Derivation-Graph-Based Characterizations of Decidable Existential Rule Sets[⋆]

Tim S. Lyon(✉) and Sebastian Rudolph

Computational Logic Group, TU Dresden, Dresden, Germany
{timothy_stephen.lyon,sebastian.rudolph}@tu-dresden.de

Abstract. This paper establishes alternative characterizations of very expressive classes of existential rule sets with decidable query entailment. We consider the notable class of greedy bounded-treewidth sets (**gbts**) and a new, generalized variant, called weakly gbts (**wgbts**). Revisiting and building on the notion of derivation graphs, we define (weakly) cycle-free derivation graph sets ((**w**)**cdgs**) and employ elaborate proof-theoretic arguments to obtain that **gbts** and **cdgs** coincide, as do **wgbts** and **wcdgs**. These novel characterizations advance our analytic proof-theoretic understanding of existential rules and will likely be instrumental in practice.

Keywords: TGDs · query entailment · bounded treewidth · proof-theory

1 Introduction

The formalism of existential rules has come to prominence as an effective approach for both specifying and querying knowledge. Within this context, a knowledge base takes the form $\mathcal{K} = (\mathcal{D}, \mathcal{R})$, where $\mathcal{D}$ is a finite collection of atomic facts (called a *database*) and $\mathcal{R}$ is a finite set of *existential rules* (called a *rule set*), which are first-order formulae of the form $\forall \mathbf{x}\mathbf{y}(\varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\psi(\mathbf{y}, \mathbf{z}))$. Although existential rules are written in a relatively simple language, they are expressive enough to generalize many important languages used in knowledge representation, including rule-based formalisms as well as such based on description logics. Moreover, existential rules have meaningful applications within the domain of ontology-based query answering [2], data exchange and integration [9], and have proven beneficial in the study of general decidability criteria [10].

The *Boolean conjunctive query entailment problem* consists of taking a knowledge base $\mathcal{K}$, a Boolean conjunctive query (BCQ) q, and determining if $\mathcal{K} \models q$. As this problem is known to be undecidable for arbitrary rule sets [7], much work has gone into identifying existential rule fragments for which decidability can be reclaimed. Typically, such classes of rule sets are described in one of two ways: either, a rule set's membership in said class can be established through easily verifiable *syntactic properties* (such classes are called *concrete classes*), or the property is more *abstract* (which is often defined on the basis of semantic

⋆ Work supported by the ERC through Consolidator Grant 771779 (DeciGUT).

notions) and may be hard or even impossible to algorithmically determine (such classes are called *abstract classes*). Examples of concrete classes include functional/inclusion dependencies [11], datalog, and guarded rules [6]. Examples of abstract classes include finite expansion sets [4], finite unification sets [3], and bounded-treewidth sets (**bts**) [6].

Yet, there is another means of establishing the decidability of query entailment: only limited work has gone into identifying classes of rule sets with decidable query entailment based on their *proof-theoretic characteristics*, in particular, based on specifics of the derivations such rules produce. To the best of our knowledge, only the class of *greedy bounded treewidth sets* (**gbts**) has been identified in such a manner (see [14]). A rule set qualifies as **gbts** when every derivation it produces is *greedy*, in a sense that it is possible to construct a tree decomposition of finite width in a "greedy" fashion alongside the derivation, ensuring the existence of a model with finite treewidth for the knowledge base under consideration, thus warranting the decidability of query entailment [6].

In this paper, we investigate the **gbts** class and three new classes of rule sets where decidability is determined proof-theoretically. First, we define a weakened version of **gbts**, dubbed **wgbts**, where the rule set need only produce *at least one greedy derivation* relative to any given database. Second, we investigate two new classes of rule sets, dubbed *cycle-free derivation graph sets* (**cdgs**) and *weakly cycle-free derivation graph sets* (**wcdgs**), which are defined relative to the notion of a *derivation graph*. Derivation graphs were introduced by Baget et al. [5] and are directed acyclic graphs encoding *how* certain facts are derived in the course of a derivation. Notably, via the application of *reduction operations*, a derivation graph may be reduced to a tree, which serves as a tree decomposition of a model of the considered knowledge base. Such objects helped establish that (weakly) frontier-guarded rule sets are **bts** [5]. In short, our key contributions are (Fig. 1):

1. We investigate how proof-theoretic structures gives rise to decidable query entailment and propose three new classes of rule sets.
2. We show that **gbts** = **cdgs** and **wgbts** = **wcdgs**, establishing a correspondence between greedy derivations and reducible derivation graphs.
3. We show that **wgbts** *properly subsumes* **gbts** via a novel proof transformation argument. Therefore, by the former point, we also find that **wcdgs** properly subsumes **cdgs**.

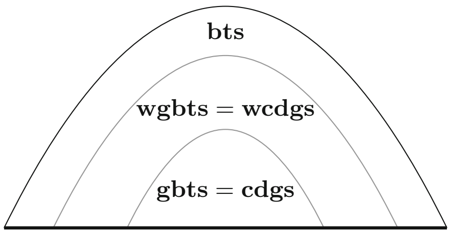

Fig. 1. A graphic depicting the containment relations between the classes of rule sets considered. The solid edges represent strict containment relations.

The paper is organized accordingly: In Section 2, we define preliminary notions. We study **gbts** and **wgbts** in Section 3, and show that the latter class properly subsumes the former via an intricate proof transformation argument. In Section 4, we define **cdgs** and **wcdgs** as well as show that **gbts** = **cdgs** and **wgbts** = **wcdgs**. In Section 5, we conclude and discuss future work. Last, we note that full proofs can be found in the appended version [12].

2 Preliminaries

Syntax and formulae. We let **Ter** be a set of *terms*, which is the the union of three countably infinite, pairwise disjoint sets, namely, the set of *constants* **Con**, the set of *variables* **Var**, and the set of *nulls* **Nul**. We use a, b, c, ... (occasionally annotated) to denote constants, and x, y, z, ... (occasionally annotated) to denote both variables and nulls. A *signature* Σ is a set of *predicates* p, q, r, ... (which may be annotated) such that for each $p \in \Sigma$, $ar(p) \in \mathbb{N}$ is the *arity* of p. For simplicity, we assume a fixed signature Σ throughout the paper.

An *atom* over Σ is defined to be a formula of the form $p(t_1, \dots, t_n)$, where $p \in \Sigma$, $ar(p) = n$, and $t_i \in \mathbf{Ter}$ for each $i \in \{1, \dots, n\}$. A *ground atom* over Σ is an atom $p(a_1, \dots, a_n)$ such that $a_i \in \mathbf{Con}$ for each $i \in \{1, \dots, n\}$. We will often use $\mathbf{t}$ to denote a tuple $(t_1, \dots, t_n)$ of terms and $p(\mathbf{t})$ to denote a (ground) atom $p(t_1, \dots, t_n)$. An *instance* over Σ is defined to be a (potentially infinite) set $\mathcal{I}$ of atoms over constants and nulls, and a *database* $\mathcal{D}$ is a finite set of ground atoms. We let $\mathcal{X}$, $\mathcal{Y}$, ... (occasionally annotated) denote (potentially infinite) sets of atoms with $\mathbf{Ter}(\mathcal{X})$, $\mathbf{Con}(\mathcal{X})$, $\mathbf{Var}(\mathcal{X})$, and $\mathbf{Nul}(\mathcal{X})$ denoting the set of terms, constants, variables, and nulls occurring in the atoms of $\mathcal{X}$, respectively.

Substitutions and homomorphisms. A *substitution* is a partial function over the set of terms **Ter**. A *homomorphism* h from a set $\mathcal{X}$ of atoms to a set $\mathcal{Y}$ of atoms, is a substitution $h : \mathbf{Ter}(\mathcal{X}) \to \mathbf{Ter}(\mathcal{Y})$ such that (i) $p(h(t_1), \dots, h(t_n)) \in \mathcal{Y}$, if $p(t_1, \dots, t_n) \in \mathcal{X}$, and (ii) $h(a) = a$ for each $a \in \mathbf{Con}$. If h is a homomorphism from $\mathcal{X}$ to $\mathcal{Y}$, we say that h *homomorphically maps* $\mathcal{X}$ to $\mathcal{Y}$. Atom sets $\mathcal{X}, \mathcal{Y}$ are *homomorphically equivalent*, written $\mathcal{X} \equiv \mathcal{Y}$, *iff* $\mathcal{X}$ homomorphically maps to $\mathcal{Y}$, and vice versa. An *isomorphism* is a bijective homomorphism h where h^{-1} is a homomorphism.

Existential rules. Whereas databases encode assertional knowledge, ontologies consist in the current setting of *existential rules*, which we will frequently refer to as *rules* more simply. An existential rule is a first-order sentence of the form:

$$\rho = \forall \mathbf{x}\mathbf{y}(\varphi(\mathbf{x}, \mathbf{y}) \to \exists \mathbf{z} \psi(\mathbf{y}, \mathbf{z}))$$

where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are pairwise disjoint collections of variables, $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over constants and the variables $\mathbf{x}, \mathbf{y}$, and $\psi(\mathbf{y}, \mathbf{z})$ is a conjunction of atoms over constants and the variables $\mathbf{y}, \mathbf{z}$. We define $body(\rho) = \varphi(\mathbf{x}, \mathbf{y})$ to be the *body* of ρ, and $head(\rho) = \psi(\mathbf{y}, \mathbf{z})$ to be the *head* of ρ. For convenience, we will often interpret a conjunction $p_1(\mathbf{t}_1) \wedge \dots \wedge p_n(\mathbf{t}_n)$ of atoms (such as the body or head of a rule) as a set $\{p_1(\mathbf{t}_1), \cdots, p_n(\mathbf{t}_n)\}$ of atoms; if h is a homomorphism, then $h(p_1(\mathbf{t}_1) \wedge \cdots \wedge p_n(\mathbf{t}_n)) := \{p_1(h(\mathbf{t}_1)), \cdots, p_n(h(\mathbf{t}_n))\}$ with h applied componentwise to each tuple $\mathbf{t}_i$ of terms. The *frontier* of ρ, written $fr(\rho)$, is the set of variables $\mathbf{y}$ that the body and head of ρ have in common, that is, $fr(\rho) = \mathbf{Var}(body(\rho)) \cap \mathbf{Var}(head(\rho))$. We define a *frontier atom* in a rule ρ to be an atom containing at least one frontier variable. We use ρ and annotated versions thereof to denote rules, as well as $\mathcal{R}$ and annotated versions thereof to denote finite sets of rules (simply called *rule sets*).

Models. We note that sets of atoms (which include instances and databases) may be seen as first-order interpretations, and so, we may use $\models$ to represent the satisfaction of formulae on such structures. A set of atoms $\mathcal{X}$ satisfies a set of atoms $\mathcal{Y}$ (or, equivalently, $\mathcal{X}$ is a model of $\mathcal{Y}$), written $\mathcal{X} \models \mathcal{Y}$, *iff* there exists a homomorphic mapping from $\mathcal{Y}$ to $\mathcal{X}$. A set of atoms $\mathcal{X}$ satisfies a rule ρ (or, equivalently, $\mathcal{X}$ is a model of ρ), written $\mathcal{X} \models \rho$, *iff* for any homomorphism h, if h is a homomorphism from $body(\rho)$ to $\mathcal{X}$, then it can be extended to a homomorphism $\overline{h}$ that also maps $head(\rho)$ to $\mathcal{X}$. A set of atoms $\mathcal{X}$ satisfies a rule set $\mathcal{R}$ (or, equivalently, $\mathcal{X}$ is a model of $\mathcal{R}$), written $\mathcal{X} \models \mathcal{R}$, *iff* $\mathcal{X} \models \rho$ for every rule $\rho \in \mathcal{R}$. If a model $\mathcal{X}$ of a set of atoms, a rule, or a rule set homomorphically maps into *every* model of that very set of atoms, rule, or rule set, then we refer to $\mathcal{X}$ as a *universal model* of that set of atoms, rule, or rule set [8].

Knowledge bases and querying. A *knowledge base (KB)* $\mathcal{K}$ is defined to be a pair $(\mathcal{D}, \mathcal{R})$, where $\mathcal{D}$ is a database and $\mathcal{R}$ is a rule set. An instance $\mathcal{I}$ is a *model* of $\mathcal{K} = (\mathcal{D}, \mathcal{R})$ *iff* $\mathcal{D} \subseteq \mathcal{I}$ and $\mathcal{I} \models \mathcal{R}$. We consider querying knowledge bases with *conjunctive queries (CQs)*, that is, with formulae of the form $q(\mathbf{y}) = \exists \mathbf{x} \varphi(\mathbf{x}, \mathbf{y})$, where $\varphi(\mathbf{x}, \mathbf{y})$ is a non-empty conjunction of atoms over the variables $\mathbf{x}, \mathbf{y}$ and constants. We refer to the variables $\mathbf{y}$ in $q(\mathbf{y})$ as *free* and define a *Boolean conjunctive query (BCQ)* to be a CQ without free variables, i.e. a BCQ is a CQ of the form $q = \exists \mathbf{x} \varphi(\mathbf{x})$. A knowledge base $\mathcal{K} = (\mathcal{D}, \mathcal{R})$ *entails* a CQ $q(\mathbf{y}) = \exists \mathbf{x} \varphi(\mathbf{x}, \mathbf{y})$, written $\mathcal{K} \models q(\mathbf{y})$, *iff* $\varphi(\mathbf{x}, \mathbf{y})$ homomorphically maps into every model $\mathcal{I}$ of $\mathcal{K}$; we note that this is equivalent to $\varphi(\mathbf{x}, \mathbf{y})$ homomorphically mapping into a universal model of $\mathcal{D}$ and $\mathcal{R}$.

As we are interested in extracting implicit knowledge from the explicit knowledge presented in a knowledge base $\mathcal{K} = (\mathcal{D}, \mathcal{R})$, we are interested in deciding the *BCQ entailment problem*:[1]

(BCQ Entailment) Given a KB $\mathcal{K}$ and a BCQ q, is it the case that $\mathcal{K} \models q$?

While it is well-known that the BCQ entailment problem is undecidable in general [7], restricting oneself to certain classes of rule sets (e.g. datalog or finite unification sets [5]) may recover decidability. We refer to classes of rule sets for which BCQ entailment is decidable as *query-decidable classes.*

Derivations. One means by which we can extract implicit knowledge from a given KB is through the use of *derivations*, that is, sequences of instances

[1] We recall that entailment of non-Boolean CQs or even query answering can all be reduced to BCQ entailment in logarithmic space.

obtained by sequentially applying rules to given data. We say that a rule $\rho = \forall \mathbf{xy}(\varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \psi(\mathbf{y}, \mathbf{z}))$ is *triggered* in an instance $\mathcal{I}$ via a homomorphism h, written succinctly as $\tau(\rho, \mathcal{I}, h)$, *iff* h homomorphically maps $\varphi(\mathbf{x}, \mathbf{y})$ to $\mathcal{I}$. In this case, we define

$$\mathbf{Ch}(\mathcal{I}, \rho, h) = \mathcal{I} \cup \overline{h}(\psi(\mathbf{y}, \mathbf{z})),$$

where $\overline{h}$ is an extension of h mapping every variable z in $\mathbf{z}$ to a fresh null. Consequently, we define an $\mathcal{R}$*-derivation* to be a sequence $\mathcal{I}_0, (\rho_1, h_1, \mathcal{I}_1), \ldots, (\rho_n, h_n, \mathcal{I}_n)$ such that (i) $\rho_i \in \mathcal{R}$ for each $i \in \{1, \ldots, n\}$, (ii) $\tau(\rho_i, \mathcal{I}_{i-1}, h_i)$ holds for $i \in \{1, \ldots, n\}$, and (iii) $\mathcal{I}_i = \mathbf{Ch}(\mathcal{I}_{i-1}, \rho, h_i)$ for $i \in \{1, \ldots, n\}$. We will use δ and annotations thereof to denote $\mathcal{R}$-derivations, and we define the length of an $\mathcal{R}$-derivation $\delta = \mathcal{I}_0, (\rho_1, h_1, \mathcal{I}_1), \ldots, (\rho_n, h_n, \mathcal{I}_n)$, denoted $|\delta|$, to be n. Furthermore, for instances $\mathcal{I}$ and $\mathcal{I}'$, we write $\mathcal{I} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}'$ to mean that there exists an $\mathcal{R}$-derivation δ of $\mathcal{I}'$ from $\mathcal{I}$. Also, if $\mathcal{I}''$ can be derived from $\mathcal{I}'$ by means of a rule $\rho \in \mathcal{R}$ and homomorphism h, we abuse notation and write $\mathcal{I} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}', (\rho, h, \mathcal{I}'')$ to indicate that $\mathcal{I} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}'$ and $\mathcal{I}' \xrightarrow{\delta'}_{\mathcal{R}} \mathcal{I}''$ with $\delta' = \mathcal{I}', (\rho, h, \mathcal{I}'')$. Derivations play a fundamental role in this paper as we aim to identify (and analyze the relationships between) query-decidable classes of rule sets based on *how* such rule sets derive information, i.e. we are interested in classes of rule sets that may be *proof-theoretically characterized.*

Chase. A tool that will prove useful in the current work is the *chase*, which in our setting is a procedure that (in essence) simultaneously constructs all $\mathcal{K}$-derivations in a breadth-first manner. Although many variants of the chase exist [5,9,13], we utilize the chase procedure (also called the *k-Saturation*) from Baget et al. [5]. We use the chase in the current work as a purely technical tool for obtaining universal models of knowledge bases, proving useful in separating certain query-decidable classes of rule sets.

We define the *one-step application* of all triggered rules from some $\mathcal{R}$ in $\mathcal{I}$ by

$$\mathbf{Ch}_1(\mathcal{I}, \mathcal{R}) = \bigcup\nolimits_{\rho \in \mathcal{R}, \tau(\rho, \mathcal{I}, h)} \mathbf{Ch}(\mathcal{I}, \rho, h),$$

assuming all nulls introduced in the "parallel" applications of $\mathbf{Ch}$ to $\mathcal{I}$ are distinct. We let $\mathbf{Ch}_0(\mathcal{I}, \mathcal{R}) = \mathcal{I}$, as well as let $\mathbf{Ch}_{i+1}(\mathcal{I}, \mathcal{R}) = \mathbf{Ch}_1(\mathbf{Ch}_i(\mathcal{I}, \mathcal{R}), \mathcal{R})$, and define the *chase* to be

$$\mathbf{Ch}_\infty(\mathcal{I}, \mathcal{R}) = \bigcup\nolimits_{i \in \mathbb{N}} \mathbf{Ch}_i(\mathcal{I}, \mathcal{R}).$$

For any KB $\mathcal{K} = (\mathcal{D}, \mathcal{R})$, the chase $\mathbf{Ch}_\infty(\mathcal{D}, \mathcal{R})$ is a universal model of $\mathcal{K}$, that is, $\mathcal{D} \subseteq \mathbf{Ch}_\infty(\mathcal{D}, \mathcal{R})$, $\mathbf{Ch}_\infty(\mathcal{D}, \mathcal{R}) \models \mathcal{R}$, and $\mathbf{Ch}_\infty(\mathcal{D}, \mathcal{R})$ homomorphically maps into every model of $\mathcal{D}$ and $\mathcal{R}$.

Rule dependence. Let ρ and ρ' be rules. We say that ρ' *depends* on ρ *iff* there exists an instance $\mathcal{I}$ such that (i) ρ' is not triggered in $\mathcal{I}$ via any homomorphism, (ii) ρ is triggered in $\mathcal{I}$ via a homomorphism h, and (iii) ρ' is triggered in $\mathbf{Ch}(\mathcal{I}, \rho, h)$ via a homomorphism h'. We define the *graph of rule dependencies* [1] of a set $\mathcal{R}$ of rules to be $G(\mathcal{R}) = (V, E)$ such that (i) $V = \mathcal{R}$ and (ii) $(\rho, \rho') \in E$ *iff* ρ' depends on ρ.

Treewidth. A *tree decomposition* of an instance $\mathcal{I}$ is defined to be a tree $T = (V, E)$ such that $V \subseteq 2^{\mathbf{Ter}(\mathcal{I})}$ (where each element of V is called a *bag*) and $E \subseteq V \times V$, satisfying the following three conditions: (i) $\bigcup_{X \in V} X = \mathbf{Ter}(\mathcal{I})$, (ii) for each $p(t_1, \ldots, t_n) \in \mathcal{I}$, there is an $X \in V$ such that $\{t_1, \ldots, t_n\} \subseteq X$, and (iii) for each $t \in \mathbf{Ter}(\mathcal{I})$, the subgraph of T induced by the bags $X \in V$ with $t \in X$ is connected (this condition is referred to as the *connectedness condition*). We define the *width* of a tree decomposition $T = (V, E)$ of an instance $\mathcal{I}$ as follows:

$$w(T) := \max\{|X| : X \in V\} - 1$$

i.e. the width is equal to the cardinality of the largest node in T minus 1. We let $w(T) = \infty$ *iff* for all $n \in \mathbb{N}$, $n \leq \max\{|X| : X \in V\}$. We define the *treewidth* of an instance $\mathcal{I}$, written $tw(\mathcal{I})$, as follows:

$$tw(\mathcal{I}) := \min\{w(T) : T \text{ is a tree decomposition of } \mathcal{I}\}$$

i.e. the treewidth of an instance equals the minimal width among all its tree decompositions. If no tree decomposition of $\mathcal{I}$ has finite width, we set $tw(\mathcal{I}) = \infty$.

3 Greediness

We now discuss a property of derivations referred to as *greediness*. In essence, a derivation is greedy when the image of the frontier of any applied rule consists solely of constants from a given KB and/or nulls introduced by a *single* previous rule application. Such derivations were defined by Thomazo et al. [14] and were used to identify the (query-decidable) class of *greedy bounded-treewidth sets* (**gbts**), that is, the class of rule sets that produce only *greedy derivations* (defined below) when applied to a database.

In this section, we also identify a new query-decidable class of rule sets, referred to as *weakly greedy bounded-treewidth sets* (**wgbts**). The **wgbts** class serves as a more liberal version of **gbts**, and contains rule sets that admit at least one greedy derivation of any derivable instance. It is straightforward to confirm that **wgbts** generalizes **gbts** since if a rule set is **gbts**, then every derivation of a derivable instance is greedy, implying that every derivable instance has *some* greedy derivation. Yet, what is non-trivial to show is that **wgbts** *properly subsumes* **gbts**. We are going to prove this fact by means of a proof-theoretic argument and counter-example along the following lines: first, we show under what conditions we can permute rule applications in a given derivation (see Lemma 1 below), and second, we provide a rule set which exhibits non-greedy derivations (witnessing that the rule set is not **gbts**), but for which every derivation can be transformed into a greedy derivation by means of rule permutations and replacements (witnessing **wgbts** membership).

Let us now formally define greedy derivations and provide examples to demonstrate the concept of (non-)greediness. Based on this, we then proceed to define the **gbts** and **wgbts** classes.

Definition 1 (Greedy Derivation [14]). *We define an $\mathcal{R}$-derivation*

$$\delta = \mathcal{I}_0, (\rho_1, h_1, \mathcal{I}_1), \ldots, (\rho_n, h_n, \mathcal{I}_n)$$

to be greedy *iff for each i such that $0 < i \leq n$, there exists a $j < i$ such that $h_i(fr(\rho_i)) \subseteq \mathbf{Nul}(\overline{h}_j(head(\rho_j))) \cup \mathbf{Con}(\mathcal{I}_0, \mathcal{R}) \cup \mathbf{Nul}(\mathcal{I}_0)$.*

To give examples of non-greedy and greedy derivations, let us define the database $\mathcal{D}_\dagger := \{p(a), r(b)\}$ and the rule set $\mathcal{R}_2 := \{\rho_1, \rho_2, \rho_3, \rho_4\}$, with

$$\begin{aligned} \rho_1 &= p(x) \to \exists yz.q(x,y,z) & \rho_3 &= p(x) \wedge r(y) \to \exists zwuv.q(x,z,w) \wedge s(y,u,v) \\ \rho_2 &= r(x) \to \exists yz.s(x,y,z) & \rho_4 &= q(x,y,z) \wedge s(w,u,v) \to \exists o.t(x,y,w,u,o) \end{aligned}$$

An example of a non-greedy derivation is the following:

$$\delta_1 = \mathcal{D}_\dagger, (\rho_1, h_1, \mathcal{I}_1), (\rho_1, h_2, \mathcal{I}_2), (\rho_2, h_3, \mathcal{I}_3), (\rho_4, h_4, \mathcal{I}_4), \quad \text{with}$$

$$\mathcal{I}_4 = \{\underbrace{p(a), r(b)}_{\mathcal{D}_\dagger}, \underbrace{q(a, y_0, z_0)}_{\mathcal{I}_1 \setminus \mathcal{D}_\dagger}, \underbrace{q(a, y_1, z_1)}_{\mathcal{I}_2 \setminus \mathcal{I}_1}, \underbrace{s(b, y_2, z_2)}_{\mathcal{I}_3 \setminus \mathcal{I}_2}, \underbrace{t(a, y_0, b, y_2, o)}_{\mathcal{I}_4 \setminus \mathcal{I}_3}\} \quad \text{and}$$

$h_1 = h_2 = \{x \mapsto a\}$, $h_3 = \{x \mapsto b\}$, $h_4 = \{x \mapsto a, y \mapsto y_0, z \mapsto z_0, w \mapsto b, u \mapsto y_2, v \mapsto z_2\}$. Note that this derivation is not greedy because

$$h_4(fr(\rho_4)) = h_4(\{x, y, w, u\}) = \{a, \overbrace{y_0}^{\in \mathbf{Nul}(\overline{h}_1(head(\rho_1)))}, b, \underbrace{y_2}_{\in \mathbf{Nul}(\overline{h}_3(head(\rho_2)))}\}$$

That is to say, the image of the frontier from the last rule application (i.e. the application of ρ_4) contains nulls introduced by *two* previous rule applications (as opposed to containing nulls from just a single previous rule application), namely, the first application of ρ_1 and the application of ρ_2. In contrast, the following is an example of a greedy derivation

$$\delta_2 = \mathcal{D}_\dagger, (\rho_3, h'_1, \mathcal{I}'_1), (\rho_1, h'_2, \mathcal{I}'_2), (\rho_4, h'_3, \mathcal{I}'_3), \text{ with}$$

$$\mathcal{I}'_3 = \{\underbrace{p(a), r(b)}_{\mathcal{D}_\dagger}, \underbrace{q(a, y_0, z_0), s(b, y_2, z_2)}_{\mathcal{I}'_1 \setminus \mathcal{D}_\dagger}, \underbrace{q(a, y_1, z_1)}_{\mathcal{I}'_2 \setminus \mathcal{I}'_1}, \underbrace{t(a, y_0, b, y_2, o)}_{\mathcal{I}'_3 \setminus \mathcal{I}'_2}\} \quad \text{and}$$

$h'_1 = \{x \mapsto a, y \mapsto b\}$, $h'_2 = \{x \mapsto a\}$, $h'_3 = \{x \mapsto a, y \mapsto y_0, z \mapsto z_0, w \mapsto b, u \mapsto y_2, v \mapsto z_2\}$.

Greediness of δ_2 follows from the frontier of any applied rule being mapped to nothing but constants and/or nulls introduced by a sole previous rule application.

Definition 2 ((Weakly) Greedy Bounded-Treewidth Set). *A rule set $\mathcal{R}$ is a* greedy bounded-treewidth set (**gbts**) *iff if $\mathcal{D} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}$, then δ is greedy. $\mathcal{R}$ is a* weakly greedy bounded-treewidth set (**wgbts**) *iff if $\mathcal{D} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}$, then there exists some greedy $\mathcal{R}$-derivation δ' such that $\mathcal{D} \xrightarrow{\delta'}_{\mathcal{R}} \mathcal{I}$.*

Remark 1. Observe that **gbts** and **wgbts** are characterized on the basis of derivations starting from given *databases* only, that is, derivations of the form $\mathcal{I}_0, (\rho_1, h_1, \mathcal{I}_1), \dots, (\rho_n, h_n, \mathcal{I}_n)$ where $\mathcal{I}_0 = \mathcal{D}$ is a database. In such a case, a derivation of the above form is greedy *iff* for each i with $0 < i \leq n$, there exists a $j < i$ such that $h_i(fr(\rho_i)) \subseteq \mathbf{Nul}(\overline{h}_j(head(\rho_j))) \cup \mathbf{Con}(\mathcal{D}, \mathcal{R})$ as databases only contain constants (and not nulls) by definition.

As noted above, it is straightforward to show that **wgbts** subsumes **gbts**. Still, establishing that **wgbts** strictly subsumes **gbts**, i.e. there are rule sets within **wgbts** that are outside **gbts**, requires more effort. As it so happens, the rule set $\mathcal{R}_2$ (defined above) serves as such a rule set, admitting non-greedy $\mathcal{R}_2$-derivations, but where it can be shown that every instance derivable using the rule set admits a greedy $\mathcal{R}_2$-derivation. As a case in point, observe that the $\mathcal{R}_2$-derivations δ_1 and δ_2 both derive the same instance $\mathcal{I}_4 = \mathcal{I}'_3$, however, δ_1 is a non-greedy $\mathcal{R}_2$-derivation of the instance and δ_2 is a greedy $\mathcal{R}_2$-derivation of the instance. Clearly, the existence of the non-greedy $\mathcal{R}_2$-derivation δ_1 witnesses that $\mathcal{R}_2$ is not **gbts**. To establish that $\mathcal{R}_2$ still falls within the **wgbts** class, we show that every non-greedy $\mathcal{R}_2$-derivation can be transformed into a greedy $\mathcal{R}_2$-derivation using two operations: (i) rule permutations and (ii) rule replacements.

Regarding rule permutations, we consider under what conditions we may swap consecutive applications of rules in a derivation to yield a new derivation of the same instance. For example, in the $\mathcal{R}_2$-derivation δ_1 above, we may swap the consecutive applications of ρ_1 and ρ_2 to obtain the following derivation:

$$\delta'_1 = \mathcal{D}_\dagger, (\rho_1, h_1, \mathcal{I}_1), (\rho_2, h_3, \mathcal{I}_1 \cup (\mathcal{I}_3 \setminus \mathcal{I}_2)), (\rho_1, h_2, \mathcal{I}_3), (\rho_4, h_4, \mathcal{I}_4).$$

$\mathcal{I}_1 \cup (\mathcal{I}_3 \setminus \mathcal{I}_2) = \{p(a), r(b), q(a, y_0, z_0), s(b, y_2, z_2)\}$ is derived by applying ρ_2 and the subsequent application of ρ_1 reclaims the instance $\mathcal{I}_3$. Therefore, the same instance $\mathcal{I}_4$ remains the conclusion. Although one can confirm that δ'_1 is indeed an $\mathcal{R}_2$-derivation, thus serving as a successful example of a rule permutation (meaning, the rule permutation yields another $\mathcal{R}_2$-derivation), the following question still remains: for a rule set $\mathcal{R}$, under what conditions will permuting rules within a given $\mathcal{R}$-derivation always yield another $\mathcal{R}$-derivation?

We pose an answer to this question, formulated as the *permutation lemma* below, which states that an application of a rule ρ may be permuted before an application of a rule ρ' so long as the former rule does not depend on the latter (in the sense formally defined in Section 2 based on the work of Baget [1]). Furthermore, it should be noted that such rule permutations preserve the greediness of derivations. In the context of the above example, ρ_2 may be permuted before ρ_1 in δ_1 because the former does not depend on the latter.

Lemma 1 (Permutation Lemma). *Let $\mathcal{R}$ be a rule set with $\mathcal{I}_0$ an instance. Suppose we have a (greedy) $\mathcal{R}$-derivation of the following form:*

$$\mathcal{I}_0, \dots, (\rho_i, h_i, \mathcal{I}_i), (\rho_{i+1}, h_{i+1}, \mathcal{I}_{i+1}), \dots, (\rho_n, h_n, \mathcal{I}_n)$$

If ρ_{i+1} does not depend on ρ_i, then the following is a (greedy) $\mathcal{R}$-derivation too:

$$\mathcal{I}_0, \dots, (\rho_{i+1}, h_{i+1}, \mathcal{I}_{i-1} \cup (\mathcal{I}_{i+1} \setminus \mathcal{I}_i)), (\rho_i, h_i, \mathcal{I}_{i+1}), \dots, (\rho_n, h_n, \mathcal{I}_n).$$

As a consequence of the above lemma, rules may always be permuted in a given $\mathcal{R}$-derivation so that its structure mirrors the graph of rule dependencies $G(\mathcal{R})$ (defined in Section 2). That is, given a rule set $\mathcal{R}$ and an $\mathcal{R}$-derivation δ, we may permute all applications of rules serving as sources in $G(\mathcal{R})$ (which do not depend on any rules in $\mathcal{R}$) to the beginning of δ, followed by all rule applications that depend only on sources, and so forth, with any applications of rules serving as sinks in $G(\mathcal{R})$ concluding the derivation. For example, in the graph of rule dependencies of $\mathcal{R}_2$, the rules ρ_1, ρ_2, and ρ_3 serve as source nodes (they do not depend on any rules in $\mathcal{R}_2$) and the rule ρ_4 is a sink node depending on each of the aforementioned three rules, i.e. $G(\mathcal{R}_2) = (V, E)$ with $V = \{\rho_1, \rho_2, \rho_3, \rho_4\}$ and $E = \{(\rho_i, \rho_4) \mid 1 \leq i \leq 3\}$. Hence, in any given $\mathcal{R}_2$-derivation δ, any application of ρ_1, ρ_2, or ρ_3 can be permuted backward (toward the beginning of δ) and any application of ρ_4 can be permuted forward (toward the end of δ).

Beyond the use of rule permutations, we also transform $\mathcal{R}_2$-derivations by making use of rule replacements. In particular, observe that $\mathit{head}(\rho_3)$ and $\mathit{body}(\rho_3)$ correspond to conjunctions of $\mathit{head}(\rho_1)$ and $\mathit{head}(\rho_2)$, and $\mathit{body}(\rho_1)$ and $\mathit{body}(\rho_2)$, respectively. Thus, we can replace the first application of ρ_1 and the succeeding application of ρ_2 in δ'_1 above by a single application of ρ_3, thus yielding the $\mathcal{R}_2$-derivation $\delta''_1 = \mathcal{D}_\dagger, (\rho_3, h, \mathcal{I}_1 \cup (\mathcal{I}_3 \setminus \mathcal{I}_2)), (\rho_1, h_2, \mathcal{I}_3), (\rho_4, h_4, \mathcal{I}_4)$, where $h(x) = a$ and $h(y) = b$. Interestingly, inspecting the above $\mathcal{R}_2$-derivation, one will find that it is identical to the greedy $\mathcal{R}_2$-derivation δ_2 defined earlier in the section, and so, we have shown how to take a non-greedy $\mathcal{R}_2$-derivation (viz. δ_1) and transform it into a greedy $\mathcal{R}_2$-derivation (viz. δ_2) by means of rule permutations and replacements. In the same way, one can prove in general that any non-greedy $\mathcal{R}_2$-derivation can be transformed into a greedy $\mathcal{R}_2$-derivation, thus giving rise to the following theorem, and demonstrating that $\mathcal{R}_2$ is indeed **wgbts**.

Theorem 1. *$\mathcal{R}_2$ is* **wgbts***, but not* **gbts***. Thus,* **wgbts** *properly subsumes* **gbts***.*

4 Derivation Graphs

We now discuss *derivation graphs* – a concept introduced by Baget et al. [5] and used to establish that certain classes of rule sets (e.g. weakly frontier guarded rule sets [6]) exhibit universal models of bounded treewidth. A derivation graph has the structure of a directed acyclic graph and encodes *how* atoms are derived throughout the course of an $\mathcal{R}$-derivation. By applying so-called *reduction operations*, a derivation graph may (under certain conditions) be transformed into a treelike graph that serves as a tree decomposition of an $\mathcal{R}$-derivable instance.

Below, we define derivation graphs and discuss how such graphs are transformed into tree decompositions by means of reduction operations. To increase comprehensibility, we provide an example of a derivation graph (shown in Figure 2) and give an example of applying each reduction operation (shown in Figure 3). After, we identify two (query-decidable) classes of rule sets on the basis of derivation graphs, namely, *cycle-free derivation graph sets* (**cdgs**) and *weakly cycle-free derivation graph sets* (**wcdgs**). Despite their prima facie distinctness, the **cdgs** and **wcdgs** classes coincide with **gbts** and **wgbts** classes, respectively, thus showing how the latter classes can be characterized in terms of derivation graphs. Let us now formally define derivation graphs, and after, we will demonstrate the concept by means of an example.

Definition 3 (Derivation Graph). *Let $\mathcal{D}$ be a database, $\mathcal{R}$ be a rule set, $C = \mathbf{Con}(\mathcal{D}, \mathcal{R})$, and δ be some $\mathcal{R}$-derivation $\mathcal{D}, (\rho_1, h_1, \mathcal{I}_1), \dots, (\rho_n, h_n, \mathcal{I}_n)$. The* derivation graph of δ *is the tuple $G_\delta := (\mathrm{V}, \mathrm{E}, \mathrm{At}, \mathrm{L})$, where $\mathrm{V} := \{X_0, \dots, X_n\}$ is a finite set of* nodes, *$\mathrm{E} \subseteq \mathrm{V} \times \mathrm{V}$ is a set of* arcs, *and the functions $\mathrm{At} : \mathrm{V} \to 2^{\mathcal{I}_n}$ and $\mathrm{L} : \mathrm{E} \to 2^{\mathbf{Ter}(\mathcal{I}_n)}$ decorate nodes and arcs, respectively, such that:*

1. *$\mathrm{At}(X_0) := \mathcal{D}$ and $\mathrm{At}(X_i) = \mathcal{I}_i \setminus \mathcal{I}_{i-1}$;*
2. *$(X_i, X_j) \in \mathrm{E}$ iff there is a $p(\mathbf{t}) \in \mathrm{At}(X_i)$ and a frontier atom $p(\mathbf{t}')$ in ρ_j such that $h_j(p(\mathbf{t}')) = p(\mathbf{t})$. We then set $\mathrm{L}(X_i, X_j) = \Big(h_j\big(\mathbf{Var}(p(\mathbf{t}')) \cap fr(\rho_j)\big)\Big) \setminus C$.*

We refer to X_0 as the initial node *and define the set of* non-constant terms *associated with a node to be $\overline{C}(X) = \mathbf{Ter}(X) \setminus C$ where $\mathbf{Ter}(X_i) := \mathbf{Ter}(\mathrm{At}(X_i)) \cup C$.*

Toward an example, assume $\mathcal{D}_\ddagger = \{p(a, b)\}$ and $\mathcal{R}_3 = \{\rho_1, \rho_2, \rho_3, \rho_4\}$ where

$$\begin{aligned} \rho_1 &= p(x, y) \to \exists z.q(y, z) & \rho_3 &= r(x, y) \wedge q(z, x) \to s(x, y) \\ \rho_2 &= q(x, y) \to \exists z.(r(x, y) \wedge r(y, z)) & \rho_4 &= r(x, y) \wedge s(z, w) \to t(y, w) \end{aligned}$$

Let us consider the following derivation:

$$\delta = \mathcal{D}_\ddagger, (\rho_1, h_1, \mathcal{I}_1), (\rho_2, h_2, \mathcal{I}_2), (\rho_3, h_3, \mathcal{I}_3), (\rho_4, h_4, \mathcal{I}_4) \quad \text{with}$$

$$\mathcal{I}_4 = \{\underbrace{p(a, b)}_{\mathcal{D}_\ddagger}, \underbrace{q(b, z_0)}_{\mathcal{I}_1 \setminus \mathcal{D}_\ddagger}, \underbrace{r(b, z_0), r(z_0, z_1)}_{\mathcal{I}_2 \setminus \mathcal{I}_1}, \underbrace{s(z_0, z_1)}_{\mathcal{I}_3 \setminus \mathcal{I}_2}, \underbrace{t(z_0, z_1)}_{\mathcal{I}_4 \setminus \mathcal{I}_3}\} \quad \text{and}$$

$h_1 = \{x \mapsto a, y \mapsto b\}$, $h_2 = \{x \mapsto b, y \mapsto z_0\}$, $h_3 = \{x \mapsto z_0, y \mapsto z_1, z \mapsto b\}$, as well as $h_4 = \{x \mapsto b, y \mapsto z_0, z \mapsto z_0, w \mapsto z_1\}$. The derivation graph $G_\delta = (\mathrm{V}, \mathrm{E}, \mathrm{At}, \mathrm{L})$ corresponding to δ is shown in Figure 2 and has fives nodes, $\mathrm{V} = \{X_0, X_1, X_2, X_3, X_4\}$. Each node $X_i \in \mathrm{V}$ is associated with a set $\mathrm{At}(X_i)$ of atoms depicted in the associated circle (e.g. $\mathrm{At}(X_2) = \{r(b, z_0), r(z_0, z_1)\}$), and each arc $(X_i, X_j) \in \mathrm{E}$ is represented as a directed arrow with $\mathrm{L}(X_i, X_j)$ shown as the associated set of terms (e.g. $\mathrm{L}(X_3, X_4) = \{z_1\}$). For each node $X_i \in \mathrm{V}$, the set $\mathbf{Ter}(X_i)$ of terms associated with the node is equal to $\mathbf{Ter}(\mathrm{At}(X_i)) \cup \{a, b\}$ (e.g. $\mathbf{Ter}(X_3) = \{z_0, z_1, a, b\}$) since $C = \mathbf{Con}(\mathcal{D}_\ddagger, \mathcal{R}_3) = \{a, b\}$.

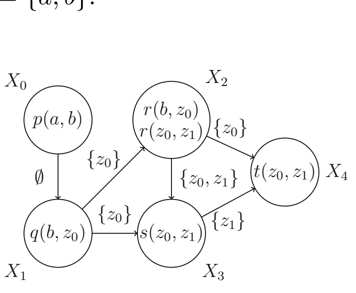

Fig. 2. The derivation graph G_δ.

As can be witnessed via the above example, derivation graphs satisfy a set of properties akin to those characterizing tree decompositions [5, Proposition 12].

Lemma 2 (Decomposition Properties). *Let $\mathcal{D}$ be a database, $\mathcal{R}$ be a rule set, and $C = \mathbf{Con}(\mathcal{D}, \mathcal{R})$. If $\mathcal{D} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}$, then G_δ satisfies the following properties:*

1. $\bigcup_{X_n \in \mathrm{V}} \mathbf{Ter}(X_n) = \mathbf{Ter}(\mathcal{I})$;
2. *For each $p(\mathbf{t}) \in \mathcal{I}$, there is an $X_n \in \mathrm{V}$ such that $p(\mathbf{t}) \in \mathrm{At}(X_n)$;*
3. *For each term $x \in \overline{C}(\mathcal{I})$, the subgraph of G_δ induced by the nodes X_n such that $x \in \overline{C}(X_n)$ is connected;*
4. *For each $X_n \in \mathrm{V}$ the size of $\mathbf{Ter}(X_n)$ is bounded by an integer that only depends on the size of $(\mathcal{D}, \mathcal{R})$, viz.* $\max\{|\mathbf{Ter}(\mathcal{D})|, |\mathbf{Ter}(head(\rho_i))|_{\rho_i \in \mathcal{R}}\} + |C|$.

Let us now introduce our set of *reduction operations*. As remarked above, in certain circumstances such operations can be used to transform derivation graphs into tree decompositions of an instance.

We make use of three reduction operations, namely, (i) *arc removal*, denoted $(\mathsf{ar})^{[i,j]}$, (ii) *term removal*, denoted $(\mathsf{tr})^{[i,j,k,t]}$, and (iii) *cycle removal*, denoted $(\mathsf{cr})^{[i,j,k,\ell]}$. The first two reduction operations were already proposed by Baget et al. [5] (they presented (tr) and (ar) as a single operation called *redundant arc removal*), whereas cycle removal is introduced by us as a new operation that will assist us in characterizing **gbts** and **wgbts** in terms of derivation graphs.[2]

Definition 4 (Reduction Operations). *Let $\mathcal{D}$ be a database, $\mathcal{R}$ be a rule set, $\mathcal{D} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}_n$, and G_δ be the derivation graph of δ. We define the set* RO *of* reduction operations *as* $\{(\mathsf{ar})^{[i,j]}, (\mathsf{tr})^{[i,j,k,t]}, (\mathsf{cr})^{[i,j,k,\ell]} \mid i,j,k,\ell \leq n,\ t \in \mathbf{Ter}(\mathcal{I}_n)\}$, *whose effect is further specified below. We let $(\mathsf{r})\Sigma(G_\delta)$ denote the output of applying the operation (r) to the (potentially reduced) derivation graph $\Sigma(G_\delta) = (\mathrm{V}, \mathrm{E}, \mathrm{At}, \mathrm{L})$, where $\Sigma \in \mathsf{RO}^*$ is a* reduction sequence*, that is, Σ is a (potentially empty) sequence of reduction operations.*

1. *Arc Removal $(\mathsf{ar})^{[i,j]}$: Whenever $(X_i, X_j) \in \mathrm{E}$ and $\mathrm{L}(X_i, X_j) = \emptyset$, then $(\mathsf{ar})^{[i,j]}\Sigma(G_\delta) := (\mathrm{V}, \mathrm{E}', \mathrm{At}, \mathrm{L}')$ where $\mathrm{E}' := \mathrm{E} \setminus \{(X_i, X_j)\}$ and $\mathrm{L}' = \mathrm{L} \restriction \mathrm{E}'$.*
2. *Term Removal $(\mathsf{tr})^{[i,j,k,t]}$: If $(X_i, X_k), (X_j, X_k) \in \mathrm{E}$ with $X_i \neq X_j$ and $t \in \mathrm{L}(X_i, X_k) \cap \mathrm{L}(X_j, X_k)$, then $(\mathsf{tr})^{[i,j,k,t]}\Sigma(G_\delta) := (\mathrm{V}, \mathrm{E}, \mathrm{At}, \mathrm{L}')$ where L' is obtained from L by removing t from $\mathrm{L}(X_j, X_k)$.*
3. *Cycle Removal $(\mathsf{cr})^{[i,j,k,\ell]}$: If $(X_i, X_k), (X_j, X_k) \in \mathrm{E}$ and there exists a node $X_\ell \in \mathrm{V}$ with $\ell < k$ such that $\mathrm{L}(X_i, X_k) \cup \mathrm{L}(X_j, X_k) \subseteq \mathbf{Ter}(X_\ell)$ then, $(\mathsf{cr})^{[i,j,k,\ell]}\Sigma(G_\delta) := (\mathrm{V}, \mathrm{E}', \mathrm{At}, \mathrm{L}')$ where*

$$\mathrm{E}' := \big(\mathrm{E} \setminus \{(X_i, X_k), (X_j, X_k)\}\big) \cup \{(X_\ell, X_k)\}$$

   and L' is obtained from $\mathrm{L} \restriction \mathrm{E}'$ by setting $L(X_\ell, X_k)$ to $\mathrm{L}(X_i, X_k) \cup \mathrm{L}(X_j, X_k)$.

[2] Beyond (tr) and (ar), we note that Baget et al. [5] introduced an additional reduction operation, referred to as *arc contraction*. We do not consider this rule here however as it is unnecessary to characterize **gbts** and **wgbts** in terms of derivation graphs and prima facie obstructs the proof of Theorem 2.

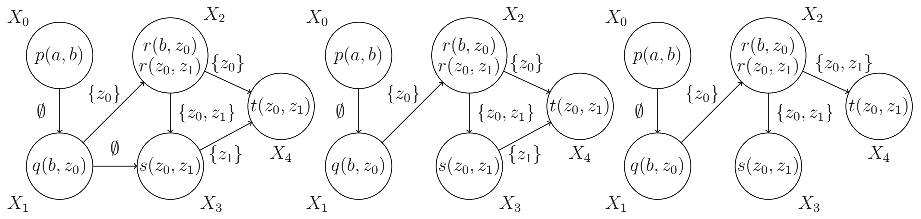

Fig. 3. Left to right: reduced derivation graphs $(\mathsf{tr})(G_\delta)$, $(\mathsf{ar})(\mathsf{tr})(G_\delta)$, and $(\mathsf{cr})(\mathsf{ar})(\mathsf{tr})(G_\delta)$.

Last, we say that a reduction sequence $\Sigma \in \mathsf{RO}^*$ *is a* complete reduction sequence *relative to a derivation graph* G_δ *iff* $\Sigma(G_\delta)$ *is cycle-free.*

Remark 2. When there is no danger of confusion, we will take the liberty to write (tr), (ar), and (cr) without superscript parameters. That is, given a derivation graph G_δ, the (reduced) derivation graph $(\mathsf{cr})(\mathsf{tr})(G_\delta)$ is obtained by applying an instance of (tr) followed by an instance of (cr) to G_δ. When applying a reduction operation we always explain *how* it is applied, so the exact operation is known.

We now describe the functionality of each reduction operation and illustrate each by means of an example. We will apply each to transform the derivation graph G_δ (shown in Figure 2) into a tree decomposition of $\mathcal{I}_4$ (which was defined above). The (tr) operation deletes a term t within the intersection of the sets labeling two converging arcs. For example, we may apply (tr) to the derivation graph G_δ from Figure 2, deleting the term z_0 from the label of the arc (X_1, X_3), and yielding the reduced derivation graph $(\mathsf{tr})(G_\delta)$, which is shown first in Figure 3. We may then apply (ar) to $(\mathsf{tr})(G_\delta)$, deleting the arc (X_1, X_3), which is labeled with the empty set, to obtain the reduced derivation graph $(\mathsf{ar})(\mathsf{tr})(G_\delta)$ shown middle in Figure 3.

The (cr) operation is more complex and works by considering two converging arcs (X_i, X_k) and (X_j, X_k) in a (reduced) derivation graph. If there exists a node X_ℓ whose index ℓ is less than the index k of the child node X_k and $\mathrm{L}(X_i, X_k) \cup \mathrm{L}(X_j, X_k) \subseteq \mathbf{Ter}(X_\ell)$, then the converging arcs (X_i, X_k) and (X_j, X_k) may be deleted and the arc (X_ℓ, X_k) introduced and labeled with $\mathrm{L}(X_i, X_k) \cup \mathrm{L}(X_j, X_k)$. As an example, the reduced derivation graph $(\mathsf{cr})(\mathsf{ar})(\mathsf{tr})(G_\delta)$ (shown third in Figure 3) is obtained from $(\mathsf{ar})(\mathsf{tr})(G_\delta)$ (shown middle in Figure 3) by applying (cr) in the following manner to the convergent arcs (X_2, X_4) and (X_3, X_4): since for X_2 (whose index 2 is less than the index 4 of X_4) $\mathrm{L}(X_2, X_4) \cup \mathrm{L}(X_3, X_4) \subseteq \mathbf{Ter}(X_2)$, we may delete the arcs (X_2, X_4) and (X_3, X_4) and introduce the arc (X_2, X_4) labeled with $\mathrm{L}(X_2, X_4) \cup \mathrm{L}(X_3, X_4) = \{z_0\} \cup \{z_1\} = \{z_0, z_1\}$. Observe that the reduced derivation graph $(\mathsf{cr})(\mathsf{ar})(\mathsf{tr})(G_\delta)$ is free of cycles, witnessing that $\Sigma = (\mathsf{cr})(\mathsf{ar})(\mathsf{tr})$ is a complete reduction sequence relative to G_δ. Moreover, if we replace each node by the set of its terms and disregard the labels on arcs, then $\Sigma(G_\delta)$ can be read as a tree decomposition of $\mathcal{I}_4$. In fact, one can show that every reduced derivation graph satisfies the decomposition properties mentioned in Lemma 2 above.

Lemma 3. *Let $\mathcal{D}$ be a database and $\mathcal{R}$ be a rule set. If $\mathcal{D} \xrightarrow{\mathcal{R}}_{\delta} \mathcal{I}$, then for any reduction sequence Σ, $\Sigma(G_{\delta}) = (\mathrm{V}, \mathrm{E}, \mathrm{At}, \mathrm{L})$ satisfies the decomposition properties 1-4 in Lemma 2.*

As illustrated above, derivation graphs can be used to derive tree decompositions of $\mathcal{R}$-derivable instances. By the fourth decomposition property (see Lemma 2 above), the width of such a tree decomposition is bounded by a constant that depends only on the given knowledge base. Thus, if a rule set $\mathcal{R}$ always yields derivation graphs that are reducible to *cycle-free* graphs – meaning that (un)directed cycles do not occur within the graph – then all $\mathcal{R}$-derivable instances have tree decompositions that are uniformly bounded by a constant. This establishes that the rule set $\mathcal{R}$ falls within the **bts** class, confirming that query entailment is decidable with $\mathcal{R}$. We define two classes of rule sets by means of reducible derivation graphs:

Definition 5 ((Weakly) Cycle-free Derivation Graph Set). *A rule set $\mathcal{R}$ is a* cycle-free derivation graph set (**cdgs**) *iff if $\mathcal{D} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}$, then G_{δ} can be reduced to a cycle-free graph by the reduction operations. $\mathcal{R}$ is a* weakly cycle-free derivation graph set (**wcdgs**) *iff if $\mathcal{D} \xrightarrow{\delta}_{\mathcal{R}} \mathcal{I}$, then there is a derivation δ' where $\mathcal{D} \xrightarrow{\delta'}_{\mathcal{R}} \mathcal{I}$ and $G_{\delta'}$ can be reduced to a cycle-free graph by the reduction operations.*

It is straightforward to confirm that **wcdgs** subsumes **cdgs**, and that both classes are subsumed by **bts**.

Proposition 1. *Every* **cdgs** *rule set is* **wcdgs** *and every* **wcdgs** *rule set is* **bts**.

Furthermore, as mentioned above, **gbts** and **wgbts** coincide with **cdgs** and **wcdgs**, respectively. By making use of the (cr) operation, one can show that the derivation graph of any greedy derivation is reducible to a cycle-free graph, thus establishing that **gbts** $\subseteq$ **cdgs** and **wgbts** $\subseteq$ **wcdgs**. To show the converse (i.e. that **cdgs** $\subseteq$ **gbts** and **wcdgs** $\subseteq$ **wgbts**) however, requires more work. In essence, one shows that for every (non-source) node X_i in a cycle-free (reduced) derivation graph there exists another node X_j such that $j < i$ and the frontier of the atoms in $\mathrm{At}(X_i)$ only consist of constants and/or nulls introduced by the atoms in $\mathrm{At}(X_j)$. This property is preserved under *reverse* applications of the reduction operations, and thus, one can show that if a derivation graph is reducible to a cycle-free graph, then the above property holds for the original derivation graph, implying that the derivation graph encodes a greedy derivation. Based on such arguments, one can prove the following:

Theorem 2. **gbts** *coincides with* **cdgs** *and* **wgbts** *coincides with* **wcdgs**. *Membership in* **cdgs**, **gbts**, **wcdgs**, *or* **wgbts** *warrants decidable BCQ entailment.*

Note that by Theorem 1, this also implies that **wcdgs** properly contains **cdgs**.

An interesting consequence of the above theorem concerns the redundancy of (ar) and (tr) in the presence of (cr). In particular, since we know that (i) if a derivation graph can be reduced to a cycle-free graph, then the derivation graph encodes a greedy derivation, and (ii) the derivation graph of any greedy derivation can be reduced to an cycle-free graph by means of applying the (cr) operation only, it follows that if a derivation graph can be reduced to a cycle-free graph, then it can be reduced by only applying the (cr) operation. We refer to this phenomenon as *reduction-admissibility*, which is defined below.

Definition 6 (Reduction-admissible). *Suppose* $S_1 = \{(\mathsf{r}_i) \mid 1 \leq i \leq n\}$ *and* $S_2 = \{(\mathsf{r}_j) \mid n+1 \leq j \leq k\}$ *are two sets of reduction operations. We say that* S_1 *is* reduction-admissible *relative to* S_2 *iff for any rule set* $\mathcal{R}$ *and* $\mathcal{R}$*-derivation* δ*, if* G_δ *is reducible to a cycle-free graph with* $S_1 \cup S_2$*, then* G_δ *is reducible to a cycle-free graph with just* S_2*.*

Corollary 1. $\{(\mathsf{tr}), (\mathsf{ar})\}$ *is reduction-admissible relative to* (cr)*.*

5 Conclusion

In this paper, we revisited the concept of a *greedy* derivation, which immediately gives rise to a bounded-width tree decomposition of the constructed instance. This well-established notion allows us to categorize rule sets as being *(weakly) greedy bounded treewidth sets* ((**w**)**gbts**), if all (some) derivations of a derivable instance are guaranteed to be greedy, irrespective of the underlying database. By virtue of being subsumed by **bts**, these classes warrant decidability of BCQ entailment, while at the same time subsuming various popular rule languages, in particular from the guarded family.

By means of an example together with a proof-theoretic argument, we exposed that **wgbts** strictly generalizes **gbts**. In pursuit of a better understanding and more workable methods to detect and analyze (**w**)**gbts** rule sets, we resorted to the previously proposed notion of *derivation graphs*. Through a refinement of the set of reduction methods for derivation graphs, we were able to make more advanced use of this tool, leading to the definition of *(weakly) cycle-free derivation graph sets* ((**w**)**cdgs**) of rules, of which we were then able to show the respective coincidences with (**w**)**gbts**. This way, we were able to establish alternative characterizations of **gbts** and **wgbts** by means of derivation graphs. En passant, we found that the newly introduced *cycle removal* reduction operation over derivation graphs is sufficient by itself and makes the other operations redundant.

For future work, we plan to put our newly found characterizations to use. In particular, we aim to investigate if a rule set's membership in **gbts** or **wgbts** is decidable. For **gbts**, this has been widely conjectured, but never formally established. In the positive case, derivation graphs might also be leveraged to pinpoint the precise complexity of the membership problem. We are also confident that the tools and insights in this paper – partially revived, partially upgraded, partially newly developed – will prove useful in the greater area of static analysis of existential rule sets. On a general note, we feel that the field of proof theory has a lot to offer for knowledge representation, whereas the cross-fertilization between these disciplines still appears to be underdeveloped.

References

1. Baget, J.F.: Improving the forward chaining algorithm for conceptual graphs rules. In: Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning. p. 407–414. KR'04, AAAI Press (2004)
2. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence. p. 677–682. IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009)
3. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence. p. 677–682. IJCAI'09, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2009)
4. Baget, J.F., Mugnier, M.L.: Extensions of simple conceptual graphs: the complexity of rules and constraints. Journal of Artificial Intelligence Research **16**, 425–465 (2002)
5. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence **175**(9), 1620–1654 (2011). https://doi.org/10.1016/j.artint.2011.03.002
6. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. Journal of Artificial Intelligence Research **48**, 115–174 (2013). https://doi.org/10.1613/jair.3873
7. Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. In: Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC'81). pp. 342–354. ACM (1981). https://doi.org/10.1145/800076.802488
8. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Lenzerini, M., Lembo, D. (eds.) Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'08). pp. 149–158. ACM (2008). https://doi.org/10.1145/1376916.1376938
9. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theoretical Computer Science **336**(1), 89–124 (2005). https://doi.org/10.1016/j.tcs.2004.10.033, database Theory
10. Feller, T., Lyon, T.S., Ostropolski-Nalewaja, P., Rudolph, S.: Finite-Cliquewidth Sets of Existential Rules: Toward a General Criterion for Decidable yet Highly Expressive Querying. In: Geerts, F., Vandevoort, B. (eds.) 26th International Conference on Database Theory (ICDT 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 255, pp. 18:1–18:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.ICDT.2023.18
11. Johnson, D.S., Klug, A.: Testing containment of conjunctive queries under functional and inclusion dependencies. In: Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. p. 164–169. PODS '82, Association for Computing Machinery, New York, NY, USA (1982). https://doi.org/10.1145/588111.588138
12. Lyon, T.S., Rudolph, S.: Derivation-graph-based characterizations of decidable existential rule sets. CoRR (2023). https://doi.org/10.48550/arXiv.2307.08481
13. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Trans. Database Syst. **4**(4), 455–469 (December 1979). https://doi.org/10.1145/320107.320115

14. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12). AAAI (2012), http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4542