit common and distinguishing characteristics of both families of formalisms. We will focus more specifically on rule classes and DLs for which the complexity of (conjunctive) query answering scales polynomially in the size of the data (i.e., is polynomial for data complexity), as this is a desired behavior in the context of data access.

Since description logics have their own syntax, we will rely on their standard translation in first-order logic to compare them with the existential rule framework. This will allow us to identify decidable rule classes that naturally cover some Horn DLs (in a sense that we will specify) and discuss the relative complexities of query answering in these fragments. As we shall see, some expressive Horn DLs are not covered by decidable rule classes already investigated in the literature. We will pinpoint the difficulties in extending relevant rule classes with knowledge constructs offered by these DLs.

We assume that the reader has basic knowledge on classical first-order logic and description logics, even if we will recall some notions and notations to make the paper self-contained. For a comprehensive introduction to query answering techniques associated with Horn description logics, we refer the reader to the survey chapters [?,?] and to the recent textbook [?]. Introductions to the existential rule framework can be found in [?,?,?,?].

The paper is organized as follows. After introducing preliminary notions, we give overviews of the existential rule framework and Horn description logics. Concerning existential rules, we will describe in more detail the *greedy bounded-treewidth* family because of its relevance to our aim [?,?,?]. Of particular interest are the subclasses of *guarded* [?], *frontier-one* [?] and *frontier-guarded* rules [?], for which the query answering problem is polynomial for data complexity [?,?,?]. Concerning Horn description logics, we will specifically focus on the lightweight DL-Lite and $\mathcal{EL}$ families, as well as on the more expressive Horn-$\mathcal{SHIQ}$ description logic, in which the query answering problem is still polynomial for data complexity (note that the survey chapter [?] is specifically devoted to these description logics). Finally, we will compare relevant fragments of both ontological languages from an expressivity and complexity viewpoint.

## 2 Preliminaries

This section recalls some basic logical notions and introduces fundamental notions about knowledge bases and query answering.

### 2.1 Logical notions

We consider classical first-order logic. A logical *vocabulary* (also called signature) is composed of finite sets of predicates and function symbols. We recall that constants are 0-ary function symbols. An *atom* is of the form $p(t_1 \dots t_k)$ where $p$ is a predicate with arity $k$ and the $t_i$ are *terms*, i.e., variables, constants, or complex functional terms. Given a formula or set of atoms $F$, we denote by $var(F)$ its set of variables, and $term(F)$ its set of terms. A formula is *atomic* if it has a single atom. A *ground* atom has no variables. A *literal* is an atom or the negation of an atom. A formula is *closed* if it has no free variables. Given a sequence of distinct variables $\mathbf{x} = x_1 \dots x_n$, we use the notations $\forall \mathbf{x}$ and $\exists \mathbf{x}$ as shortcuts for $\forall x_1, \dots, \forall x_n$ and $\exists x_1, \dots, \exists x_n$, respectively. Given a formula $F$ without quantifiers, the *universal* (respectively *existential*) closure of $F$ is the closed formula $\forall \mathbf{x}\ F$ (respectively, $\exists \mathbf{x}\ F$), where $\mathbf{x}$ is an ordering of $var(F)$. A *theory* is a finite set of closed formulas. A *clause* is the universal closure of a disjunction of literals. A *Horn clause* is a clause with at most one positive literal and a *definite clause* is a Horn clause with exactly one positive literal. A closed formula is in *clausal form* if it is a conjunction of clauses. We consider the classical notions of an *interpretation* of a vocabulary, denoted by $(\Delta_I, .^I)$, where $\Delta_I$ is the domain of $I$ and $.^I$ the interpretation function of $I$, of a *model* of a formula and of (semantic) *entailment*, denoted by $\models$ (given formulas $f$ and $g$, $f \models g$ means that every model of $f$ is a model of $g$).

### 2.2 Knowledge bases

From a conceptual viewpoint, an ontology expresses general knowledge about a domain, defined on a vocabulary given by sets of concept names, relation names, and possibly individuals. Relation names of binary arity are also called properties (e.g., in a Semantic Web setting) or role names (in description logics). From a logical viewpoint, a concept name $C$ is a unary predicate (also denoted by $C$), a relation name $r$ is a predicate of the same arity (also denoted by $r$), and an individual is a constant (denoted in the same way). Hence, the logical vocabulary associated with a given ontology is composed of predicates and constants (however, function symbols of arity greater than one may be introduced by the operation called Skolemization, see later). Generally speaking, an ontology can be seen as a theory composed of formulas of the form $\forall \mathbf{x}$ (*condition* $\rightarrow$ *conclusion*), where $\mathbf{x}$ denotes the set of variables that occur in both *condition* and *conclusion*. In the existential rule framework, an ontology is directly given as a set of such formulas, while in description logics, an ontology (called a TBox) has a standard translation into such formulas. A *fact* is a ground atom on constants and a *database instance* (or simply: instance) is a finite set of facts. A *knowledge base (KB)* is a pair composed of an ontology and an instance, on the same vocabulary. The *theory associated with a KB* is the union of its ontology and its instance.

We will need the notion of an *extended instance*, in which the atoms may contain variables denoting unknown entities. Such variables are called *nulls*, a name coming from databases. The logical formula assigned to an extended instance is the existential closure of the conjunction of its atoms: e.g., $\{p(x, a), q(x), p(b, y)\}$, where $x$, $y$ are variables and $a$,$b$ constants, corresponds to the formula $\exists x \exists y\ (p(x, a) \wedge q(x) \wedge p(b, y))$.

The *unique name assumption* (UNA), which states that distinct constants necessarily denote distinct individuals, is generally made in the framework of existential rules, and less often in description logics. Whether this assumption is made or not is only relevant when equalities / inequalities are involved.

### 2.3 Semantic emulation

To compare ontological fragments, we will sometimes transform a theory $T_1$ on a vocabulary $V_1$ into a theory $T_2$ on a vocabulary $V_2$ that extends $V_1$ by fresh predicates or function symbols, while preserving the semantics of $T_1$. Strictly speaking, we will not be able to say that these theories are logically equivalent because they are not defined on the same vocabulary, hence they do not have the same models. To specify their relationship, we will rely on the notion of semantic emulation [**?**]. We say that a vocabulary $V_2$ *extends* a vocabulary $V_1$ (notation $V_1 \subseteq V_2$) if the set of predicates (respectively of function symbols) of $V_1$ is included into that of $V_2$. Given two logical theories $T_1$ and $T_2$ on vocabularies $V_1$ and $V_2$ such that $V_1 \subseteq V_2$, we say that $T_2$ *semantically emulates* $T_1$ if (1) every model of $T_2$ becomes a model of $T_1$ when restricted to the interpretation of the symbols of $V_1$, and (2) every model $I_1$ of $T_1$ can be extended into a model $I_2$ of $T_2$ that has the same domain as $I_1$ and agrees with $I_1$ on $V_1$. This notion of semantic emulation is close to the classical notion of *conservative extension*, for which slightly different definitions can be found (for instance, the additional constraint that $T_1 \subseteq T_2$ is often enforced). See the DL texbook [**?**] for a definition of a conservative extension that corresponds to the above semantic emulation for DLs specifically. When $T_2$ is a semantic emulation of $T_1$, both theories are equisatisfiable (i.e., both satisfiable or both unsatisfiable), furthermore they entail exactly the same for-

mulas on $V_1$. In particular, *Skolemization* is a classical process that transforms any closed formula into a formula in (or equivalent to) a clausal form. This process replaces all existentially quantified variables with functional terms, using a fresh set of function symbols. Applied to a theory $T$, Skolemization produces a theory $Skolem(T)$ that is a semantic emulation of $T$.

### 2.4 Query answering

We consider conjunctive queries, which are the basic queries in relational or Semantic Web databases. A *conjunctive query* (CQ) is of the form $q(\mathbf{x}) = \exists \mathbf{y}\ \phi[\mathbf{x}, \mathbf{y}]$, where $\phi$ is a conjunction of atoms whose terms are constants or variables, and $\mathbf{x} \cup \mathbf{y} = var(\phi)$ (see Example **??**); the free variables in $\phi$ (i.e., $\mathbf{x}$) are called answer variables. A *Boolean conjunctive query* (BCQ) has no free variables. An instance $\mathcal{I}$ answers positively to a BCQ $q$ if $\mathcal{I} \models q$. More generally, a tuple of constants $(c_1 \dots c_k)$ is an answer to a CQ $q(x_1 \dots x_k)$ in $\mathcal{I}$ if $\mathcal{I}$ entails the BCQ obtained from $q$ by substituting each $x_i$ with $c_i$, i.e., $\mathcal{I} \models q[c_1/x_1 \dots c_k/x_k]$. Without loss of generality, we will restrict our focus to BCQs in the following.

Given sets of atoms $A_1$ and $A_2$, a *homomorphism* $h$ from $A_1$ to $A_2$ is a substitution from $var(A_1)$ to $term(A_2)$ such that $h(A_1) \subseteq A_2$. It is well-known that when $A_1$ and $A_2$ stand for two existentially closed conjunctions of atoms, $A_1 \models A_2$ if and only if there is a homomorphism from $A_2$ to $A_1$. In particular, for a BCQ $q$ and an instance $\mathcal{I}$, it holds that $\mathcal{I} \models q$ if and only if there is a homomorphism from (the set of atoms of) $q$ to $\mathcal{I}$. Similarly, a tuple of constants $(c_1 \dots c_k)$ is an answer to a CQ $q(x_1 \dots x_k)$ in $\mathcal{I}$ if and only if there is a homomorphism $h$ from $q(x_1 \dots x_k)$ to $\mathcal{I}$ such that $h(x_i) = c_i$ for all $i$.

The notion of homomorphism can also be defined on interpretations: given two interpretations $I$ and $J$ of the same vocabulary, a homomorphism $h$ from $I$ to $J$ is a mapping from $\Delta_I$ to $\Delta_J$ such that $h(p^I) \subseteq p^J$ for each predicate $p$ and $h(f^I) = f^J$ for each function symbol $f$ (note that function symbols other than constants are not relevant for conjunctive queries). An important property of BCQs is that they are "closed under homomorphism", which means that for any BCQ $q$, if an interpretation $I$ is a model of $q$, then every interpretation $J$ to which $I$ homomorphically maps is also a model of $q$. Indeed, an interpretation $I = (\Delta_I, .^I)$ is a model of $q$ if there is *match* of $q$ in $I$, i.e., a mapping $\pi$ from $term(q)$ to $\Delta_I$ such that $\pi(c) = c^I$ for each constant $c$ occurring in $q$ and $\pi(t_1, \dots, t_k) \in p^I$ for each atom $p(t_1, \dots, t_k) \in q$. [1] Given a match $\pi$ of $q$ in $I$ and a homomorphism $h$ from $I$ to an interpretation $J$, the mapping $h \circ \pi$ is a match of $q$ in $J$.

Finally, the *query answering* problem (QA) we consider is the following: given a BCQ $q$ and a KB $\mathcal{K}$, does the theory associated with $\mathcal{K}$ entail $q$ (which we denote by $\mathcal{K} \models q$)?

We will distinguish between two complexity measures for this problem: *combined* complexity, where $\mathcal{K}$ and $q$ are part of the problem input, and *data* complexity, where $q$ and the ontology are supposed to be fixed, and only the instance is part of the input. This second complexity measure allows one to focus on the data, with the assumption that the sizes of the query and the ontologies are small with respect to the size of the instance. With respect to combined complexity, we will in turn distinguish two cases, depending on whether the predicate arity is unbounded or bounded. This distinction is particularly relevant for a comparison of existential rules with description logics, since concepts and roles in DLs correspond to unary and binary predicates.

## 3 Existential Rules

In this section, we will first present the main ingredients of the existential rule framework, then provide an overview of decidable classes of rules, with a special focus on the "greedy bounded-treewidth" family.

### 3.1 The existential rule framework

In the existential rule framework, an ontology may contain three kinds of formulas, as illustrated by Example **??**: existential rules, negative constraints or equality rules. We will use the name *rule* to denote any of these three constructs.

*Example 2* To ease comparison with description logics, only unary and binary predicates are used in the following rules. Universal quantifiers are omitted. The first three rules are existential rules. Rule 4 is a negative constraint. Rule 5 is an equality rule.

1. Authors of a common thing are co-authors:
   $author(x, z) \wedge author(y, z) \rightarrow coauthors(x, y)$
2. Coauthors who are researchers are authors of a common research paper:
   $Researcher(x) \wedge Researcher(y) \wedge coauthors(x, y) \rightarrow \exists z\ author(x, z) \wedge author(y, z) \wedge ResearchPaper(z)$

[1] If we make the simplifying assumption that $I$ interprets constants by themselves, a match of $q$ in $I$ can be seen as a homomorphism from $q$ to the instance naturally associated with $I$.

3. The author of an impactful research paper is a happy researcher:
   $author(x, y) \wedge ResearchPaper(y) \wedge hasImpact(y, z) \wedge HighImpact(z) \rightarrow HappyResearcher(x)$
4. One cannot review a research paper submitted by a coauthor:
   $submitted(x, y) \wedge ResearchPaper(y) \wedge coauthor(x, z) \wedge reviewer(z, y) \rightarrow \bot$
5. Books have at most one ISBN:
   $Book(x) \wedge hasISBN(x, y) \wedge hasISBN(x, z) \rightarrow y = z$

Generally, an existential rule is a closed first-order formula $R = \forall \mathbf{x} \forall \mathbf{y} \big(B[\mathbf{x}, \mathbf{y}] \rightarrow \exists \mathbf{z}\ H[\mathbf{x}, \mathbf{z}]\big)$ where $B$ and $H$, respectively called the *body* and the *head* of $R$, also denoted by $body(R)$ and $head(R)$, are finite conjunctions of atoms on constants and variables, and $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are disjoint sets of variables. Variables $\mathbf{z}$, which occur only in $H$, are called the *existential* variables of $R$. Variables $\mathbf{x}$, which are shared between $B$ and $H$, are called the *frontier* of $R$. The name existential rules is usually restricted to rules on standard predicates, we will follow this usage. Besides, two kinds of rules are often considered, namely negative constraints and equality rules. A *negative constraint* has a head restricted to the special symbol $\bot$ (the "absurd" symbol, which has no model), i.e., it is of the form $C = \forall \mathbf{x}\ (B[\mathbf{x}] \rightarrow \bot)$. Note that this formula is equivalent to $\forall \mathbf{x}\ \neg B[\mathbf{x}]$. An *equality rule* has a head restricted to an equality, i.e., it is of the form $R_e = \forall \mathbf{x}\ \big(B[\mathbf{x}] \rightarrow x_1 = x_2\big)$, where $x_1$ and $x_2$ are variables from $\mathbf{x}$.[2] A typical use of negative constraints and equality rules is to assert, respectively, the disjointness of concepts (or relations), e.g., $C_1(x) \wedge C_2(x) \rightarrow \bot$, and the functionality of binary relations, e.g., $r(x, y) \wedge r(x, z) \rightarrow y = z$. However, they can express much more, since the shape of their body is not limited.

These three kinds of rules have long been studied in database theory, where they represent integrity constraints. In this context, existential rules and equality rules are respectively known as TGDs (tuple-generating dependencies) and EGDs (equality-generating dependencies) [**?**]. More general frameworks may consider existential rules with default negation, e.g., [**?**,**?**], or with disjunction in rule heads [**?**,**?**], which we will not consider here.

A KB is a pair $\mathcal{K} = (\mathcal{R}, \mathcal{I})$, where $\mathcal{R}$ is a set of rules and $\mathcal{I}$ is an instance. Hence, the QA problem in this framework takes as input a KB $\mathcal{K} = (\mathcal{R}, \mathcal{I})$ and a BCQ $q$, and asks whether $\mathcal{R} \cup \mathcal{I} \models q$.

[2] The framework in [**?**] considers more general equality rules, where $x_1$ and $x_2$ may also be constants.

In the following, it is assumed that distinct rules in $\mathcal{R}$ have disjoint sets of variables, even if we reuse variables in examples for the sake of simplicity. The basic operation on rules is the application of a rule to an instance. A rule $R$ is *applicable* to an instance $\mathcal{I}$ if there is a homomorphism $h$ from $body(R)$ to $\mathcal{I}$. The pair $(R, h)$ is called a *trigger* on $\mathcal{I}$. The application of $R$ according to $h$ (or: the application of $(R, h)$) produces a set of atoms obtained from $head(R)$ by replacing each frontier variable $x$ with $h(x)$ and each existential variable with a fresh variable, called a *null*. We denote by $h^{\text{safe}}$ this extension of $h$ that "safely" renames existential variables, so that distinct applications of the same rule produce disjoint sets of nulls. The (extended) instance resulting from the application of $(R, h)$ to $\mathcal{I}$ is $\mathcal{I} \cup h^{\text{safe}}(head(R))$.

*Example 3* Let $\mathcal{I} = \{Researcher(a), Researcher(b), Researcher(c), coauthors(a, b), coauthors(a, c)\}$ and $R_2$ be the second rule from Example **??**: $R_2$ can be applied to $\mathcal{I}$ according to two homomorphisms, $h_1 = \{x \mapsto a, y \mapsto b\}$ and $h_2 = \{x \mapsto a, y \mapsto c\}$. Assume the trigger $(R_2, h_1)$ is first applied: this produces the atoms $author(a, z_0)$, $author(b, z_0)$, $ResearchPaper(z_0)$, where $z_0$ is a fresh variable. The application of the trigger $(R_2, h_2)$ produces the atoms $author(a, z_1)$, $author(b, z_1)$, $ResearchPaper(z_1)$, where $z_1$ is another fresh variable.

The application of a negative constraint produces $\bot$, which shows that the KB is unsatisfiable. The application of an equality rule produces an atom of the form $t_1 = t_2$, where each $t_i$ is a constant or a null; assume UNA is made and $t_1 \neq t_2$: if $t_1$ and $t_2$ are two constants, the KB is unsatisfiable; otherwise, instead of actually adding the produced atom to the instance, one usually performs the corresponding substitution of terms in the instance.

### 3.2 The chase

The fundamental tool for reasoning in this framework is a forward chaining procedure known as the *chase*. The chase enriches the given instance by applying rules until a fixpoint is reached. This process may be infinite, as for instance with $\mathcal{R} = \{h(x) \rightarrow \exists z\ p(x, z) \wedge h(z)\}$ ("every human has a parent who is a human") and $\mathcal{I} = \{h(a)\}$. Formally, an $\mathcal{R}$-*derivation* from $\mathcal{I}$ is a possibly infinite sequence of extended instances and triggers $\mathcal{I}_0 (= \mathcal{I})\ (R_1, h_1)\ \mathcal{I}_1 \ldots\ (R_n, h_n)\ \mathcal{I}_n, \ldots$, where, for $i \geq 1$, $R_i \in \mathcal{R}$, $(R_i, h_i)$ is a trigger on $\mathcal{I}_{i-1}$, $\mathcal{I}_i$ results from the application of $(R_i, h_i)$ to $\mathcal{I}_{i-1}$, and no trigger occurs twice in this sequence. A *chase sequence* on $\mathcal{K} = (\mathcal{R}, \mathcal{I})$ is an $\mathcal{R}$-derivation from $\mathcal{I}$ that cannot be

extended, i.e., for any $\mathcal{I}_i$, $i \geq 0$, all triggers for $\mathcal{I}_i$ occur in the derivation. The *chase* is the set of atoms obtained in the limit of such a chase sequence, i.e., $\bigcup_{i\geq 0} \mathcal{I}_i$. Note that the name chase is used to denote both the forward chaining process and its result. The chase *terminates* on $\mathcal{K}$ if there is a finite chase sequence on $\mathcal{K}$.

Negative constraints or equality rules may lead to an unsatisfiable KB, while a KB that contains only existential rules is always satisfiable. For a satisfiable KB $\mathcal{K}$, the (output of the) chase can be seen as a logical interpretation $I$, which is a model of $\mathcal{K}$: the domain of $I$ is the set of terms in the chase, and its interpretation function is defined by the atoms of the chase (i.e., for each constant $c$, $c^I = c$ holds, and, for each predicate $p$, $p^I$ is the set of all tuples $(t_1, \dots t_k)$ such that $p(t_1, \dots t_k)$ belongs to the chase). This model of $\mathcal{K}$ has the fundamental property of being a *universal* model of $\mathcal{K}$, i.e., a model of $\mathcal{K}$ that homomorphically maps to any other model of $\mathcal{K}$ [**?**]. It follows that it can be used as a canonical model to answer conjunctive queries, and more generally all kinds of queries that are closed under homomorphism. Indeed, for any such Boolean query $q$ and any satisfiable KB $\mathcal{K}$, it holds that $\mathcal{K} \models q$ iff $M^{\mathcal{K}}$ is a model of $q$, where $M^{\mathcal{K}}$ is any universal model of $\mathcal{K}$; in particular, if $q$ is a BCQ, $\mathcal{K} \models q$ if and only if $q$ homomorphically maps to the chase of $\mathcal{K}$.

Actually, several variants of the chase have been defined in the literature (for a presentation of the main ones, see in particular [**?**]). The above chase is the simplest chase, known as the naive or *oblivious* chase [**?**]. More sophisticated chase variants try to avoid some (or even all) redundancies that can be introduced by nulls. Hence, they compute subsets of the oblivious chase, which are still universal models. To illustrate, let us consider the following example.

*Example 4* Let $\mathcal{I} = \{Researcher(a), Researcher(b), author(a, c), author(b, c), ResearchPaper(c)\}$ and $R_1$, $R_2$ be two first rules from Example **??**. Assume $R_1$ is first applied, leading to $\mathcal{I}_1 = \mathcal{I} \cup \{coauthor(a, b)\}$. Then $R_2$ can be applied, which leads to $\mathcal{I}_2 = \mathcal{I}_1 \cup \{author(a, z_0), author(b, z_0), ResearchPaper(z_0)\}$,with $z_0$ a null. However, $\mathcal{I}_2$ is logically equivalent to $\mathcal{I}_1$: indeed, it homomorphically maps to $\mathcal{I}_1$ (with $z_0$ being mapped to $c$) and the converse is trivially true as $\mathcal{I}_1 \subseteq \mathcal{I}_2$. Hence, the second rule application only adds redundant atoms. A cleverer chase variant like the *restricted* chase [**?**] would avoid it.

Crucially, the power of detecting redundancies is directly related to the power of terminating, as illustrated next (Example **??**). The most powerful chase, known as the *core* chase [**?**], maintains at each step the extended instance as a core, i.e., as a set of atoms that does not homomorphically map to any of its strict subsets.[3] A fundamental property of the core chase is that it terminates on a (satisfiable) KB if and only if this KB has a finite universal model [**?**]. However, keeping a core is costly,[4] hence the other chase variants achieve tradeoffs between redundancy elimination and computational efficiency.

*Example 5* Let $\mathcal{I} = \{p(a, b), r(b)\}$ and $\mathcal{R} = \{R_1, R_2\}$ with $R_1 = p(x, y) \rightarrow \exists z\ p(y, z)$ and $R_2 = r(x) \wedge p(x, y) \rightarrow p(x, x)$. Note that $R_2$ is not applicable to $\mathcal{I}$. The oblivious chase does not terminate on this KB, since $R_1$ can be applied indefinitely. The core chase outputs the finite instance $\mathcal{I} \cup \{p(b, b)\}$, obtained by first applying the trigger $(R_1, \{x \mapsto a, y \mapsto b\})$, which produces $p(b, z_0)$, and at some later step the trigger $(R_2, \{x \mapsto b, y \mapsto z_0\}$, which produces $p(b, b)$: then, all atoms already produced become redundant with respect to $p(b, b)$. Among known chase variants, only the core chase is able to terminate on this example.

### 3.3 Alternative forms of existential rules

Existential rules are sometimes translated into specific logic-programming rules, corresponding to Horn clauses. This is done by Skolemization. More precisely, given an existential rule $R$ with frontier $\mathbf{x}$, a fresh function symbol $f_z^R$ of arity $|\mathbf{x}|$ is assigned to each existential variable $z$. Then, the Skolemization of $R$ is the rule obtained from $R$ by replacing each existential variable $z$ with the functional term $f_z^R(\mathbf{x})$. For instance, the Skolemization of the rule $R_2$ in Example **??** replaces $z$ with $f_z(x, y)$ (we omit the $R_2$ superscript), which yields the formula $\forall x \forall y\ (R(x) \wedge R(y) \wedge c(x, y) \rightarrow a(x, f_z(x, y)) \wedge a(y, f_z(x, y)) \wedge RP(f_z(x, y)))$ (here predicates have been abbreviated), equivalent to a clausal form with three definite clauses:

$$(\neg R(x) \vee \neg R(y) \vee \neg c(x, y) \vee a(x, f_z(x, y))),$$
$$(\neg R(x) \vee \neg R(y) \vee \neg c(x, y) \vee a(y, f_z(x, y))),$$
$$(\neg R(x) \vee \neg R(y) \vee \neg c(x, y) \vee RP(f_z(x, y))).$$

Moreover, a negative constraint directly yields a Horn (but not definite) clause, e.g., the rule $R_4$ from Example **??** yields the clause $(\neg s(x, y) \vee \neg RP(y) \vee \neg c(x, z) \vee \neg r(z, y))$, while an equality rule yields a definite clause.

[3] Note that the core chase is not defined when it does not terminate [**?**].

[4] First, computing the core of an extended instance is difficult. The associated decision problem is both NP-hard and coNP-hard, precisely DP-complete [**?**]. Second, the extended instance built by the core chase does not grow monotonically, which in practice does not allow to update it incrementally.

Hence, any ontology in the existential rule framework can be translated into a set of Horn clauses. As pointed out in Section **??**, *Skolem*($\mathcal{K}$), obtained from $\mathcal{K}$ by the Skolemization of $\mathcal{R}$, is a semantic emulation of $\mathcal{K}$. In particular, it preserves the entailment of BCQs, in which the Skolem symbols do not occur: given a BCQ $q$, $\mathcal{K} \models q$ holds if and only if *Skolem*($\mathcal{K}$) $\models q$ holds. The associated chase variant is known as the Skolem chase [**?**].

Independently, it is sometimes assumed that existential rules have an atomic head (i.e., restricted to a single atom), based on the fact that any existential rule can be decomposed into a set of atomic-head rules by adding a fresh predicate, whose arity is the number of variables in the rule head. For instance, the rule $R_2$ from Example **??** can be decomposed into four atomic-head rules, using a fresh ternary predicate $p_{R_2}$:

$$body(R_2) \rightarrow \exists z\ p_{R_2}(x, y, z),$$
$$p_{R_2}(x, y, z) \rightarrow a(x, z),$$
$$p_{R_2}(x, y, z) \rightarrow a(y, z),$$
$$p_{R_2}(x, y, z) \rightarrow RP(z).$$

The obtained KB is again a semantic emulation of the initial KB, hence it preserves entailment of BCQs. As a consequence of this property, the query answering problem remains undecidable with atomic-head existential rules. Note, however, that this translation can generally not be done in a bounded predicate arity setting (as the arity of the fresh predicate is not bounded).

### 3.4 The decidable landscape

Existential variables in rule heads associated with arbitrarily complex conjunctions of atoms in bodies and heads make query answering undecidable. Decidable classes are obtained by imposing syntactic restrictions on rules, which apply to each rule individually (like having a body restricted to a single atom) or to a set of rules (like forbidding cyclic interactions between rules). Most of these restrictions fall into three "abstract" classes, introduced in [**?**] to describe the behavior of associated query answering procedures.

The first abstract class, called *finite expansion sets* (fes), ensures that the core chase will halt on any instance. Finite expansion sets are exactly those sets of rules $\mathcal{R}$ for which $(\mathcal{R}, \mathcal{I})$ has a finite universal model for any instance $\mathcal{I}$. Hence, query answering can be solved by first running the (core) chase, then evaluating the query on the chase output, an approach called materialisation. More restricted classes can be defined to also ensure the termination of a weaker chase variant.

The second abstract class, called *finite unification sets* (fus), ensures that any BCQ $q$ can be rewritten using the rules into a union of BCQs $\mathcal{Q}$ (i.e., a finite disjunction of BCQs) such that: for any instance $\mathcal{I}$, it holds that $\mathcal{R}, \mathcal{I} \models q$ if and only if $\mathcal{I} \models \mathcal{Q}$. Hence, query answering can be solved by first rewriting the query, then evaluating the rewriting on the (unchanged) instance. The fus property is actually equivalent to the property called *first-order rewritability*,[5] which was first introduced in the context of DL-Lite [**?**]. Put in our setting: $\mathcal{R}$ is first-order rewritable if, for any BCQ $q$, there is a first-order query $q'$ (i.e., a first-order formula on variables and constants), such that $\mathcal{R}, \mathcal{I} \models q$ if and only if $\mathcal{I} \models q'$, for any instance $\mathcal{I}$.

The third class, called *bounded-treewidth sets* (bts), generalizes fes by accepting sets of existential rules that have a universal model that may not be finite but has a tree-like structure (formally, a bounded-treewidth). Contrarily to the two first classes, the bts class is not directly associated with a practical query answering algorithm. [6] However, an expressive subclass of bts was later defined, namely "greedy bounded-treewidth sets" (gbts), which comes with an effective query answering technique [**?**,**?**,**?**,**?**]. This class enjoys the following property: for any instance, there is a bound $b$, which depends on the set of rules and the instance, such that a tree decomposition of the chase of width $b$ can be built *greedily* (hence the name of the class). Then, a finite representation of the infinite chase can be built by detecting regularities in the chase.

Unsurprisingly, whether a set of rules is fes (or ensures the termination of some chase variant), fus or bts is an undecidable problem [**?**,**?**]. The question is open for gbts.

Before defining the gbts class and its "concrete" subclasses (that is, subclasses defined by syntactic conditions), let us briefly discuss the impact of *negative constraints* and *equality rules* on the decidability of query answering. Negative constraints can make a KB unsatisfiable, but have no influence of the query answering process itself. Moreover, a KB $(\mathcal{R}^+ \cup \mathcal{R}^-, \mathcal{I})$, where $\mathcal{R}$ is partitioned into positive rules ($\mathcal{R}^+$), i.e., existential and equality rules, and negative constraints ($\mathcal{R}^-$), is satisfiable if and only if $(\mathcal{R}^+, \mathcal{I})$ satisfies each constraint $C$ in $\mathcal{R}^-$, i.e., does not entail $body(C)$, which can be seen as a BCQ. Hence, the presence of negative constraints has no influence on the decidability of query answering, and does generally not increase its complexity. On the

[5] The equivalence between the rewritability into a union of CQs and first-order rewritability follows from the (Finite) Homomorphism Preservation Theorem [**?**].

[6] The decidability of query answering in the bts class follows from a theorem by Courcelle [**?**], which states that satisfiability is decidable for any class of first-order formulas enjoying the bounded-treewidth model property (i.e., satisfiability implies the existence of a model with a bounded-treewidth).

contrary, equality rules are a well-known cause of undecidability, in particular when added to (g)bts rules, as they may destroy the tree-like structure of the chase. A way of dealing with them is to enforce a *separability* condition between existential rules and equality rules, which essentially turns equality rules into constraints on the (initial) instance [**?**].

### 3.5 Greedy bounded-treewidth sets

The gbts class is defined by a simple condition: when a rule is applied during the chase, all frontier variables that are not mapped to constants[7] are jointly mapped to nulls introduced by a *single* previous rule application. Formally, an $\mathcal{R}$-derivation $\mathcal{I}_0(=\mathcal{I})\ (R_1, h_1)\ \mathcal{I}_1\ \ldots\ (R_n, h_n)\ \mathcal{I}_n$ is said to be *greedy* if, for all $i$ with $0 < i \leq n$, there is $0 \leq j < i$ such that:

$$h_i(\mathit{fr}(R_i)) \subseteq \mathit{var}(h_j^{\text{safe}}(\mathit{head}(R_j)) \cup \mathcal{C}$$

where $\mathit{fr}(R_i)$ is the frontier of $R_i$, $h_j^{\text{safe}}(\mathit{head}(R_j))$ is the set of atoms produced by $(R_j, h_j)$, and $\mathcal{C}$ is the set of constants occurring in $\mathcal{I}$ and $\mathcal{R}$. A set of existential rules $\mathcal{R}$ is a *greedy bounded-treewidth set* if any $\mathcal{R}$-derivation (from any instance) is greedy.

When the set of rules is gbts, the output of the chase can be decomposed into a tree (in the classical sense of a tree decomposition), whose root corresponds to the initial instance and each node corresponds to the atoms brought by a rule application. Moreover, this tree can be built in a greedy manner: each rule application creates a new node, which is attached as a child of the highest node in the tree that fulfills the condition on frontier variables.

Among types of rules often desired in knowledge modeling, some are well-known to endanger the gbts property. As already mentioned, this is the case of equality rules. This is also the case of existential rules that compose binary relations, such as *transitivity rules* (stating that a binary relation is transitive), as illustrated by the next example. The syntactic conditions associated with concrete gbts classes exclude such rules or constrain the way they can be applied to nulls, in order to preserve the tree-like structure of the chase.

*Example 6* Let $\mathcal{R} = \{R_1 : r(x) \rightarrow \exists z\ p(x,z),\ R_2 : s(x) \rightarrow \exists z\ p(z,x),\ R_3 : p(x,y) \wedge p(y,z) \rightarrow p(x,z)\}$, where the last rule expresses that $p$ is transitive. Let $\mathcal{I} = \{r(a), s(a)\}$. By applying $R_1$, which creates a null $z_0$ and produces the atom $p(a, z_0)$, then $R_2$, which creates another null $z_1$ and produces $p(z_1, a)$, and finally $R_3$, one obtains a nongreedy derivation: $\mathcal{I}_0 = \mathcal{I}\ (R_1, \{x \mapsto a\})\ \mathcal{I}_1\ (R_2, \{x \mapsto a\})\ \mathcal{I}_2\ (R_3, \{x \mapsto z_1, y \mapsto a, z \mapsto z_0\})\ \mathcal{I}_3$. Indeed, $x$ and $z$, the frontier variables of $R_3$, are mapped to nulls introduced by distinct rule applications. Hence, $\mathcal{R}$ is not gbts.

[7] Constants may occur in the initial instance but they may also be brought by the rules, since rule heads may contain constants.

### 3.6 Concrete gbts classes

We now review the main concrete gbts classes. These classes are pictured in Figure **??**. There are three basic classes, with each of them providing a different way of enforcing greedy bounded-treewidth: (plain) *datalog* (e.g., [**?**]), in which there are no existential variables at all; *guarded* rules (g), in which at least one body atom, called a guard, contains all the variables from the body [**?**,**?**]; *frontier-one* rules (fr1), in which the frontier of the rule is restricted to (at most) one variable [**?**]. An important subclass of guarded rules are *linear* rules, whose body is restricted to a single atom [**?**,**?**]. Notably, linear rules generalize relational database inclusion dependencies.

Combining guardedness and frontier-based restrictions leads to *frontier-guarded* rules (fg), in which only the frontier of the rule needs to be guarded: at least one body atom contains the rule frontier [**?**]. Beside, combining guardedness and conditions on how variables are mapped during the chase leads to *weakly-guarded* rules (wg), in which only so-called affected variables need to be guarded: a variable from a rule body is said to be *affected* if it is possibly mapped to a null during the chase (which can be recognized by a simple marking procedure on the ruleset); then a set of rules is weakly-guarded if in each rule an atom of the body contains (or: guards) all affected variables [**?**]. Note that weak-guardedness is a property of a *set of rules*, as the notion of affected variable requires taking into account interactions between rules. The most general member of the gbts family is *weakly-frontier guarded* (wfg) rules, which generalize both frontier-guarded and weakly-guarded rules: in each rule, an atom contains all the affected variables from the rule frontier [**?**]. This class essentially has the same expressivity as gbts itself, since each gbts KB can be semantically emulated by a wfg KB (note that the translation is polynomial but is not data-independent, as it requires to transform the instance as well) [**?**,**?**].

Finally, we mention restrictions of frontier-guarded rules obtained by imposing that the rule bodies are *acyclic*. These restrictions are relevant, because, if a description logic axiom can be logically translated into

a frontier-guarded rule, then the obtained rule generally falls into one of these body-acyclic classes. Here, the considered acyclicity notion is that of hypergraph-acyclicity, since the set of atoms composing the body of a rule can naturally be seen as a hypergraph (see e.g., [**?**]). Intuitively, a frontier-guarded rule is acyclic if its body can be decomposed into a rooted tree, in which a node may contain several atoms provided that this set of atoms has a guard, and the root of the tree contains the frontier of the rule. Importantly, guarded rules are a special case of body-acyclic frontier-guarded rules. Restrictions of frontier-guarded to rules with an acyclic body include the class of *body-acyclic frontier-guarded* rules (ba-fg), its direct subclasses *guarded* rules and *body-acyclic frontier-one* rules (ba-fr1), and the common specialization of these two classes, namely *guarded frontier-one* rules (g-fr1) [**?**,**?**].

*Example 7* Consider the three existential rules from Example **??**. The frontier of $R_1$ is $\{x, y\}$, hence $R_1$ is not frontier-guarded. Rules $R_2$ and $R_3$ are frontier-guarded. $R_2$ is more specifically guarded (as the atom $c(x, y)$ contains all the variables from the body), and $R_3$ is not guarded but its frontier is $\{x\}$, hence it is frontier-one, and more specifically body-acyclic frontier-one. Finally, the only affected variable in the rule set is $z$ in $R_1$ (because it can be mapped to a null created by $R_2$), but $\{z\}$ is guarded in $R_1$, hence the whole rule set is weakly-guarded, hence weakly-frontier guarded as well.

### 3.7 Complexity results for gbts classes

The data complexity of BCQ answering under gbts is pictured in Figure **??**. Rule classes are grouped into three complexity classes: $AC^0$ (a low subclass of PTime), PTime-complete and ExpTime-complete. Linear rules are first-order rewritable and query rewriting is independent from the data; hence, the data complexity of BCQ answering under linear rules is that of answering a union of BCQs on a relational database, i.e., $AC^0$. All classes up to frontier-guarded have polynomial data complexity. Syntactically, a guarded rule is a body-acyclic frontier-guarded rule, and, in turn, any body-acyclic frontier-guarded rule can be polynomially decomposed into a set of guarded rules that semantically emulates it, using fresh predicates [**?**]. The translation from ba-fg to guarded is without predicate arity increase, hence these two classes behave similarly for all complexity measures. As soon as the notion of guardedness of affected variables comes in (i.e., for wg, wfg, gbts), the data complexity raises to ExpTime-complete.

Combined complexities are not pictured in Figure **??** but given in Table **??** for existential rule classes directly relevant to main Horn DLs. Combined complexity with unbounded arity goes from PSpace-complete for linear rules, to ExpTime-complete for datalog and g-fr1, and 2ExpTime-complete for the other classes (fr1, g, fg, wg, wfg, gbts)[8]. When arity is bounded, combined complexity drops to NP-complete for linear rules and datalog, and to ExpTime-complete for guarded and wg rules; it remains ExpTime-complete for g-fr1 and 2ExpTime-complete for the other classes (fr1, fg, wfg, gbts). For proofs of these data and combined complexity results, see [**?**,**?**,**?**,**?**,**?**,**?**,**?**], with the last reference providing an updated picture of results on the gbts family.

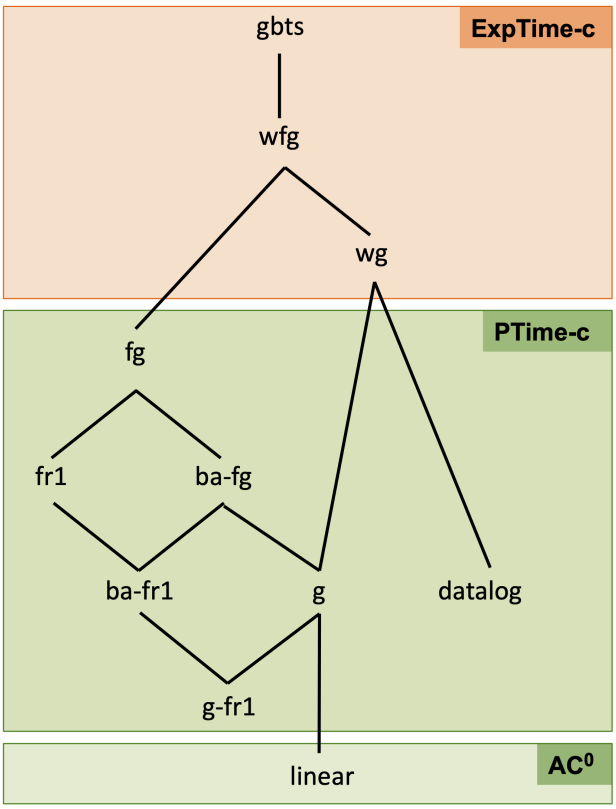


**Fig. 1** Greedy-bounded-treewidth classes ordered by syntactic inclusion and their data complexity

## 4 Horn Description Logics

This section is devoted to Horn description logics and popular members of this family. We will first recall basic notations about description logics and their standard translation in first-order logic.

[8] For ba-fr1, 2ExpTime-hardness is still open in the unbounded arity case, while ExpTime-completeness holds in the bounded arity case [**?**].

### 4.1 Description logics as first-order logic fragments

A DL KB is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$, the TBox, is composed of concept inclusions $C_1 \sqsubseteq C_2$, role inclusions $r_1 \sqsubseteq r_2$, and possibly other axioms asserting properties of roles (like transitivity), and $\mathcal{A}$, the ABox, contains factual assertions of the form $C(a)$ and $r(a, b)$, with $C$ a concept name, $r$ a relation name, and $a$, $b$ individuals (i.e., constants). Note that we do not allow for complex concepts and relations in ABoxes, contrarily to some description logics. However, a knowledge base can generally be put in a normal form such that no complex concept or relation appears in the ABox.

The constructors used in this paper and their standard translation in first-order logic are shown in Table **??** (see e.g., [**?**]). These constructors are those found in the classical DL $\mathcal{ALC}$ (concept conjunction, disjunction and negation; existential and universal restrictions) and the Horn DLs we consider (which leads to add number restrictions and several role constructors: inverse roles, role conjunction and negation).[9] Concept and role names are respectively viewed as unary and binary predicates. To define the formulas assigned to concepts and roles, we use translation functions, of the form $\phi_x(C)$ to translate a concept $C$ into a formula with free variable $x$ (expressing that $x$ is an instance of $C$), and $\phi_{x,y}(r)$ to translate a role $r$ into a formula with free variables $x$ and $y$ (expressing that $x$ is linked to $y$ by $r$). A concept name $C$ is translated into $C(x)$ and a role name $r$ into $r(x, y)$, then complex concepts and roles are inductively defined according to their structure. In the DL setting, $\top$ and $\bot$ are special concept names, respectively interpreted as the whole domain and the empty set in any interpretation of the vocabulary. The translation of TBox axioms (concept inclusions, role inclusions and transitivity axioms) is shown at the bottom of the table.

The semantics of DLs are usually defined in a model theory. Since concept and role names can be considered as predicates, and individuals as constants, an interpretation in the usual DL sense can be seen as an interpretation in first-order logic. Then a DL axiom and its translation have the same models. Finally, the QA problem is defined as in Section **??**, considering that the theory associated with a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is the theory obtained by the logical translation of $\mathcal{T}$ and $\mathcal{A}$.

[9] To simplify the discussion, we chose not to include nominals nor role composition (except for its restricted use in the transitivity axiom can be seen as a specific which could be written $r \circ r \sqsubseteq r$).

### 4.2 Horn DLs: avoiding disjunctive reasoning

Intuitively, the characteristic of a Horn DL is that it is unable express disjunction in the right hand side of inclusions ($C_1 \sqsubseteq C_2 \sqcup C_3$), or equivalently, negation in the left hand side of inclusions ($C_1 \sqcap \neg C_2 \sqsubseteq C_3$). This restriction makes reasoning generally simpler, as disjunction leads to reasoning by cases. Indeed, whereas $C_1 \sqcup C_2 \sqsubseteq C_3$ is equivalent to the simpler inclusions $C_1 \sqsubseteq C_3$ and $C_2 \sqsubseteq C_3$, the converse inclusion $C_3 \sqsubseteq C_1 \sqcup C_2$ cannot be rewritten without using $\sqcup$ or $\neg$.

*Example 8* Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{A} = \{A(a)\}$ and $\mathcal{T} = \{A \sqsubseteq B \sqcup C; B \sqsubseteq D; C \sqsubseteq D\}$. From $A(a)$ and the first axiom, $B(a)$ or $C(a)$ must be true. In both cases, the other axioms imply that $D(a)$ is true, hence $\mathcal{K} \models D(a)$. Note that $\mathcal{K}$ does not have a universal model.

To illustrate the specificities of Horn DLs, let us begin with the central DL $\mathcal{ALC}$. An $\mathcal{ALC}$ TBox contains only concept inclusions, where concepts have the following form:

$$C := \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid \exists r.C \mid \neg C \mid C_1 \sqcup C_2 \mid \forall r.C$$

with $A$ a concept name and $r$ a role name. Note that $C_1 \sqcup C_2$ and $\forall r.C$ can be equivalently written $\neg(\neg C_1 \sqcap \neg C_2)$ and $\neg\exists r.\neg C$, hence could be omitted. Similarly, $C_1 \sqcap C_2$ and $\exists r.C$ are redundant with $C_1 \sqcup C_2$ and $\forall r.C$. Finally, the same holds for $\top$ ($C \sqcup \neg C$) and $\bot$ ($C \sqcap \neg C$).

To define Horn-$\mathcal{ALC}$, the Horn subset of $\mathcal{ALC}$, by a syntactic restriction of $\mathcal{ALC}$ concept inclusions, we have to distinguish between both sides of inclusions. In particular, $\forall r.C \sqsubseteq B$ is excluded, while $B \sqsubseteq \forall r.C$ is allowed. Indeed, $\forall r.C \sqsubseteq B$ hides a negation in the left hand side of the inclusion, as it is equivalent to $\neg(\exists r.\neg C) \sqsubseteq B$, whereas $B \sqsubseteq \forall r.C$ is equivalent to $\exists r^-.B \sqsubseteq C$ (we use here the inverse role constructor, even if it is not part of $\mathcal{ALC}$ definition). Using the constructors of $\mathcal{ALC}$, Horn-$\mathcal{ALC}$ axioms are of the form $C_l \sqsubseteq C_r$ with:

$$C_l := \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid \exists r.C \mid C_1 \sqcup C_2$$

$$C_r := \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid \exists r.C \mid \forall r.C$$

More generally, the term *Horn DL* refers to description logics whose TBoxes can be semantically emulated by a set of Horn clauses. This set of Horn clauses is typically obtained by the standard logical translation of each TBox axiom, followed by a Skolemization step. As already pointed out (Section **??**), Skolemization preserves entailment of Boolean conjunctive queries.

Consider for instance the $\mathcal{ALC}$ axiom $\forall r.C \sqsubseteq B$: assuming that $r$, $C$ and $B$ are simply names, its clausal

**Table 1** Logical translation of concepts, roles and TBox axioms

| **Name** | **Syntax** | **Logical translation** |
|---|---|---|
| Concept name | $A$, $\top, \bot$ | $\phi_x(A) = A(x)$, $\phi_x(\top) = \top$, $\phi_x(\bot) = \bot$ |
| Role name | $r$ | $\phi_{x,y}(r) = r(x,y)$ |
| Concept/role conjunction | $C_1 \sqcap C_2$, $r_1 \sqcap r_2$ | $\phi_x(C_1 \sqcap C_2) = \phi_x(C_1) \wedge \phi_x(C_2)$, $\phi_{x,y}(r_1 \sqcap r_2) = \phi_{x,y}(r_1) \wedge \phi_{x,y}(r_2)$ |
| Concept/role negation | $\neg C, \neg r$ | $\phi_x(\neg C) = \neg\phi_x(C), \phi_{x,y}(\neg r) = \neg\phi_{x,y}(r)$ |
| Concept disjunction | $C_1 \sqcup C_2$ | $\phi_x(C_1 \sqcup C_2) = \phi_x(C_1) \vee \phi_x(C_2)$ |
| Existential restriction | $\exists r.C$ | $\phi_x(\exists r.C) = \exists y\ (\phi_{x,y}(r) \wedge \phi_y(C))$ |
| Universal restriction | $\forall r.C$ | $\phi_x(\forall r.C) = \forall y\ (\phi_{x,y}(r) \rightarrow \phi_y(C))$ |
| Inverse role | $r^-$ | $\phi_{x,y}(r^-) = \phi_{y,x}(r)$ |
| Number restriction | $\leq 1\ r.C$ | $\phi_x(\leq 1\ r.C) = \exists y_1 \exists y_2 (\phi_{x,y_1}(r) \wedge \phi_{x,y_2}(r) \wedge \phi_{y_1}(C) \wedge \phi_{y_2}(C)) \rightarrow y_1 = y_2$ |
| at most 1/ at least $m$ | $\geq m\ r.C$ | $\phi_x(\geq m\ r.C) = \exists y_1 \ldots y_m (\bigwedge_{i=1}^{m} (\phi_{x,y_i}(r) \wedge \phi_{y_i}(C)) \bigwedge_{j=1}^{i-1} y_i \neq y_j))$ |
| Concept inclusion | $C_1 \sqsubseteq C_2$ | $\forall x\ (\phi_x(C_1) \rightarrow \phi_x(C_2))$ |
| Role inclusion | $r_1 \sqsubseteq r_2$ | $\forall x \forall y\ (\phi_{x,y}(r_1) \rightarrow \phi_{x,y}(r_2))$ |
| Transitive role | $trans(r)$ | $\forall x \forall y \forall z\ (\phi_{x,y}(r) \wedge \phi_{y,z}(r) \rightarrow \phi_{x,z}(r))$ |

**Table 2** Normalized axioms in Horn DLs

| | |
|---|---|
| (1) | $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A$ |
| (2) | $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \bot$ |
| (3) | $\exists r.A_1 \sqsubseteq A_2$ |
| (4) | $\exists r.A_1 \sqsubseteq \bot$ |
| (5) | $A_1 \sqsubseteq \exists r.A_2$ |
| (6) | $A_1 \sqsubseteq \ \leq 1\ r.A_2$ |
| (7) | $A_1 \sqsubseteq \ \geq m\ r.A_2$ |
| (8) | $r_1 \sqsubseteq r_2$ |
| (9) | $r_1 \sqcap r_2 \sqsubseteq \bot$ (equivalent to $r_1 \sqsubseteq \neg r_2$) |
| (10) | $trans(r)$ |
| where $A_{(i)}$ denotes a concept name or $\top$, and $r_{(i)}$ denotes a role name or its inverse. | |

form is $\forall x \forall y\ ((r(x,y) \vee B(x)) \wedge (\neg C(x) \vee B(x)))$, where the first clause is not Horn; in contrast, $B \sqsubseteq \forall r.C$ yields the definite clause $\forall x \forall y\ (\neg B(y) \vee \neg r(y,x) \vee C(x))$. Fresh functional symbols are introduced to convert existential restrictions that (explicitly or implicitly) occur in the right hand side of inclusions; e.g., the standard translation of $A \sqsubseteq \exists r.B$ is $\forall x\ (A(x) \rightarrow \exists y\ r(x,y) \wedge B(y))$, which yields the clausal form $\forall x\ ((\neg A(x) \vee r(x, f_y(x))) \wedge (\neg A(x) \vee B(f_y(x))))$.

One could also see Horn DLs as the DLs for which all TBoxes can be semantically emulated in the existential rule framework, again based on their standard translation (Table **??**), although this is not the usual way of defining them. Such translation would avoid the Skolemization step.

From a semantic viewpoint, Horn DLs share with existential rules the desirable *universal model property*: every satisfiable KB has a universal model.

### 4.3 Horn DLs defined by normalized axioms

Defining the Horn restriction of a DL, based on the constructors allowed in this DL, can be cumbersome, since it requires analysing all possible interactions between constructors to fully eliminate disjunctive reasoning (see in particular [**?**,**?**] which define Horn restrictions of expressive DLs, in particular Horn-$\mathcal{SHIQ}$ that we will review later). A simpler approach consists in considering *normalized* axioms for a specific Horn DL, such that any TBox on this DL can be semantically emulated by a TBox that uses solely the normalized axioms.

Table **??** shows a set of normalized axioms, similar to those selected in [**?**], that define a Horn DL (actually a slight generalization of Horn-$\mathcal{SHIQ}$). These axioms cover all the Horn DLs we shall consider, up to TBox normalization. For instance, $\exists r.\exists s.A \sqsubseteq B_1 \sqcap B_2$ can be rewritten as three normalized axioms: $\exists s.A \sqsubseteq B_3$, $\exists r.B_3 \sqsubseteq B_1$ and $\exists r.B_3 \sqsubseteq B_2$, where $B_3$ is a fresh concept name. In particular, Horn-$\mathcal{ALC}$ can be defined by axioms (1)-(5); note these axioms use the inverse role constructor, which is not considered in the definition of $\mathcal{ALC}$.

Let us review the ten axioms of Table **??** from the angle of rules. Axioms (1), (3), (5), (8) and (10) yield existential rules; axioms (2), (4) and (9) yield negative constraints; axiom (6) yields an equality rule. The case of axiom (7) requires more attention; although it introduces inequalities, it can be rewritten into the following set of axioms, using $m$ fresh concept names $B_1 \ldots B_m$: axioms $A_1 \sqsubseteq \exists r.B_i$ (axiom (5)) and $B_i \sqsubseteq A_2$ (axiom (1)) for $1 \leq i \leq m$, as well as axioms $B_i \sqcap B_j \sqsubseteq \bot$ for $i < j \leq m$, stating that the $B_i$ concepts are pairwise disjoint (axiom (9)). Note that the $\neq$ relation could also be processed here as a standard relation, provided with rules that axiomatize its behavior, i.e., a rule stating that $\neq$ is symmetric and a negative constraint stating that it is irreflexive.

Let us now introduce some salient Horn DLs. Their precise relationships with decidable rule classes will be studied in the next section. Our focus will be on DLs for which query answering scales polynomially in the size of the data: the lightweight families DL-Lite and $\mathcal{EL}$, and the expressive Horn-$\mathcal{SHIQ}$, which generalizes Horn-$\mathcal{ALC}$.

The most well-known member of the family is DL-Lite$_R$, in which TBox axioms have the following shape:

$$B_1 \sqsubseteq B_2 \quad B_1 \sqsubseteq \neg B_2 \quad s_1 \sqsubseteq s_2 \quad s_1 \sqsubseteq \neg s_2$$

where $B_i := A \mid \exists s.\top$ and $s_i := r \mid r^-$, with $A$ and $r$ names.

In terms of Table **??**, DL-Lite$_R$ provides the following axioms: (1) with n = 1, (2) with n = 2, (3) and (5) with $A_2 = \top$, (8) and (9). For a systematic study of the DL-Lite family, see [**?**].

The $\mathcal{EL}$ family is named after its simplest member, the DL $\mathcal{EL}$, in which a TBox contains only concept inclusions, with concepts of the following shape:

$$C := \top \mid A \mid C_1 \sqcap C_2 \mid \exists r.C$$

where $A$ and $r$ are names.

In terms of Table **??**, $\mathcal{EL}$ provides axioms (1), (3) and (5) with $r$ restricted to a role name. Extensions of $\mathcal{EL}$ with inverse roles (indicated by letter $\mathcal{I}$) and role hierarchies (indicated by letter $\mathcal{H}$) yield the Horn DLs $\mathcal{ELI}$, $\mathcal{ELH}$ and $\mathcal{ELHI}$. Furthermore, the concept name $\bot$ (indicated by letter $\bot$) can be added at no computational cost. Note that Horn-$\mathcal{ALC}$ is included in $\mathcal{ELI}^{\bot}$.

The DL Horn-$\mathcal{SHI}$ extends Horn-$\mathcal{ALC}$ with transitivity, role hierarchies and role inverses. Horn-$\mathcal{SHIQ}$ is the extension of Horn-$\mathcal{SHI}$ with restricted forms of cardinalities, see axioms (6) and (7) in Table **??**. Moreover, in these cardinality axioms, $r$ has to be a so-called simple role, i.e., it is not transitive and does not have a transitive subrole. Up to this constraint, Horn-$\mathcal{SHIQ}$ provides all axioms from Table **??** except role disjointness (axiom (9)), which could be added at no cost.

## 5 Relationships between Horn Description Logics and Decidable Existential Rule Classes

We now study in more detail the relationships between Horn DLs and decidable rule classes. Note that none of the above Horn DLs yields a *fes* rule class and only one (DL-Lite$_R$) is *fus*, and it is also *gbts*. Hence, we will focus on the *gbts* family. We will ask ourselves the following questions: How do these Horn DLs fit into the gbts landscape? Does the shift from a Horn DL to a strictly more general gbts class necessarily imply an increase in the complexity of query answering? Can gbts rule classes be extended to handle features like transitivity, which are often offered by Horn DLs?

### 5.1 From Horn DLs to gbts

Let us consider again the Horn DL axioms from Table **??**. Concept inclusions are naturally translated into frontier-one rules (plus negative constraints). Furthermore, the obtained rules are body-acyclic. Similarly, role inclusions are naturally translated into linear rules (plus negative constraints). By contrast, the transitivity axiom is only captured by datalog. Indeed, let $p(x, y) \wedge p(y, z) \rightarrow p(x, z)$ be a transitivity rule: this rule is not frontier-guarded and, to be allowed in a weakly-(frontier)-guarded set of rules, it must satisfy the constraint that $x$ and $z$ are not both affected. Note that some other role axioms not mentioned in the table do naturally fit into gbts, like symmetry (a linear rule), irreflexivity and anti-symmetry (negative constraints).

Let us now review the specific Horn DLs introduced in the previous section. DL-Lite$_R$ is generalized by linear rules (plus negative constraints). The DLs $\mathcal{EL}$, Horn-$\mathcal{ALC}$, and $\mathcal{ELI}$, the extension of $\mathcal{EL}$ to inverse roles, are captured by body-acyclic frontier-one rules, and, if we consider their normalized axioms (as in Table **??**), by the subclass of guarded frontier-one rules. As soon as role hierarchies come in ($\mathcal{ELH}$ and $\mathcal{ELHI}$), the closest covering rule fragment is body-acyclic frontier-guarded rules, which we recall is equivalent to the guarded rule fragment. Note that the standard translation of these DLs directly produces existential rules in the target class. Even if we exclude constants from rules and consider only predicates with unary and binary arity, these inclusions are *strict*. For instance, the atom $p(x, x)$ cannot be expressed, neither can rules that create complex structures, like the rule $q(x) \rightarrow \exists y \exists z\ p(x, y) \wedge p(x, z) \wedge p(y, z)$, which belongs to the simplest gbts classes (note that the decomposition into atomic heads given in Section **??** would require here a ternary predicate). On the other hand, Horn-$\mathcal{SHI}$ cannot be translated into a gbts class, as it allows for unrestricted transitivity.

Table **??** synthetizes these relationships and shows the data and combined complexities for these Horn DLs and related existential rule classes. Specific references to these complexity results can be obtained from the work cited in Section **??** for the gbts family and from [**?**] for Horn DLs. Each DL is separated by a dotted line from the closest covering rule class. Concerning DLs, note that inverse roles in the $\mathcal{EL}$ family are responsible for a combined complexity increase, which is not the case for role hierarchies. Concerning frontier-guarded rules and

its subclasses, bounding the predicate arity does not always decrease the combined complexity: it does for linear, guarded and ba-fg (which are all body-acyclic classes), but not for fg, fr1 and the body-acyclic class gfr1 (while the question remains open for ba-fr1)[10].

Finally, we can see in the table that the most expressive DL covered by a rule class behaves like this class with respect to data and bounded-arity combined complexity, i.e., DL-Lite$_R$ compared to linear, $\mathcal{ELI}$ compared to g-fr1, and $\mathcal{ELHI}$ compared to guarded. Hence, these rule fragments can be considered as achieving better *expressivity-tractability* tradeoff than their DL counterpart. Nevertheless, beyond worse-case complexity, algorithmic aspects should also be taken into account. Indeed, these DLs have been provided with dedicated query answering techniques (see [**?**,**?**] for an introduction), which are often not transferable to more general rule fragments because they heavily rely on the specific shapes of their axioms. See in particular [**?**] about the impact of higher-arity predicates on algorithmic techniques, in the context of guarded existential rules.

**Table 3** Data and combined complexities for Horn DLs and related existential rule classes.

| **Class** | **Data** | **arity bounded** | **arity unbounded** |
|---|---|---|---|
| **linear** | $AC^0$ | NP-c | PSpace-c |
| **DL-Lite$_\mathcal{R}$** | $AC^0$ | NP-c | — |
| **fr1** | PTime-c | 2ExpTime-c | 2ExpTime-c |
| **ba-fr1** | PTime-c | ExpTime-c | in 2ExpTime |
| **g-fr1** | PTime-c | ExpTime-c | ExpTime-c |
| $\mathcal{ELI}$ | PTime-c | ExpTime-c | — |
| $\mathcal{EL}$ | PTime-c | NP-c | — |
| **fg** | PTime-c | 2ExpTime-c | 2ExpTime-c |
| **ba-fg, guarded** | PTime-c | ExpTime-c | 2ExpTime-c |
| $\mathcal{ELHI}$ | PTime-c | ExpTime-c | — |
| $\mathcal{ELH}$ | PTime-c | NP-c | — |
| **Horn-$\mathcal{SHI}(\mathcal{Q})$** | PTime-c | ExpTime-c | — |
| **fr1+trans** | PTime-c | 2ExpTime-c | 2ExpTime-c |

## 5.2 Pushing gbts further

Transitivity is an emblematic example of feature whose full support goes beyond the gbts fragment. In contrast, several expressive Horn DLs, like Horn-$\mathcal{SHI}(\mathcal{Q})$, include transitivity axioms, while controlling the interaction with cardinality constraints. This feature does often not increase the complexity of query answering (e.g., Horn-$\mathcal{SHI}(\mathcal{Q})$ with respect to $\mathcal{ELHI}$). However, the design of query answering algorithms becomes much more delicate because DL algorithms typically exploit the tree-like structure of the chase, which is destroyed by transitivity.

Fully understanding the effects of adding transitivity rules to decidable existential rule classes is still ongoing research (see [**?**] for a synthesis). Let us review results obtained on the gbts landscape. Given a rule class $\mathcal{C}$, we denote by $\mathcal{C} + trans$ the class obtained by adding the ability to express unrestricted transitivity rules. Atomic entailment over guarded+trans KBs is undecidable, even under very strong restrictions: when the guarded rules are restricted to the two-variable fragment of first-order logic, using only unary and binary predicates and only two transitive predicates [**?**]. In the wider fg+trans class, conjunctive query answering is decidable at the condition that transitivity and guardedness do not interact (i.e., each rule body has a frontier-guard that does not use a "directly or undirectly" transitive predicate) [**?**]. [11] The same article shows that fr1+trans is decidable without restrictions and not more difficult than fr1 (Table **??**). Surprisingly, it is currently not known if query answering with linear+trans is decidable without restrictions. On the positive side, linear+trans has been shown decidable for atomic entailment, as well as for conjunctive query answering when a (seemingly minor) safety condition is enforced [**?**,**?**]. This safety condition is always satisfied for unary and binary predicates but is not as soon as ternary predicates are allowed. Under this safety condition, query answering is in NL (a subclass of PTime) for data complexity and ExpTime-complete for combined complexity. Since linear rules are first-order rewritable, it may be interesting to notice that fus+trans is undecidable, even with the restriction to unary and binary predicates [**?**,**?**].

Similarly to transitivity, the ability of expressing the functionality of binary relations (as specific equality rules) is a feature that destroys the tree-structure of the chase if no restriction is enforced. Whereas it is offered by several DLs, how to integrate it in the gbts framework by suitable restrictions is again ongoing research.

## 6 Concluding remarks

Description logics and the existential rule framework both offer a variety of specific fragments, providing different expressivity-tractability tradeoffs. While classical

[10] Surprisingly, ba-fr1 is not in a lower complexity class than ba-fg for data and bounded-arity combined complexity.

[11] Actually, this result is established for a larger first-order logic fragment, itself included in the so-called guarded negation fragment.

DLs based on $\mathcal{ALC}$ and existential rules are quite "orthogonal" formalisms, the more recent Horn DLs can be naturally translated into the existential rule framework. More specifically, lightweight Horn DLs are covered by decidable rule classes from the greedy bounded-treewidth family. These rule classes allow for higher predicate arity and complex structures in rule heads, at no additional computational cost compared to the closest DLs when predicate arity is bounded, at least with respect to worst-case complexity. On the other hand, thanks to a careful control of the interactions between constructors, expressive Horn DLs can accomodate both transitivity and functionality, which are often desirable features in data modeling. These latter DLs do not have natural covering classes in the existential rule landscape. In both research fields, the question of how to extend well-behaved fragments with practically useful features without sacrifying computational efficiency is actively studied, which should provide new insight in the near future.

Finally, let us put the approach we adopted here in a wider context. We relied on semantic emulation, using the standard translation of description logic axioms into first-order logic, which quite directly yields rules from the existential rule framework. This led to conceptually simple translations, moreover computable in polynomial time, which made the comparison in terms of combined complexity relevant. A more general way of comparing the expressivity of two ontological languages is to require only that answers to queries are preserved, without enforcing a correspondence between the models of the ontologies. This latter approach is notably followed by a large body of research that investigates the relative expressivity of ontological languages coupled with database query languages. In this context, the objects of study are so-called ontology-mediated queries (OMQs), which are pairs $(\mathcal{T}, q)$, where $\mathcal{T}$ is a TBox in some DL (more generally, a theory in some logical fragment) and $q$ a conjunctive query (more generally, a database query). Then, an OMQ language $\mathcal{Q}_1$ is deemed at least as expressive as another OMQ language $\mathcal{Q}_2$ if every OMQ in $\mathcal{Q}_2$ can be expressed as an OMQ in $\mathcal{Q}_1$, in such a way that both queries yields the same answers on all databases. Prominent work in this area includes in particular [**?**] for DLs and [**?**] for guarded existential rules and related classes.

**Acknowledgements** Many thanks to the anonymous reviewers for their helpful comments and suggestions, and to Jean-François Baget for proofreading an earlier version of this paper. This work was partially funded by the ANR project CQFD (ANR-18-CE23-0003).