Forgetting and Unfolding for Existential Rules

Zhe Wang, Kewen Wang
School of Information and Communication Technology
Griffith University, Australia

Xiaowang Zhang
School of Computer Science and Technology
Tianjin University, China

Abstract

Existential rules, a family of expressive ontology languages, inherit desired expressive and reasoning properties from both description logics and logic programming. On the other hand, forgetting is a well studied operation for ontology reuse, obfuscation and analysis. Yet it is challenging to establish a theory of forgetting for existential rules. In this paper, we lay the foundation for a theory of forgetting for existential rules by developing a novel notion of unfolding. In particular, we introduce a definition of forgetting for existential rules in terms of query answering and provide a characterisation of forgetting by the unfolding. A result of forgetting may not be expressible in existential rules, and we then capture the expressibility of forgetting by a variant of boundedness. While the expressibility is undecidable in general, we identify a decidable fragment. Finally, we provide an algorithm for forgetting in this fragment.

Introduction

Existential rules (a.k.a. Datalog$^{\pm}$ rules and tuple-generating dependencies in Databases) (Baget et al. 2011; Calì, Gottlob, and Lukasiewicz 2012), have recently been rediscovered as a promising family of ontology languages for ontology-based query answering and attracted a great deal of interest. Existential rules are sufficiently expressive to describe ontologies in datalog, $\mathcal{EL}$ and the DL-Lite family (Calì, Gottlob, and Lukasiewicz 2012), which underpin the three profiles of OWL 2 web ontology language; and reasoning for existential rules benefits from the rich legacy of rule-based systems. However, issues of ontology maintenance, including module extraction, ontology reuse and ontology change, are much less studied for existential rules.

Forgetting (Lin and Reiter 1994) is an operation of eliminating or hiding a certain set Σ of non-logical symbols (deemed to be irrelevant or private) from a knowledge base Π to obtain a new knowledge base Π' that contains no occurrence of the symbols from Σ and preserves all relevant logical consequences of Π over the remaining symbols. It has found its application in ontology maintenance, such as ontology comparison, version control, reuse, obfuscation, and revision (Konev, Walther, and Wolter 2009; Lutz and Wolter 2011; Ludwig and Konev 2014; Wang, Wang, and Topor 2015). However, due to its technical challenge, a theory of forgetting for existential rules is still missing in the literature, while researchers are aware of this problem and its importance.

Most of approaches in description logics define forgetting to preserve concept inclusions, i.e., Π' entails the same concept inclusions outside Σ as Π does (Lutz and Wolter 2011; Lutz, Seylan, and Wolter 2012; Wang et al. 2014; Nikitina and Rudolph 2014; Ludwig and Konev 2014; Koopmann and Schmidt 2014), which we refer to as inclusion-based forgetting. Inclusion-based forgetting is useful for ontological schema reasoning, but is not suitable for query answering. Others define forgetting to preserve models, i.e., the models of Π' and Π have the same interpretation on symbols outside Σ (Wang et al. 2008; Zhao and Schmidt 2016), which we call model-based forgetting. Model-based forgetting is the strongest form of forgetting in that it requires Π' to preserve all second-order entailment of Π (outside Σ) (Romero et al. 2016), which is too strong for ontology-based query answering and often renders the results of forgetting inexpressible in first-order logic. It seems a definition of forgetting that preserves query answering (Konev, Walther, and Wolter 2009; Wang et al. 2010), called query-based forgetting, is more suitable for existential rules.

To develop a theory of forgetting, one also has to deal with two important issues—expressibility and computation of forgetting. The expressibility concerns whether a result of forgetting can be expressed as a finite theory in the same language of the initial ontology, and inexpressibility was discovered even for rather simple ontologies. For instance, it is shown that model- and query-based forgetting about a role (i.e., binary predicate) are in general inexpressible for DL-Lite ontologies (Wang et al. 2010), and all the three notions of forgetting are generally inexpressible for $\mathcal{EL}$ ontologies (Konev, Walther, and Wolter 2009). Regarding the computation of forgetting, several algorithms have been proposed for various description logics. Algorithms to inclusion-based forgetting are proposed based on automata theory (Lutz and Wolter 2011; Lutz, Seylan, and Wolter 2012) and regular tree language (Nikitina and Rudolph 2014) mostly for deriving theoretical bounds. More practical algorithms are developed based on resolution (Wang et al. 2010; Ludwig and Konev 2014; Koopmann and Schmidt 2014) and the Ackermann approach

(Zhao and Schmidt 2016). We note that these approaches usually rely on the tree-model property, restricted predicate arity, or certain normal forms of description logics. Since predicates of arbitrary arity and models of complex structures are allowed, it is unclear how existing results for forgetting can be extended to existential rules.

In this paper we tackle the above open problem by establishing a theory of forgetting for existential rules. We investigate some important issues for forgetting, including (i) suitable definition and syntactic characterisation of forgetting; (ii) expressibility of forgetting; and (iii) practical computation of forgetting. Our contribution to forgetting in existential rules in this paper can be summarised as follows.

1. Based on query answering, we introduce a definition of forgetting for existential rules.
2. To provide a syntactic characterisation for the notion of forgetting, we introduce a novel form of unfolding for existential rules and show that the forgetting can be captured by the unfolding. Unfolding is a standard and important technique in logic programming. But it is non-trivial to define a suitable form of unfolding for existential rules due to the presence of existential quantifiers and conjunctions in the rule heads.
3. We show that the expressibility of forgetting is undecidable; yet for ontologies that have acyclic rule dependency, forgetting is always expressible. These results are obtained by establishing a connection between the expressibility of forgetting and a variant of boundedness, which again is an important notion in datalog but less explored for existential rules.
4. We develop an algorithm of unfolding, which actually provides an algorithm for computing forgetting for existential rules. Our algorithm first transforms the existential rules into datalog rules and tracks the application of those datalog rules. This generates a graph of rule application (GRA), which extends the existing notion of graphs of rule dependency (GRD). By establishing the termination and correctness of our algorithm on ontologies with acyclic GRA, we obtain a larger class of ontologies than the known class of acyclic GRD, on which forgetting is always expressible.

Preliminaries

We assume standard first-order logic notions, such as predicates, constants, variables, terms, (ground) atoms, formulas, entailment ($\models$) and equivalence ($\equiv$). A signature is a set of predicates; for a formula ϕ, $\mathsf{sig}(\phi)$ denotes the signature of ϕ, which naturally extends to sets of formulas. An *instance* is a (possibly infinite) set of atoms. For brevity, we often identify a finite instance with the conjunction of its atoms where all the variables are existentially quantified, and vice versa. For an instance I, $\mathsf{var}(I)$ denotes the variables in I. A *dataset* is a finite ground instance.

A substitution, expressed as a (possibly empty) set $\{t_1/t'_1, \ldots, t_n/t'_n\}$, is a functional mapping between two sets of terms $\{t_1, \ldots, t_n\}$ and $\{t'_1, \ldots, t'_n\}$. For a set T of terms, $\sigma|_T$ denotes the substitution $\{t/t' \in \sigma \mid t \in T\}$. If t is a constant or is not in the domain of σ, we write $t\sigma = t$; otherwise if $t/t' \in \sigma$, $t\sigma = t'$; and it naturally extends to (sets of) atoms and formulas. A *homomorphism* from an instance I to an instance I' is a substitution σ from the terms occurring in I to those in I' such that $I\sigma \subseteq I'$. A *unifier* between two instances I and I' is a substitution τ such that $I\tau = I'\tau$; and τ is a *most general* unifier satisfying condition C if for each unifier τ' between I and I' satisfying C, there exists a substitution σ such that $\tau' = \tau\sigma$.

An *existential rule* (or a rule) r is a formula of the form

$$\forall \vec{x}.\forall \vec{y}.[\phi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z}.\psi(\vec{x}, \vec{z})]$$

where $\vec{x}$, $\vec{y}$ and $\vec{z}$ are pairwise disjoint vectors of variables, and $\phi(\vec{x}, \vec{y})$ and $\psi(\vec{x}, \vec{z})$ are conjunctions of atoms with variables from respectively $\vec{x} \cup \vec{y}$ and $\vec{x} \cup \vec{z}$. Formula ϕ is the *body* of the rule r, denoted $\mathsf{body}(r)$, and formula ψ is the *head* of r, denoted $\mathsf{head}(r)$; again, they can be seen as (existentially quantified conjunctions of) sets of atoms. For brevity, universal quantifiers in a rule are often omitted, and we sometimes express rule r as $\mathsf{body}(r) \rightarrow \mathsf{head}(r)$. We use X_r, Y_r and Z_r to denote the sets of the variables in respectively, $\vec{x}$, $\vec{y}$ and $\vec{z}$ of rule r. A *datalog* rule r is an existential rule whose head consists of a single atom and Z_r is empty. For convenience, we assume each rule employs a disjoint set of variables from other rules and from the instances. [1] An *ontology* is a finite set of rules. For simplicity, we disallow trivial rules r, i.e., $\mathsf{body}(r) \models \mathsf{body}(r) \wedge \mathsf{head}(r)$, and duplicated rules (up to variable renaming) in an ontology. When we talk about expressibility, we refer to expressibility as ontologies.

A rule r is *applicable* to an instance I if there is a homomorphism σ from $\mathsf{body}(r)$ to I, and the result of *applying* r to I with σ is $r^{\sigma}_{\uparrow}(I) = I \cup \mathsf{head}(r)\sigma'$, where σ' is a *safe extension* of $\sigma|_{X_r}$ on Z_r, that is, $\sigma|_{X_r} \subseteq \sigma'$ and for each $z \in Z_r$, $z\sigma'$ is a distinct fresh variable not occurring in I. The *forward chaining*, a.k.a. *chase*, of a set of rules Π on I is a sequence of instances I_i $(i \geq 0)$, where $I_0 = I$, $I_{i+1} = \bigcup_{r \in \Pi, \sigma} r^{\sigma}_{\uparrow}(I_i)$. For $k \geq 0$, let $\Pi^k_{\uparrow}(I) = I_k$, and $\Pi_{\uparrow}(I) = \bigcup_{i=0}^{\infty} I_i$.

A Boolean conjunctive query (BCQ) is an existentially closed conjunction of atoms, which can be seen as a finite instance. A union of BCQ is a disjunction of BCQs, which is seen as a finite set of finite instances.

Backward chaining is defined through the notion of piece unification for a given query and a rule (Leclère, Mugnier, and Ulliana 2016). So, we give the definition of piece unification for a given instance I and a rule r. First, for a subset I' of I, we say that a variable occurring in I' but not in $I \setminus I'$ is an *exclusive variable* for I' in I. A *piece unification* of I and r is a triple $\mu = (I', H, \tau)$, where $\emptyset \subset I' \subseteq I$, $H \subseteq \mathsf{head}(r)$, and τ is a most general unifier between I' and H such that the following condition is satisfied for each $z \in Z_r$: If a term t $(t \neq z)$ in $I' \cup H$ is unified with z, i.e., $z\tau = t\tau$, then t is an exclusive variable for I' in I. We call a minimal subset I' satisfying the above conditions a *piece* of I w.r.t. r.

I is *rewritable* by r if there exists a piece unification $\mu = (I', H, \tau)$ of I and r, and the result of *rewriting* I by r with μ

[1]This is not reflected in our examples for simplicity.

is $r_{\downarrow}^{\mu}(I) = (I \setminus I')\tau \cup \mathsf{body}(r)\tau'$ where τ' is a safe extension of $\tau|_{\mathsf{var}(H)}$ on $Y_r \cup (X_r \setminus \mathsf{var}(H))$. For a set S of instances and a set Π of rules, the *backward chaining*, a.k.a. *rewriting*, of Π on S is a sequence of sets of instances S_i ($i \geq 0$), where $S_0 = S$ and $S_{i+1} = S_i \cup \{r_{\downarrow}^{\mu}(I_i) \mid r \in \Pi, I_i \in S_i\}$. For $k \geq 0$, let $\Pi_{\downarrow}^{k}(S) = S_k$, and $\Pi_{\downarrow}(S) = \bigcup_{i=0}^{\infty} S_i$.

Ontology-based query answering (König et al. 2015; Gottlob, Orsi, and Pieris 2014) concerns essentially the reasoning problem of deciding for an ontology Π, a dataset D and a BCQ Q, whether $\Pi \cup D \models Q$. Forward chaining and backward chaining for existential rules are both sound and complete for query answering. In particular, it holds that $\Pi \cup D \models Q$ iff $\Pi_{\uparrow}^{k}(D) \models Q$ for some $k \geq 0$, and iff $D \models \Pi_{\downarrow}^{l}(\{Q\})$ for some $l \geq 0$ (Leclère, Mugnier, and Ulliana 2016).

Forgetting and Unfolding

As discussed above, query answering is a primary reasoning task for ontologies in existential rules, and we adopt a query-based definition for forgetting. More specifically, a result of forgetting about a signature Σ in an ontology Π is a logically weaker ontology Π' not containing any occurrence of predicates from Σ, such that Π' preserves query answering of Π for any BCQ and any dataset outside Σ.

Definition 1 (Forgetting). *Let Π be an ontology and Σ a signature. An ontology Π' is a result of* forgetting *about Σ in Π if (1) $\mathsf{sig}(\Pi') \subseteq \mathsf{sig}(\Pi) \setminus \Sigma$, (2) $\Pi \models \Pi'$, and (3) for each dataset D and each BCQ Q such that $\mathsf{sig}(D \cup Q) \cap \Sigma = \emptyset$, $\Pi \cup D \models Q$ implies that $\Pi' \cup D \models Q$.*

We use the following running example to illustrate our definitions and methods in the rest of this paper.

Example 1. *Consider an ontology Π_e consisting of the following rules*

$$\begin{aligned} r_1 &= \mathsf{A}(x) \to \exists y.[\mathsf{B}(x,y) \wedge \mathsf{C}(x,y)], \\ r_2 &= \mathsf{C}(x,y) \wedge \mathsf{D}(x) \to \exists z.\mathsf{E}(y,z), \\ r_3 &= \mathsf{B}(x,y) \wedge \mathsf{E}(y,z) \to \mathsf{E}(x,z), \\ r_4 &= \mathsf{F}(x) \to \mathsf{D}(x). \end{aligned}$$

A result of forgetting about $\Sigma = \{\mathsf{B}, \mathsf{D}\}$ in Π_e consists of the following three rules:

$$\begin{aligned} &\mathsf{A}(x) \to \exists y.\mathsf{C}(x,y), \\ &\mathsf{C}(x,y) \wedge \mathsf{F}(x) \to \exists z.\mathsf{E}(y,z), \\ &\mathsf{A}(x) \wedge \mathsf{F}(x) \to \exists y,z.[\mathsf{C}(x,y) \wedge \mathsf{E}(y,z) \wedge \mathsf{E}(x,z)]. \end{aligned}$$

We show that a result of forgetting is unique (up to logical equivalence). Hence, we denote a result of forgetting as $\mathsf{forget}(\Pi, \Sigma)$.

Proposition 1. *If Π' and Π'' are both results of forgetting about signature Σ in ontology Π then $\Pi' \equiv \Pi''$.*

To prove this proposition, we need a lemma.

Lemma 1. *For an ontology Π and a rule r, $\Pi \models r$ iff $\Pi \cup \mathsf{body}(r)\sigma \models \mathsf{head}(r)\sigma$, where $\sigma = \{x/c_x \mid x \in \mathsf{var}(\mathsf{body}(r))\}$ and each c_x is a fresh constant for x.*

Proof Sketch for Proposition 1: For each rule $r \in \Pi' \setminus \Pi''$, let σ be as in Lemma 1. Clearly, $\Pi' \cup \mathsf{body}(r)\sigma \models \mathsf{head}(r)\sigma$. Since $\mathsf{sig}(r) \subseteq \mathsf{sig}(\Pi) \setminus \Sigma$, by the definition of forgetting, $\Pi \cup \mathsf{body}(r)\sigma \models \mathsf{head}(r)\sigma$, and also, $\Pi'' \cup \mathsf{body}(r)\sigma \models \mathsf{head}(r)\sigma$. By Lemma 1, $\Pi'' \models r$; that is, $\Pi'' \models \Pi'$. The other direction also follows similarly. □

The definition of forgetting does not provide us any clue on the existence or the computation of forgetting. In order to develop an algorithm for the forgetting, we first provide a syntactic characterisation of forgetting in terms of a form of unfolding for existential rules.

Definition 2 (Unfolding). *A rule r is* unfoldable *by a rule r' if there exists a piece unification $\mu = (B, H, \tau)$ of $\mathsf{body}(r)$ and r'. Here r and r' are not necessarily different.*

The result of unfolding *r by r' with μ is the following rule denoted $r \bowtie_{\mu} r'$:*

$$(\mathsf{body}(r) \setminus B)\tau \cup \mathsf{body}(r')\tau' \to \exists \vec{z}.[\mathsf{head}(r)\tau'' \cup \mathsf{head}(r')\tau']$$

where τ' is a safe extension of $\tau|_{\mathsf{var}(H)}$ on $\mathsf{var}(r') \setminus \mathsf{var}(H)$, τ'' is a safe extension of $\tau|_{X_r}$ on Z_r, and $\vec{z}$ consists of all the variables in the head but not in the body.

The unfold chaining *on a set of rules Π is a sequence of rule sets Π^i ($i \geq 0$), where $\Pi^0 = \Pi$ and $\Pi^{i+1} = \Pi^i \cup \{\, r \bowtie_{\mu} r' \mid r, r' \in \Pi^i \,\}$. The* unfolding *of Π is $\mathsf{unfold}(\Pi) = \bigcup_{i=0}^{\infty} \Pi^i$.*

Unlike unfolding in propositional and first-order logic programs (Zhou 2015), our definition requires piece unifications and that H is not eliminated from $\mathsf{head}(r')$ after unfolding, in order to correctly handle the existential quantifiers in the rule heads.

Example 2. *In Π_e of Example 1, r_3 is unfoldable by r_2 with piece unification $\mu = (\,\{\mathsf{E}(y,z)\}, \{\mathsf{E}(y,z)\}, \emptyset\,)$, and the result of unfolding r_3 by r_2 with μ is*

$$r = \mathsf{B}(x',y) \wedge \mathsf{C}(x,y) \wedge \mathsf{D}(x) \to \exists z.[\mathsf{E}(y,z) \wedge \mathsf{E}(x',z)].$$

And r is unfoldable by r_1 with piece unification $\mu' = (\,\{\mathsf{B}(x',y), \mathsf{C}(x,y)\}, \{\mathsf{B}(x,y), \mathsf{C}(x,y)\}, \{x'/x\}\,)$, and the result of unfolding r by r_1 with μ' is

$$\mathsf{A}(x) \wedge \mathsf{D}(x) \to \exists y,z.[\mathsf{B}(x,y) \wedge \mathsf{C}(x,y) \wedge \mathsf{E}(y,z) \wedge \mathsf{E}(x,z)].$$

Note that $\mu'' = (\,\{\mathsf{B}(x',y)\}, \{\mathsf{B}(x,y)\}, \{x'/x\}\,)$ is not a piece unification of $\mathsf{body}(r)$ and r_1, as y is not an exclusive variable for $\{\mathsf{B}(x,y)\}$ in $\mathsf{body}(r)$.

Let Π be a set of rules of the form $B_i \to H_i$ ($1 \leq i \leq n$), and we recall that distinct rules have disjoint sets of variables. An *aggregated rule* of Π is of the form $B_{l_1} \wedge \ldots \wedge B_{l_k} \to H_{l_1} \wedge \ldots \wedge H_{l_k}$ where $1 \leq l_j \leq n$ for $j = 1, \ldots, k$, and where if an aggregated rule involves multiple copies of the same rule from Π then each copy has a distinct set of variables (Leclère, Mugnier, and Ulliana 2016). The following result shows the soundness and completeness of unfolding regarding ontological reasoning.

Proposition 2. *Let Π be an ontology, r a rule, D a dataset and Q a BCQ. Then*

(1) $\Pi \models r$ *for* $r \in \mathsf{unfold}(\Pi)$.

(2) $\Pi \cup D \models Q$ *iff there are* $k \geq 0$ *and an aggregated rule* r *of* Π^k *s.t.* $\{r\} \cup D \models Q$.

(3) $\Pi \models r$ *iff* $r' \models r$ *for some* $k \geq 0$ *and some aggregated rule* r' *of* Π^k.

Proof Sketch: (1) is easy to see, and (3) can be shown by (2) and Lemma 1.

(2, $\Rightarrow$): By the completeness of backward chaining, $\Pi \cup D \models Q$ implies that $D \models \Pi^l_\downarrow(\{Q\})$ for some $l \geq 0$. If $Q' \in \Pi^l_\downarrow(\{Q\})$ is mapped to D with homomorphism σ', then by the definition of $\Pi^l_\downarrow(\{Q\})$, there are $0 \leq n \leq l$ and a sequences of BCQs Q_i ($0 \leq i \leq n$), such that $Q_0 = Q$, $Q_{i+1} = r_{i\downarrow}^{\mu_i}(Q_i)$ for some $r_i \in \Pi$ and piece unification μ_i, and $Q_n = Q'$. W.l.o.g., we assume $r_0, \ldots, r_m$ ($0 \leq m \leq n$) rewrites atoms solely from Q, and each r_i with $m < i \leq n$ rewrites body atoms of some r_j with $0 \leq j < i$. It is not hard to see that r_j is unfoldable by r_i with some piece unification μ'_i. Let $k = n - m$, we have an aggregated rule r by combining all the rules in Π^k that are obtained by unfolding $r_0, \ldots, r_m$ by $r_{m+1}, \ldots, r_n$. It is not hard to see that $\mathsf{body}(r)$ can be mapped to Q' with some homomorphism τ. Taking $\sigma = \tau\sigma'$, we have $r^\sigma_\uparrow(D) \models Q$.

(2, $\Leftarrow$): By (1), $\Pi \models r$ and hence $\Pi \cup D \models Q$. □

For an instance I and a signature Σ, $I|_\Sigma$ consists of the atoms in I that are over Σ. Let

$$\Pi|_\Sigma = \{\, \mathsf{body}(r) \to \mathsf{head}(r)|_\Sigma \mid r \in \Pi, \mathsf{head}(r)|_\Sigma \neq \emptyset, \text{ and } \mathsf{body}(r)|_\Sigma = \mathsf{body}(r) \,\}$$

and $\Pi|_{\overline{\Sigma}} = \Pi|_{\mathsf{sig}(\Pi)\setminus\Sigma}$.

The following result shows that rule unfolding captures the notion of forgetting.

Theorem 1. *For an ontology* Π *and a signature* Σ*, a result of forgetting about* Σ *in* Π *is expressible iff* $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$ *is so. In this case,* $\mathsf{forget}(\Pi, \Sigma) \equiv \mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$.

Proof Sketch: Suppose $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$ can be expressed as an ontology Π', assume $\mathsf{sig}(\Pi') \subseteq \mathsf{sig}(\Pi) \setminus \Sigma$. By Proposition 2 (1), $\Pi \models \Pi'$. For each dataset D and each BCQ Q s.t. $\mathsf{sig}(D \cup Q) \cap \Sigma = \emptyset$, by Proposition 2 (2), $\Pi \cup D \models Q$ implies that there is an aggregated rule r of $\mathsf{unfold}(\Pi)$ s.t. r is applicable to D with homomorphism σ and $r^\sigma_\uparrow(D) \models Q$. If r is obtained by aggregating $r_0, \ldots r_n$ in $\mathsf{unfold}(\Pi)$, then since r is applicable to D, $\mathsf{body}(r_i)$ is over $\mathsf{sig}(\Pi) \setminus \Sigma$ for each $0 \leq i \leq n$. Then, $r'_i = \mathsf{body}(r_i) \to \mathsf{head}(r_i)|_{\mathsf{sig}(\Pi)\setminus\Sigma}$ is in Π' for each $0 \leq i \leq n$. Let r' be the aggregated rule obtained by combining all the r'_i ($0 \leq i \leq n$). As $r^\sigma_\uparrow(D) \models Q$ and $\mathsf{sig}(Q) \cap \Sigma = \emptyset$, $r'^\sigma_\uparrow(D) \models Q$. That is, $\Pi' \cup D \models Q$. By the uniqueness of forgetting, $\mathsf{forget}(\Pi, \Sigma) \equiv \Pi'$.

Suppose $\mathsf{forget}(\Pi, \Sigma)$ can be expressed as an ontology Π', thus $\mathsf{sig}(\Pi') \subseteq \mathsf{sig}(\Pi) \setminus \Sigma$. For each $r \in \Pi'$, by the definition of forgetting, $\Pi \models r$. By Proposition 2 (3), $r' \models r$ for some aggregated rule of $\mathsf{unfold}(\Pi)$. Similar as above, there is an aggregated rule r'' of $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$ s.t. $r'' \models r$. That is, $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}} \models \Pi'$. For each dataset D and each BCQ Q s.t. $\mathsf{sig}(D \cup Q) \cap \Sigma = \emptyset$ and $\Pi \cup D \models Q$, $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}} \cup D \models Q$. By the uniqueness of forgetting, $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}} \equiv \Pi'$. □

Expressibility and Boundedness

For an ontology Π and a signature Σ, there may not exist a result of forgetting about Σ in Π. For example, consider $\Pi = \{\, \mathsf{A}(x) \to \mathsf{B}(x), \mathsf{B}(x) \wedge \mathsf{C}(x, y) \to \mathsf{B}(y), \mathsf{B}(x) \to \mathsf{D}(x) \,\}$, and Π entails $\mathsf{A}(x_1) \to \mathsf{D}(x_1)$, $\mathsf{A}(x_1) \wedge \mathsf{C}(x_1, x_2) \to \mathsf{D}(x_2)$, $\mathsf{A}(x_1) \wedge \mathsf{C}(x_1, x_2) \wedge \mathsf{C}(x_2, x_3) \to \mathsf{D}(x_3), \ldots$ Thus, a result of forgetting about $\{\mathsf{B}\}$ in Π needs to capture all these infinite number of rules, which is not expressible as a finite first-order theory. Hence, it is important to study whether the expressibility of forgetting is decidable and to identify conditions under which expressibility is guaranteed.

In this section, we show the expressibility of forgetting is undecidable for existential rules, by establishing a connection between the expressibility and a variant of boundedness called *predicate boundedness*. We also show that for a class of ontologies that are known to be bounded, their forgetting is guaranteed to be expressible.

Boundedness is a well studied notion for datalog (Cosmadakis et al. 1988; Gaifman et al. 1993), and recently for existential rules (Leclère, Mugnier, and Ulliana 2016). Formally, an ontology Π is *program bounded* if there exists $k \geq 0$ such that for each dataset D, $\Pi^k_\uparrow(D) \equiv \Pi^{k+1}_\uparrow(D)$. Program boundedness of an ontology is undecidable, which holds already for datalog programs (Gaifman et al. 1993). Special classes of bounded ontologies have been identified in the literature, a notable example being the class of *aGRD* ontologies (Baget et al. 2011; Grau et al. 2013). Given two rules r and r', r' *depends on* r if there is a dataset D, a homomorphism σ from $\mathsf{body}(r)$ to D, and a homomorphism σ' from $\mathsf{body}(r')$ to $r^\sigma_\uparrow(D)$ such that $\mathsf{body}(r')\sigma' \not\subseteq D$ and $\mathsf{head}(r')\sigma' \not\subseteq r^\sigma_\uparrow(D)$. The *graph of rule dependency (GRD)* for Π is a directed graph whose nodes are the rules in Π and whose edges are the dependency relationships between rules. We say Π is *aGRD* if its GRD is acyclic.

Since the expressibility of forgetting clearly depends on the predicates to be forgotten, we define a variant of boundedness in terms of predicates.

Definition 3 (Predicate Boundedness)**.** *An ontology* Π *is* predicate bounded *w.r.t. signatures* Σ_d *and* Σ_q *if there exists* $k \geq 0$*, for each dataset* D *over* Σ_d *and each BCQ* Q *over* Σ_q *with* $\Pi \cup D \models Q$*, the following condition is satisfied:*

For each instance I_i *and each minimal subset* $I \subseteq I_i$ *s.t.* $\Pi \cup I \models Q$*, there exists some instance* I_j *with* $j \leq i$ *and a subset* $I' \subseteq I_j$ *s.t.* $\mathsf{sig}(I') \subseteq \Sigma_d$ *and* $\Pi^k_\uparrow(I') \models I$.

Here the sequence $I_0, \ldots, I_n$ *is a forward chaining for* $\Pi \cup D \models Q$*, i.e.,* $I_0 = D$ *and* $I_n \models Q$ *with the minimum* n.

Intuitively, Π is predicate bounded w.r.t. Σ_d and Σ_q if there exists a bound $k \geq 0$ such that in the forward chaining derivation of every query Q over Σ_q from Π and some D over Σ_d, each intermediate result (i.e., I) can be derived from some facts over Σ_d (i.e, I') within k steps of forward chaining. This definition thus excludes any unbounded chain of derivation on facts outside Σ_d (while derivation in Σ_d can be unbounded). Our definition generalises the predicate boundedness for datalog (Gaifman et al. 1993), where Π is a datalog program, Σ_d consists of the EDB predicates[2], Σ_q consists of a single IDB predicate, and Q is a singleton of a ground atom. Note that in this case, I' is over Σ_d and hence must be a subset of D, and so Π is predicate bounded on Σ_q. Moreover, for existential rules, our predicate boundedness coincides program boundedness when we require $I' = D$, $\mathsf{sig}(\Pi) \subseteq \Sigma_d$ and $\mathsf{sig}(\Pi) \subseteq \Sigma_q$, which can be seen from Proposition 4 in (Leclère, Mugnier, and Ulliana 2016).

We are now able to establish a connection between predicate boundedness and the expressibility of forgetting.

Theorem 2. *For an ontology Π and a signature Σ, let $\Sigma_d = \Sigma_q = \mathsf{sig}(\Pi) \setminus \Sigma$, a result of forgetting about Σ in Π is expressible iff Π is predicate bounded w.r.t. Σ_d and Σ_q.*

To prove Theorem 2, we first define a boundedness on unfolding. A set Π of rules is *unfolding bounded* w.r.t. a signature Σ if there exists $k \geq 0$ such that $\Pi^k|_\Sigma \equiv \mathsf{unfold}(\Pi)|_\Sigma$.

Lemma 2. *For an ontology Π and a signature Σ, a result of forgetting about Σ in Π is expressible iff Π is unfolding bounded w.r.t. $\mathsf{sig}(\Pi) \setminus \Sigma$.*

Lemma 3. *Let D be a dataset, Q be a BCQ and Π be an ontology in which each rule has at most m body atoms. Then*

(1) For any $k \geq 0$, if there is an aggregated rule r of Π^k s.t. $\{r\} \cup D \models Q$ then $\Pi_\uparrow^{2^k}(D) \models Q$.

(2) For any $l \geq 0$, if $\Pi_\uparrow^l(D) \models Q$ then there is an aggregated rule r of Π^{lm} s.t. $\{r\} \cup D \models Q$.

Proof Sketch for Theorem 2: By Lemma 2, we only need to show that Π is predicate bounded w.r.t. Σ_d and Σ_q iff Π is unfolding bounded w.r.t. $\mathsf{sig}(\Pi) \setminus \Sigma$.

($\Rightarrow$): We only need to show that there exists $k \geq 0$ s.t. for each rule $r \in \mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$, $\Pi^k|_{\overline{\Sigma}} \models r$. By Proposition 2 (1), $\Pi \models r$, and by Lemma 1, $\Pi \cup \mathsf{body}(r)\sigma \models \mathsf{head}(r)\sigma$. Let $D = \mathsf{body}(r)\sigma$ and $Q = \mathsf{head}(r)\sigma$. We want to show that $\Pi^k|_{\overline{\Sigma}} \cup D \models Q$, which by Lemma 1, would imply $\Pi^k|_{\overline{\Sigma}} \models r$.

Consider the *derivation forest* F for Q where each node is a pair of the form $(f, 0)$ with $f \in D$ or (f, i) with $f \in I_i \setminus I_{i-1}$ for some $0 < i \leq n$, and each edge is labelled with $r\sigma$ with $r \in \Pi$ and σ a substitution. The roots consist of the facts in $Q\sigma'$ s.t. $Q\sigma' \subseteq I_n$. A node (f, i) has a child (g, j) connected via an edge labelled $r\sigma$ iff $f \in \mathsf{head}(r\sigma)$, $g \in \mathsf{body}(r\sigma)$ and $j < i$. We only need to show that $\Pi^k|_{\overline{\Sigma}} \cup D \models f$ for each f occurring in the derivation forest.

Consider a node (f, i) with the largest i that has a child over Σ, suppose the edge between them is labelled with $r\sigma$. For each ancestor (g, x) of (f, i), (g, x) connects to its children via edges labelled by rules whose bodies are over $\mathsf{sig}(\Pi) \setminus \Sigma$. That is, to show $\Pi^k|_{\overline{\Sigma}} \cup D \models g$, we only need to show $\Pi^k|_{\overline{\Sigma}} \cup D \models f$. Thus, (f, i) can be our starting point. Since $\Pi \cup D \models f$ and f is clearly over Σ_q, from the assumption that Π is predicate bounded w.r.t. Σ_d and Σ_q, there are $l \geq 0$ (independent of D and f) and (by taking $I = \mathsf{body}(r\sigma)$) an instance $I' \subseteq I_j$ for some $j < i$ satisfying $\mathsf{sig}(I') \subseteq \Sigma_d$ and $\Pi_\uparrow^l(I') \models \mathsf{body}(r\sigma)$. Since $\Pi_\uparrow^{l+1}(I') \models f$, by Lemma 3 (2), let $k = (l+1)m$ (m is as in Lemma 3 (2)), there is an aggregated rule r' of Π^k s.t. $\{r'\} \cup I' \models f$. Since $\mathsf{sig}(I' \cup \{f\}) \cap \Sigma = \emptyset$, $\mathsf{body}(r')$ is over $\mathsf{sig}(\Pi) \setminus \Sigma$. Thus, there is an aggregated rule r'' of $\Pi^k|_{\overline{\Sigma}}$ that can be obtained from r' by eliminating head atoms s.t. $\{r''\} \cup I' \models f$. That is, by Proposition 2 (2), $\Pi^k|_{\overline{\Sigma}} \cup I' \models f$. To show $\Pi^k|_{\overline{\Sigma}} \cup D \models f$, we only need to show that $\Pi^k|_{\overline{\Sigma}} \cup D \models I'$. We have reduced the proof to showing $\Pi^k|_{\overline{\Sigma}} \cup D \models g$ for some nodes (g, j) with $j < i$. By recursively applying the reduction, it finally reduces to $\Pi^k|_{\overline{\Sigma}} \cup D \models h$ for some nodes $(h, 0)$, which clearly holds.

($\Leftarrow$): Again, we consider the derivation forest F for Q, and w.l.o.g., we assume all the facts $f \in I$ occur in F. Note that the roots (leaves) of F consist of facts in $Q\sigma'$ (resp., D) and thus are all outside Σ. Take the sub-forest F' of F where

- the roots consist of all the nodes (f, i) outside Σ with smallest i s.t. there is a path through (f, i) in F from a root of F to a node containing a fact in I;
- the leaves consist of all the nodes (g, j) outside Σ with largest j s.t. there is a path in F from a root of F' to it;
- an edge connects two nodes in F' iff it does in F.

Let Q' consists of all such facts f, I' consists of all such facts g, m be the maximum i and n be the minimum j. We want to show that $m - n \leq k$ for the bound k. It is clear that $\Pi \cup I' \models Q'$. As $\mathsf{sig}(I' \cup Q') \cap \Sigma = \emptyset$, by the definition of forgetting, $\mathsf{forget}(\Pi, \Sigma) \cup I' \models Q'$. By Theorem 1, $\mathsf{unfold}(\Pi)|_{\overline{\Sigma}} \cup I' \models Q'$. From the assumption that Π is unfolding bounded w.r.t. Σ, there is $l \geq 0$ (independent of D and Q) s.t. $\Pi^l|_{\overline{\Sigma}} \equiv \mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$. That is, $\Pi^l|_{\overline{\Sigma}} \cup I' \models Q'$. Since that every internal node in F' are over Σ, there is an aggregated rule r' of $\Pi^l|_{\overline{\Sigma}}$ s.t. $\{r'\} \cup I' \models Q'$ (as otherwise $(\Pi^l|_{\overline{\Sigma}})_\uparrow^1(I')$ would be outside Σ and have smaller indexes than Q', leading to a contradiction). Thus, there is an aggregated rule r'' of Π^l that can be obtained from r' by adding head atoms s.t. $\{r''\} \cup I' \models Q'$. By Lemma 3 (1), let $k = 2^l$ (independent of D and Q), then $\Pi_\uparrow^k(I') \models Q'$. From the construction of Q', $\Pi_\uparrow^k(I') \models I$. □

Further, we show that program boundedness can be reduced to the expressibility of forgetting, thus establishing its undecidability. For an ontology Π, $\Pi^\dagger$ is obtained from Π by adding two rules $\mathsf{A}_l(\vec{x}) \to \mathsf{A}(\vec{x})$ and $\mathsf{A}(\vec{x}) \to \mathsf{A}_r(\vec{x})$ for each predicate A occurring in Π, where A_l and A_r are fresh predicates and have the same arity as A.

Proposition 3. *An ontology Π is program bounded iff a result of forgetting about $\mathsf{sig}(\Pi)$ in $\Pi^\dagger$ is expressible.*

As with boundedness, the expressibility of forgetting is guaranteed for aGRD ontologies.

Proposition 4. *For an aGRD ontology Π and a signature Σ, a result of forgetting about Σ in Π is expressible.*

Computation of Forgetting

Although unfolding provides a syntactic characterisation for forgetting, computation based on blind unfolding is impractical due to the huge numbers of possible rule combinations and piece unifications during complete unfolding. To provide an algorithm for forgetting, we first show that unfolding of rules can always be done in a sequential manner. An

[2]EDB predicates are those not occurring in the rule heads, and the others are IDB predicates.

unfolding sequence of an ontology Π is inductively defined as follows: each rule $r \in \Pi$ is an unfolding sequence and the result of the sequence is r; and if α is an unfolding sequence whose result is r and r is unfoldable by a rule $r' \in \Pi$ with piece unification μ, then $\alpha\mu r'$ is an unfolding sequence, and the result of the sequence is $r \bowtie_\mu r'$.

Proposition 5. *For an ontology Π, $\mathsf{unfold}(\Pi)$ consists of exactly the results of all the unfolding sequences of Π (up to variable renaming).*

Proof Sketch: We only need to show that for each rule r as a result of $r_1\mu_1(r_2 \bowtie_{\mu_2} r_3)$, where $r_i \in \mathsf{unfold}(\Pi)$ ($i = 1, 2, 3$), r is also a result of $r_1\mu_1' r_2 \mu_2' r_3$ or $r_1\mu_1' r_3 \mu_2' r_2$ for some μ_j' ($j = 1, 2$). Let $r' = r_2 \bowtie_{\mu_2} r_3$, then $\mathsf{head}(r') \subseteq (\mathsf{head}(r_2) \cup \mathsf{head}(r_3))\tau_2$. Suppose $\mu_j = (B_j, H_j, \tau_j)$ and $\mu_j' = (B_j', H_j', \tau_j')$, and for simplicity we use τ_j and τ_j' to denote their safe extensions in unfolding. $B_1 \subseteq \mathsf{body}(r_1)$ and $H_1 \subseteq \mathsf{head}(r')$, and thus $H_1 \subseteq (\mathsf{head}(r_2) \cup \mathsf{head}(r_3))\tau_2$. If $H_1 \subseteq \mathsf{head}(r_2)\tau_2$, it is not hard to see that r is a result of $r_1\mu_1' r_2 \mu_2' r_3$; and if $H_1 \subseteq \mathsf{head}(r_3)\tau_2$ then r is a result of $r_1\mu_1' r_3 \mu_2' r_2$. Otherwise, if B_1 does not contain a piece I of $\mathsf{body}(r_1)$ w.r.t. r' such that I unifies (by μ_1) with atoms from both $\mathsf{head}(r_2)\tau_2$ and $\mathsf{head}(r_3)\tau_2$, then by taking $H_1' = H_1\tau_2^- \cap \mathsf{head}(r_2)$ and $H_2' = H_2 \cup (H_1\tau_2^- \cap \mathsf{head}(r_3))$, r is a result of $r_1\mu_1' r_2 \mu_2' r_3$. If B_1 contains such a piece I, let $Z = Z_{r'} \cap \mathsf{var}(\mathsf{head}(r_2)\tau_2) \cap \mathsf{var}(\mathsf{head}(r_3)\tau_2)$. That is, Z are the existential variables in r' that "glue" atoms from the heads of r_2 and r_3 together (that lead to piece I). We want to show that $Z\tau_2^-$ are not existential variables in r_2, and thus I is not a piece w.r.t. r_2 and as a result, r_1 is unfoldable by r_2. By the definition of unfolding, $Z \subseteq \mathsf{var}(\mathsf{head}(r_2)\tau_2) \cap \mathsf{var}(\mathsf{head}(r_3)\tau_2)$ implies that $Z \subseteq \mathsf{var}(B_2\tau_2) \cap \mathsf{var}(H_2\tau_2)$. That is, $Z \subseteq \mathsf{var}(\mathsf{body}(r_2)\tau_2)$. Thus, I is not a piece w.r.t. r_2. By taking $H_1' = H_1\tau_2^- \cap \mathsf{head}(r_2)$ and $H_2' = H_2 \cup (H_1\tau_2^- \cap \mathsf{head}(r_3))$, r is a result of $r_1\mu_1' r_2 \mu_2' r_3$. $\square$

To compute forgetting, we can generate rules by searching all valid unfolding sequences. However, the search space can still be infinite, noting the large numbers of potential combinations of rules and a significant portion of unfolding sequences resulting rules with unwanted predicates. In what follows, we introduce graphs of rule application (GRA), which can be constructed by tracking application of datalog rules, and use GRA to guide the unfolding in order to narrow down the search.

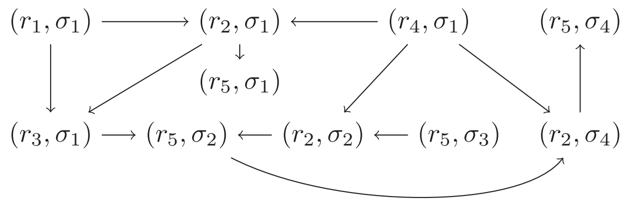
Our approach is inspired by (Romero et al. 2016), which employs datalog reasoning to extract a module of an ontology. It first transforms an ontology Π into a datalog program Π^d as follows: each rule $r \in \Pi$ is transformed into a datalog rule r^d by replacing existential variables $z \in Z_r$ with fresh constants c_z. It also uses a dataset D^* based on a fresh constant $*$ and a signature Γ such that $D^* = \{\, \mathsf{A}(*, \ldots, *) \mid \mathsf{A} \in \Gamma \,\}$. It then applies the rules in Π^d (through forward chaining) on D^* and keeps track of the rule application. The module Π' consists of all the rules r from Π such that r^d is involved in the derivation of some facts over Γ. It is shown that $\Pi \cup D \models Q$ iff $\Pi' \cup D \models Q$ for each BCQ Q and dataset D over Γ (Romero et al. 2016). Clearly, by taking $\Gamma = \mathsf{sig}(\Pi) \setminus \Sigma$, forgetting about Σ in Π can be computed via unfolding Π' instead of Π.

Moreover, we define the GRA of Π w.r.t. Σ based on the forward chaining of Π^d on D^*, denoted $I_0, \ldots, I_n$[3]. We say a pair (r_1, σ_1) triggers a (not necessarily different) pair (r_2, σ_2) if (i) r_2 depends on r_1, (ii) r_1^d is applicable to some I_i ($0 \leq i < n$) with homomorphism σ_1, and (iii) r_2^d is applicable to I_j for some $i < j \leq n$ with a homomorphism σ_2 such that $\mathsf{body}(r_2^d)\sigma_2 \cap \mathsf{head}(r_1^d)\sigma_1 \neq \emptyset$. The *graph of rule application (GRA)* of Π w.r.t. Σ is defined as a directed graph $G_\Pi^\Sigma = (N, E)$ with the nodes $N = \{\, (r, \sigma) \mid r \in \Pi, r^d$ is applicable to some I_i with homomorphism $\sigma \,\}$, and the edges $E = \{\, ((r, \sigma), (r', \sigma')) \mid (r, \sigma)$ triggers $(r', \sigma') \}$. That is, a node of the GRA is a rule in Π coupled its instantiation that is involved in the forward chaining, and the edges in the GRA capture the triggering relationships.

Example 3. *Let Π_e in Example 1 be extended with one more rule $r_5 = \mathsf{E}(y, z) \to \mathsf{C}(y, z)$. Then, Π_e^d can be obtained by replacing r_1 and r_2 with the following two datalog rules*

$$r_1^d = \mathsf{A}(x) \to \mathsf{B}(x, c_1) \wedge \mathsf{C}(x, c_1),$$
$$r_2^d = \mathsf{C}(x, y) \wedge \mathsf{D}(x) \to \mathsf{E}(y, c_2).$$

Take $\Sigma = \{\mathsf{B}, \mathsf{D}\}$ and thus $\Gamma = \{\mathsf{A}, \mathsf{C}, \mathsf{E}, \mathsf{F}\}$. Then, $D^* = \{\mathsf{A}(*), \mathsf{C}(*, *), \mathsf{E}(*, *), \mathsf{F}(*)\}$. New facts derived in the forward chaining of Π_e^d on D^* include $\mathsf{B}(*, c_1)$, $\mathsf{C}(*, c_1)$, $\mathsf{D}(*)$, $\mathsf{E}(c_1, c_2)$, $\mathsf{E}(*, c_2)$, $\mathsf{C}(c_1, c_2)$, $\mathsf{C}(*, c_2)$, $\mathsf{E}(c_2, c_2)$, and $\mathsf{C}(c_2, c_2)$. Let $\sigma_1 = \{x/*, y/c_1, z/c_2\}$, $\sigma_2 = \{x/*, y/*, z/c_2\}$, $\sigma_3 = \{x/*, y/*, z/*\}$, and $\sigma_4 = \{x/*, y/c_2, z/c_2\}$. The GRA of Π_e w.r.t. Σ is as follows.



$$\begin{tikzcd}
(r_1,\sigma_1) \arrow[r] \arrow[dd] & (r_2,\sigma_1) \arrow[d] \arrow[ddl] & & (r_4,\sigma_1) \arrow[ll] \arrow[ddl] \arrow[ddr] & (r_5,\sigma_4) \\
& (r_5,\sigma_1) & & & \\
(r_3,\sigma_1) \arrow[r] & (r_5,\sigma_2) \arrow[rrr, bend right] & (r_2,\sigma_2) \arrow[l] & (r_5,\sigma_3) \arrow[l] & (r_2,\sigma_4) \arrow[uu]
\end{tikzcd}$$

Figure 1: The GRA of Π_e w.r.t. $\Sigma = \{\mathsf{B}, \mathsf{D}\}$.

A GRA is always finite. If Π has n rules and each rule has at most m occurrences of variables, then the number of nodes (i.e., rule instantiations) in G_Π^Σ is at most $n(nm+1)^m$.

We show that a GRA captures all relevant unfolding sequences. Define an *unravelling tree* of a GRA G to be a pair (T, λ) where T is a tree and λ is a mapping from each node in T to a node in G such that n has child n' in T iff there is an edge from $\lambda(n')$ to $\lambda(n)$ in G.

Proposition 6. *For an ontology Π and a signature Σ, if $r \in \mathsf{unfold}(\Pi)|_{\overline{\Sigma}}$ is obtained from the result of an unfolding sequence $r_1\mu_1 r_2 \mu_2 \ldots \mu_{n-1} r_n$ (by eliminating head atoms), then there is an unravelling tree of G_Π^Σ whose nodes are labelled with $(r_1, \sigma_1), \ldots, (r_n, \sigma_n)$ for some substitutions $\sigma_1, \ldots, \sigma_n$ and whose root is (r_1, σ_1).*

[3]We do not assume classical chase termination here. Instead, the forward chaining terminates only when the GRA is stable.

Proof Sketch: We only need to show that for each r_i ($1 \leq i \leq n$), there is a node (r_i, σ_i) in G_Π^Σ (for some σ_i), and if $i > 1$, there are a node (r_j, σ_j) with $1 \leq j < i$ (and some σ_j) and an edge from (r_i, σ_i) to (r_j, σ_j) in G_Π^Σ.

We first show by induction that in the forward chaining $I_0, \dots, I_m$ of Π^d on D^*, each r_i^d is applicable to some I_l ($0 \leq l \leq m$) with some homomorphism σ_i. Clearly, r_n^d is applicable to D^* with the homomorphism σ_n mapping all the variables to the constant $*$. For $1 \leq k < n$, suppose the claim holds for all r_i^d with $i > k$, we show it for $i = k$. By the definition of unfolding, each atom $\mathbf{a} \in \mathsf{body}(r_k^d)$ either remains in the body of r (under variable renaming) or unifies with a head atom $\mathbf{a}'$ in some rule r_j with $k < j \leq n$. In the former case, $\mathbf{a}$ maps to D^*; and in the latter case, by the induction hypothesis, $\mathbf{a}'\sigma_j$ exists in some I_{l_j} for some σ_j. Let l be the maximum l_j, then r_k^d is applicable to I_l.

We have shown that (r_i, σ_i) is in G_Π^Σ for each $1 \leq i \leq n$. If $i > 1$, we want to show that there are a node (r_j, σ_j) with $1 \leq j < i$ (and some σ_j) and an edge from (r_i, σ_i) to (r_j, σ_j) in G_Π^Σ. In the unfolding sequence, r_i unfolds the result of $r_1\mu_1 \dots r_{i-1}$, and thus some head atoms of r_i unifies with some body atoms from $r_1, \dots, r_{i-1}$. Suppose $H \subseteq \mathsf{head}(r_i)$ unifies with $B \subseteq \mathsf{body}(r_j)$ with $1 \leq j < i$. From the above discussion, r_i^d and r_j^d are applicable to respectively some I_{l_i} and I_{l_j} with homomorphisms σ_i and σ_j s.t. $l_i < l_j$. Also, $\mathsf{body}(r_j^d)\sigma_j \cap \mathsf{head}(r_i^d)\sigma_i = H\sigma_i \neq \emptyset$. Taking $D = \mathsf{body}(r_i^d)\sigma_i \cup (\mathsf{body}(r_j^d)\sigma_j \setminus \mathsf{head}(r_i^d)\sigma_i)$, it is clear that r_j depends on r_i. By the definition of rule triggering, (r_i, σ_i) connects to (r_j, σ_j) in G_Π^Σ. $\square$

For ontology Π_e in Example 3, an unfolding sequence is $r_3\mu r_2\mu' r_1\mu r_4$ with $\mu = \emptyset$ and $\mu' = \{x'/x\}$. A correspondent unravelling tree has four nodes n_1, n_2, n_3 and n_4, labelled with respectively (r_3, σ_1), (r_1, σ_1), (r_2, σ_1) and (r_4, σ_1), where n_1 is the root and has two children n_2 and n_3, and n_4 is a child of n_3.

Now, we present an algorithm (Algorithm 1) to compute forgetting via unfolding guided by the GRA. For a set N of nodes in the GRA G and a predicate A, let $\mathsf{pre}_\mathsf{A}(N)$ consist of all the nodes (r, σ) in G such that there are a node $(r', \sigma') \in N$ and an edge from (r, σ) to (r', σ') in G with A occurring in $\mathsf{head}(r)\sigma \cap \mathsf{body}(r')\sigma'$.

In the following example, we explain the execution of Algorithm 1 on the GRA in Example 3.

Example 4. *We focus on the case when r_3 is popped from R at line 3. Then, $A_{r_3} = \{\mathsf{B}(x, y)\}$ and $N_{r_3} = \{(r_3, \sigma_1)\}$. At line 7, $\mathsf{B}(x, y)$ is popped, and $\mathsf{pre}_\mathsf{B}(N_{r_3})$ consists of only (r_1, σ_1). Yet r_3 is not unfoldable by r_1. At line 14, $\mathsf{E}(y, z)$ is pushed into A_{r_3}. Again, at line 7, $\mathsf{E}(y, z)$ is popped, and $\mathsf{pre}_\mathsf{E}(N_{r_3})$ consists of only (r_2, σ_1). r_3 is unfoldable by r_2, which results in a rule $r = \mathsf{B}(x', y) \wedge \mathsf{C}(x, y) \wedge \mathsf{D}(x) \to \exists z.[\mathsf{E}(y, z) \wedge \mathsf{E}(x', z)]$ pushed into R, with $N_r = \{(r_3, \sigma_1), (r_2, \sigma_1)\}$ and $A_r = \{\mathsf{B}(x', y), \mathsf{D}(x)\}$.*

In the next iteration, at line 3, r is popped, and at line 7, $\mathsf{B}(x', y)$ is popped with $\mathsf{pre}_\mathsf{B}(N_r)$ consisting of only (r_1, σ_1). This time, r is unfoldable by r_1, which result in a rule $r' = \mathsf{A}(x) \wedge \mathsf{D}(x) \to \exists y, z.[\mathsf{B}(x, y) \wedge \mathsf{C}(x, y) \wedge \mathsf{E}(y, z) \wedge \mathsf{E}(x, z)]$ pushed into R, with $N_{r'} = \{(r_3, \sigma_1), (r_2, \sigma_1), (r_1, \sigma_1)\}$ and $A_{r'} = \{\mathsf{D}(x)\}$.

In a third iteration, at line 3, r' is popped, and at line 7, $\mathsf{D}(x)$ is popped, and $\mathsf{pre}_\mathsf{D}(N_{r'})$ consists of only (r_4, σ_1). r' is unfoldable by r_4, which result in a rule $r'' = \mathsf{A}(x) \wedge \mathsf{F}(x) \to \exists y, z.[\mathsf{B}(x, y) \wedge \mathsf{C}(x, y) \wedge \mathsf{E}(y, z) \wedge \mathsf{E}(x, z)]$ to be pushed into R. Finally, $\mathsf{A}(x) \wedge \mathsf{F}(x) \to \exists y, z.[\mathsf{C}(x, y) \wedge \mathsf{E}(y, z) \wedge \mathsf{E}(x, z)]$ is added into Π' at line 5.

Algorithm 1 Compute forgetting via GRA-guided unfolding

```
Input: an ontology $\Pi$ and a signature $\Sigma$
Output: an ontology $\Pi'$ as $\mathsf{forget}(\Pi, \Sigma)$
 1: initiate $\Pi' := \emptyset$, a stack $R$ containing the rules in $\Pi$,
    and for each $r \in \Pi$, a stack $A_r$ containing $\mathsf{body}(r)|_\Sigma$
    and a set $N_r$ consisting of all the nodes $(r, \sigma)$ in $G_\Pi^\Sigma$
 2: while $R$ is not empty do
 3:     pop a rule $r$ from $R$
 4:     if $\mathsf{body}(r)|_\Sigma = \emptyset$ and $\mathsf{head}(r)|_{\overline{\Sigma}} \neq \emptyset$ then
 5:         add $\mathsf{body}(r) \to \mathsf{head}(r)|_{\overline{\Sigma}}$ to $\Pi'$
 6:     while $A_r$ is not empty do
 7:         pop an atom $\mathbf{a}$ from $A_r$, with its predicate $\mathsf{A}$
 8:         for each node $(r', \sigma') \in \mathsf{pre}_\mathsf{A}(N_r)$ do
 9:             if $r$ is unfoldable by $r'$ with some $\mu$ then
10:                 push $r'' = r \bowtie_\mu r'$ into $R$
11:                 assign $N_{r''} := N_r \cup \{(r', \sigma')\}$
12:                 let $A_{r''}$ contain the atoms in $\mathsf{body}(r'')|_\Sigma$
13:             else
14:                 push atoms in $\mathsf{body}(r) \setminus \{\mathbf{a}\}$ to $A_r$
15: return $\Pi'$
```

For a signature Σ, we say an ontology Π is Σ-*aGRA* if G_Π^Σ is acyclic. The class of Σ-aGRA ontologies strictly subsumes the class of aGRD ontologies, as it is easy to see that for an aGRD ontology, its GRA is also acyclic; and on the other hand, some Σ-aGRA ontologies are not aGRD. The ontology Π_e in Example 3 is such a case.

The following result states that our algorithm terminates on Σ-aGRA ontologies and computes a result of forgetting.

Theorem 3. *For a signature Σ and an ontology Π satisfying Σ-aGRA, Algorithm 1 terminates and computes a result of forgetting about Σ in Π.*

The following corollary extends the known class (i.e., aGRD) of ontologies whose forgetting is expressible.

Corollary 1. *For a signature Σ, a result of forgetting about Σ in a Σ-aGRA ontology is expressible.*

Conclusion and Future Work

In this paper, we present to the best of our knowledge, the first study on forgetting for existential rules. We have defined forgetting and unfolding for existential rules, used unfolding to characterise forgetting, established expressibility results, and developed an algorithm for forgetting.

There are several interesting open problems. First, through the connection to boundedness, it is possible to identify fragments of existential rules for which the expressibility of forgetting is decidable. Candidate fragments include (frontier-) guarded existential rules and those satisfy certain acyclicity conditions. Other related problems include the computational complexity of deciding the expressibility and a bound on the size of a result of forgetting. Also, we hope to implement our algorithm and evaluate it over practical ontologies. Furthermore, inseparability and conservative extension (Botoeva et al. 2016) are closely related to forgetting, which are less studied for existential rules. Finally, we are working on extending forgetting for logic programming (Zhang and Foo 2006; Eiter and Wang 2008; Zhang and Zhou 2009; Ji, You, and Wang 2015; Gonçalves, Knorr, and Leite 2016) to existential rules (with negation).

Acknowledgments

This work was supported by Australian Research Council (ARC) under grant DP130102302, National Key Research and Development Program of China (2017YFC0908400) and National Natural Science Foundation of China (61502336).

References

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

Botoeva, E.; Konev, B.; Lutz, C.; Ryzhikov, V.; Wolter, F.; and Zakharyaschev, M. 2016. Inseparability and conservative extensions of description logic ontologies: A survey. In *Lecture Notes of the 12th International Summer School on Reasoning Web*, 27–89.

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

Cosmadakis, S. S.; Gaifman, H.; Kanellakis, P. C.; and Vardi, M. Y. 1988. Decidable optimization problems for database logic programs (preliminary report). In *Proc. of the 20th Annual ACM Symposium on Theory of Computing*, 477–490.

Eiter, T., and Wang, K. 2008. Semantic forgetting in answer set programming. *Artif. Intell.* 172(14):1644–1672.

Gaifman, H.; Mairson, H. G.; Sagiv, Y.; and Vardi, M. Y. 1993. Undecidable optimization problems for database logic programs. *J. ACM* 40(3):683–713.

Gonçalves, R.; Knorr, M.; and Leite, J. 2016. The ultimate guide to forgetting in answer set programming. In *Proc. of the 15th KR*, 135–144.

Gottlob, G.; Orsi, G.; and Pieris, A. 2014. Query rewriting and optimization for ontological databases. *ACM Trans. Database Syst.* 39(3):25:1–25:46.

Grau, B. C.; Horrocks, I.; Krötzsch, M.; Kupke, C.; Magka, D.; Motik, B.; and Wang, Z. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res.* 47:741–808.

Ji, J.; You, J.; and Wang, Y. 2015. On forgetting postulates in answer set programming. In *Proc. of the 24th IJCAI*, 3076–3083.

Konev, B.; Walther, D.; and Wolter, F. 2009. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proc. of the 21st IJCAI*, 830–835.

König, M.; Leclère, M.; Mugnier, M.; and Thomazo, M. 2015. Sound, complete and minimal UCQ-rewriting for existential rules. *Semantic Web* 6(5):451–475.

Koopmann, P., and Schmidt, R. A. 2014. Count and forget: Uniform interpolation of $\mathcal{SHQ}$-ontologies. In *Proc. of the 7th IJCAR*, 434–448.

Leclère, M.; Mugnier, M.; and Ulliana, F. 2016. On bounded positive existential rules. In *Proc. of the 29th International Workshop on Description Logics*.

Lin, F., and Reiter, R. 1994. Forget it. In *Proc. of the AAAI Fall Symposium on Relevance*, 154–159.

Ludwig, M., and Konev, B. 2014. Practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes with applications to logical difference. In *Proc. of the 14th KR*.

Lutz, C., and Wolter, F. 2011. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proc. of the 22nd IJCAI*, 989–995.

Lutz, C.; Seylan, I.; and Wolter, F. 2012. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proc. of the 13th KR*.

Nikitina, N., and Rudolph, S. 2014. (Non-)Succinctness of uniform interpolants of general terminologies in the description logic $\mathcal{EL}$. *Artif. Intell.* 215:120–140.

Romero, A. A.; Kaminski, M.; Grau, B. C.; and Horrocks, I. 2016. Module extraction in expressive ontology languages via datalog reasoning. *J. Artif. Intell. Res.* 55:499–564.

Wang, Z.; Wang, K.; Topor, R. W.; and Pan, J. Z. 2008. Forgetting concepts in DL-Lite. In *Proc. of 5th ESWC*, 245–257.

Wang, Z.; Wang, K.; Topor, R. W.; and Pan, J. Z. 2010. Forgetting for knowledge bases in DL-Lite. *Ann. Math. Artif. Intell.* 58(1-2):117–151.

Wang, K.; Wang, Z.; Topor, R. W.; Pan, J. Z.; and Antoniou, G. 2014. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Computational Intelligence* 30(2):205–232.

Wang, Z.; Wang, K.; and Topor, R. W. 2015. DL-Lite ontology revision based on an alternative semantic characterization. *ACM Trans. Comput. Log.* 16(4):31:1–31:37.

Zhang, Y., and Foo, N. Y. 2006. Solving logic program conflict through strong and weak forgettings. *Artif. Intell.* 170(8-9):739–778.

Zhang, Y., and Zhou, Y. 2009. Knowledge forgetting: Properties and applications. *Artif. Intell.* 173(16-17):1525–1537.

Zhao, Y., and Schmidt, R. A. 2016. Forgetting concept and role symbols in $\mathcal{ALCOIH}^{+}_{\mu}(\nabla, \sqcap)$-ontologies. In *Proc. of the 25th IJCAI*, 1345–1353.

Zhou, Y. 2015. First-order disjunctive logic programming vs normal logic programming. In *Proc. of the 24th IJCAI*, 3292–3298.