Forgetting and Unfolding for Existential Rules

Zhe Wang and 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 \( ^{+} \) 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 \( \Sigma \) of non-logical symbols (deemed to be irrelevant or private) from a knowledge base \( \Pi \) to obtain a new knowledge base \( \Pi' \) that contains no occurrence of the symbols from \( \Sigma \) and preserves all relevant logical consequences of \( \Pi \) 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, 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., \( \Pi' \) entails the same concept inclusions outside \( \Sigma \) as \( \Pi \) 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 \( \Pi' \) and \( \Pi \) have the same interpretation on symbols outside \( \Sigma \) (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 \( \Pi' \) to preserve all second-order entailment of \( \Pi \) (outside \( \Sigma \)) (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.
We develop an algorithm of unfolding, which actually
2. To provide a syntactic characterisation for the notion of
1. Based on query answering, we introduce a definition of
φ

We assume standard first-order logic notions, such as predi-
tion of forgetting. Our contribution to forgetting in existen-
tial rules in this paper can be summarised as follows.

1. There exists a substitution \( \sigma \) such that the forgetting can be captured in existential rules.
2. Unfolding is a standard and important
3. To establish a connection between the expressibility of forgetting and a variant of boundedness, which
4. We develop an algorithm of unfolding, which actually

1. We tackle the above open problem by estab-
2. For a set \( T \) of terms, \( \sigma|_T \) denotes the substitution \( t/t' \in \sigma \mid t \in T \).
3. 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 (Leclègre, 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 \( \emptyset \in I' \subseteq I \), and \( \tau \) is a most general unifier satisfying condition \( C \) if for each unifier \( \tau' \) between \( I \) and \( I' \) satisfying \( C \), there exists a substitution \( \sigma \) such that \( \tau' = \tau \sigma \).

An existential rule (or a rule) \( r \) is a formula of the form
\[
\forall \vec{x}, \forall \vec{y}, \forall \vec{z}. \phi(\vec{x}, \vec{y}) \rightarrow \exists \vec{x}, \psi(\vec{x}, \vec{z})
\]
where \( \vec{x}, \vec{y}, \vec{z} \) 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 \( \mathbb{R}, \mathbb{R} \) and \( \mathbb{R}, \mathbb{R} \). Formula \( \phi \) is the body of the rule \( r \), denoted \( \text{body}(r) \), and formula \( \psi \) is the head of \( r \), denoted \( \text{head}(r) \); 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 \( \text{body}(r) \rightarrow \text{head}(r) \). 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. An ontology is a finite set of rules. For simplicity, we disallow trivial rules, i.e., \( \text{body}(r) = \text{head}(r) \) 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 \( \sigma \) from \( \text{body}(r) \) to \( I \), and the result of applying \( r \) to \( I \) with \( \sigma \) is \( r^*_\sigma(I) = I \cup \text{head}(r)\sigma' \), where \( \sigma' \) 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 \( I \) on \( I \) is a sequence of instances \( I_i \) \((i \geq 0)\), where \( I_0 = I \), \( I_{i+1} = \bigcup_{r \in \Pi, \sigma} r^*_\sigma(I_i) \). For \( k \geq 0 \), let \( \Pi^k(I) = I_k \) and \( \Pi(I) = \bigcup_{i=0}^{\infty} I_i \).

We assume standard first-order logic notions, such as predic-
cates, constants, variables, terms, (ground) atoms, formula

equivalence (\( \equiv \)). A signature is a set of

A signature is a set of predicates; for a formula \( \phi \), \( \text{sig}(\phi) \) denotes the signature of \( \phi \), 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 \( I \), \( \text{var}(I) \) 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 \( \sigma \), we write \( t\sigma = t \); otherwise if \( t/t' \in \sigma \) and \( t\sigma = t' \), it actually extends to (sets of) atoms and formulas. A homomorphism from an instance \( I \) to an instance \( I' \) is a substitution \( \sigma \) 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 \( \tau \) such that \( I\tau = I'\tau \); and \( \tau \) is a most general unifier satisfying condition \( C \) if for each unifier \( \tau' \) between \( I \) and \( I' \) satisfying \( C \), there exists a substitution \( \sigma \) such that \( \tau' = \tau \sigma \).

1. This is not reflected in our examples for simplicity.
is \( \tau_i^r(I) = (I \setminus I') \cup \text{body}(r) \tau' \) where \( \tau' \) is a safe extension of \( \tau|_{\text{var}(H)} \) on \( Y_i \cup (X_i \setminus \text{var}(H)) \). For a set \( S \) of instances and a set \( \Pi \) of rules, the backward chaining, a.k.a. rewriting, of \( \Pi \) on \( S \) is a sequence of sets of instances \( S_i \) \((i \geq 0)\), where \( S_0 = S \) and \( S_{i+1} = S_i \cup \{ \tau_i^r(I) \mid r \in \Pi, I_i \in S_i \} \). For \( k \geq 0 \), let \( \Pi_k^i(S) = S_k \), and \( \Pi_0^i(S) = \bigcup_{i=0}^{\infty} S_i \).

Ontology-based query answering (König et al. 2015; Gottlob, Orsi, and Pieris 2014) concerns essentially the reasoning problem of deciding for an ontology \( \Pi \), a dataset \( D \) and a BCQ \( Q \), whether \( \Pi \cup D \vDash Q \). Forward chaining and backward chaining for existential rules are both sound and complete for query answering. In particular, it holds that \( \Pi \cup D \vDash Q \) if and only if \( \Pi_k^i(D) \vDash Q \) for some \( k \geq 0 \), and if \( D \vDash \Pi_k^i(Q) \) for some \( l \geq 0 \) (Leclère, Mugnier, and Uliliana 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 \( \Sigma \) in an ontology \( \Pi \) is a logically weaker ontology \( \Pi' \) not containing any occurrence of predicates from \( \Sigma \), such that \( \Pi' \) preserves query answering of \( \Pi \) for any BCQ and any dataset outside \( \Sigma \).

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

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

**Example 1.** Consider an ontology \( \Pi_e \) consisting of the following rules

\[
\begin{align*}
r_1 &= A(x) \rightarrow \exists y, [B(x, y) \land C(x, y)], \\
r_2 &= C(x, y) \land D(x) \rightarrow \exists z, E(y, z), \\
r_3 &= B(x, y) \land (E(y, z) \rightarrow E(x, z)), \\
r_4 &= F(x) \rightarrow D(x).
\end{align*}
\]

A result of forgetting about \( \Sigma = \{B, D\} \) in \( \Pi_e \) consists of the following three rules:

\[
\begin{align*}
A(x) &\rightarrow \exists y, C(x, y), \\
C(x, y) \land F(x) &\rightarrow \exists z, E(y, z), \\
A(x) &\land F(x) \rightarrow \exists y, z, [C(x, y) \land E(y, z) \land E(x, z)].
\end{align*}
\]

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

**Proposition 1.** If \( \Pi' \) and \( \Pi'' \) are both results of forgetting about signature \( \Sigma \) in ontology \( \Pi \) then \( \Pi' \equiv \Pi''. \)

To prove this proposition, we need a lemma.

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

**Proof Sketch for Proposition 1:** For each rule \( r \in \Pi' \setminus \Pi'', \) let \( \sigma \) be as in Lemma 1. Clearly, \( \Pi' \cup \text{body}(r) \sigma \models \text{head}(r) \sigma \).

Since \( \sig(r) \subseteq \sig(\Pi') \setminus \sig(\Sigma) \), by the definition of forgetting, \( \Pi \cup \text{body}(r) \sigma \models \text{head}(r) \sigma \), and also, \( \Pi'' \cup \text{body}(r) \sigma \models \text{head}(r) \sigma \). By Lemma 1, \( \Pi'' \models r \); that is, \( \Pi'' \equiv \Pi' \). 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 \( \mu = (B, H, r) \) of \( \text{body}(r) \) and \( r' \). Here \( r \) and \( r' \) are not necessarily different.

The result of unfolding \( r \) by \( r' \) with \( \mu \) is the following rule denoted \( r \underset{r'}{\triangleright} \mu \):

\[
(b \text{ody}(r) \setminus B) \bigcup \text{body}(r') \rightarrow \exists \exists \bar{z}, [\text{head}(r) \tau'' \land \text{head}(r') \tau']
\]

where \( \tau' \) is a safe extension of \( \tau|_{\text{var}(H)} \) on \( \text{var}(r') \setminus \text{var}(H) \). \( \tau'' \) is a safe extension of \( \tau|_{\Sigma} \) on \( \Sigma \), and \( \bar{z} \) consists of all the variables in the head but not in the body.

The unfolding chaining on a set of rules \( \Pi \) is a sequence of rule sets \( \Pi^\prime \) \((i \geq 0)\), where \( \Pi^{i+1} = \Pi^i \cup \{ r \triangleright r' | r, r' \in \Pi^i \} \). The unfolding of \( \Pi \) is 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 \( \text{head}(r') \) after unfolding, in order to correctly handle the existential quantifiers in the rule heads.

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

\[
 r = B(x', y) \land C(x, y) \land D(x) \rightarrow \exists z, [E(y, z) \land E(x', z)].
\]

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

\[
 A(x) \land D(x) \rightarrow \exists y, z, [B(x, y) \land C(x, y) \land E(y, z) \land E(x, z)].
\]

Note that \( \mu'' = \{ (B(x', y)), (B(x', y)), \{ x'/x \} \} \) is not a piece unification of \( \text{body}(r) \) and \( r_3 \), as \( y \) is not an exclusive variable for \( \{ B(x, y) \} \) in \( \text{body}(r) \).

Let \( \Pi \) be a set of rules of the form \( B_i \rightarrow H_i \) \((1 \leq i \leq n)\), and we recall that distinct rules have disjoint sets of variables. An aggregated rule of \( \Pi \) is of the form \( B_1 \rightarrow H_1 \), \( \ldots \), \( B_k \rightarrow H_k \) 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 \( \Pi \) then each copy has a distinct set of variables (Leclère, Mugnier, and Uliliana 2016). The following result shows the soundness and completeness of unfolding regarding ontological reasoning.

**Proposition 2.** Let \( \Pi \) be an ontology, \( r \) a rule, \( D \) a dataset and \( Q \) a BCQ. Then

\[(I) \quad \Pi \models r \text{ for } r \in \text{unfold}(\Pi) .\]
(2) \( \Pi \cup D \models Q \) iff there are \( k \geq 0 \) and an aggregated rule \( r \) of \( \Pi^k \) 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 \( \Pi^k \).

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 \( D \models \Pi^l(\{ Q \}) \) for some \( l \geq 0 \). If \( Q' \in \Pi^l(\{ Q \}) \) is mapped to \( D \) with homomorphism \( \sigma' \), then by the definition of \( \Pi^l(\{ Q \}) \), there are \( 0 \leq n \leq l \) and a sequence of BCQs \( Q_i \) (\( 0 \leq i \leq n \)), such that \( Q_0 = Q \), \( Q_{i+1} = r_i(\sigma_i(Q_i)) \) for some \( r_i \in \Pi \) and piece unification \( \mu_i \), 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 \( \mu_i' \). Let \( k = n - m \), we have an aggregated rule \( r \) by combining all the rules in \( \Pi^k \) that are obtained by unfolding \( r_0, \ldots, r_m \) by \( r_{m+1}, \ldots, r_n \). It is not hard to see that body(\( r \)) can be mapped to \( Q' \) with some homomorphism \( r \). Taking \( \sigma = \sigma' \), we have \( r(\sigma)(D) \models Q \).

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

For an instance \( I \) and a signature \( \Sigma \), \( I(\Sigma) \) consists of the atoms in \( I \) that are over \( \Sigma \). Let

\[
\Pi(\Sigma) = \{ \text{body}(r) \rightarrow \text{head}(r) \mid r \in \Pi, \text{head}(r) \not\in \emptyset, \text{and body}(r)(\Sigma) = \text{body}(r) \}
\]

and \( \Pi(\Sigma) = \Pi(\Sigma) \cup (\Sigma) \).

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

**Theorem 1.** For an ontology \( \Pi \) and a signature \( \Sigma \), a result of forgetting about \( \Sigma \) in \( \Pi \) is expressible iff unfold(\( \Pi(\Sigma) \)) is so. In this case, forget(\( \Pi(\Sigma) \)) \( \equiv \) unfold(\( \Pi(\Sigma) \)).

**Proof Sketch:** Suppose unfold(\( \Pi(\Sigma) \)) can be expressed as an ontology \( \Pi' \), assume sig(\( \Pi' \)) \( \subseteq \) sig(\( \Pi \) \( \setminus \) \( \Sigma \)). By Proposition 2 (1), \( \Pi \models \Pi' \). For each dataset \( D \) and each BCQ \( Q \) s.t. \( \text{sig}(D \cup Q) \cap \Sigma = \emptyset \), by Proposition 2 (2), \( \Pi \cup D \models Q \) implies that there is an aggregated rule \( r \) of \( \Pi(\Sigma) \) s.t. \( r \) is applicable to \( D \) with homomorphism \( \sigma \) and \( r(\sigma)(D) \models Q \).

If \( r \) is obtained by aggregating \( r_0, \ldots, r_n \) in unfold(\( \Pi(\Sigma) \)), then since \( r \) is applicable to \( D \), \( \text{body}(r_i) \) is over sig(\( \Pi \) \( \setminus \) \( \Sigma \)) for each \( 0 \leq i \leq n \). Then, \( r'_i = \text{body}(r_i) \rightarrow \text{head}(r_i) \) if sig(\( \Pi(\Sigma) \)) \( \subseteq \) sig(\( \Pi(\Sigma) \)).

For each \( 0 \leq i \leq n \), let \( r'_i = \text{the aggregated rule obtained by combining all the } r'_i \) (\( 0 \leq i \leq n \)). As \( r(\sigma')(D) \models Q \) and sig(\( Q \) \( \cap \Sigma = \emptyset \), \( r(\sigma')(D) \models Q \). That is, \( \Pi(\Sigma) \models Q \).

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 \( \Pi \) is predicate bounded w.r.t. signatures \( \Sigma_d \) and \( \Sigma_q \) if there exists \( k \geq 0 \), for each dataset \( D \) over \( \Sigma_d \) and each BCQ \( Q \) over \( \Sigma_q \) with \( \Pi \cup D \models Q \), the following condition is satisfied:

For each instance \( I \) and each minimal subset \( I \subseteq I_q \) 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. sig(\( I' \)) \( \subseteq \) sig(\( I \)) and \( \Pi^k(I') \models I \).

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

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

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

**Lemma 3.** Let $D$ be a dataset, $Q$ be a BCQ and $\Pi$ 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 $\Pi^k$ s.t. $\{r\} \cup D \models Q$ then $\Pi^k |_{\Sigma} (D) \models Q$.
2. For any $l \geq 0$, if $\Pi^l |_{\Sigma} (D) \models Q$ then there is an aggregated rule $r$ of $\Pi^m$ s.t. $\{r\} \cup D \models Q$.

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

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

Consider the derivation forest $F$ for $Q$ where each node is a pair of the form $(f, i)$ 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 $\sigma$ 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 $r \sigma$ if $f \in \text{head}(r) \sigma$, $g \in \text{body}(r) \sigma$ and $j < i$. We only need to show that $\Pi^k |_{\Sigma} \cup D \models f$ for each $f$ occurring in the derivation forest.

Consider a node $(f, i)$ with the largest $i$ that has a child over $\Sigma$, 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 $\text{sig}(\Pi) \setminus \Sigma$. That is, to show $\Pi \models f$, we only need to show $\Pi^k |_{\Sigma} \cup D \models g$, where $g$ is obtained from $f$ by head atom $r \sigma$ with $r \in \Pi$ and $f \in \text{body}(r) \sigma$. By Lemma 2, we only need to show $\Pi^k |_{\Sigma} \cup D \models f$. Thus, $(f, i)$ can be our starting point. Since $\Pi \cup D \models f$ and $f$ is clearly over $\Sigma$ from the assumption that $\Pi$ is predicate bounded w.r.t. $\Sigma_d$ and $\Sigma_q$, there is $l \geq 0$ (independent of $D$ and $f$) and (by taking $I' = \text{body}(r \sigma)$) an instance $I' \subseteq I_l$ for some $j < i$ satisfying $\text{sig}(I') \subseteq \Sigma_d$ and $\Pi^l |_{\Sigma} (I') \models \text{body}(r \sigma)$. Since $\Pi^l |_{\Sigma} (I') \models f$, by Lemma 3 (2), let $k = (l + 1)m$, $m$ is

as in Lemma 3 (2)), there is an aggregated rule $r'$ of $\Pi^k$ s.t. $\{r'\} \cup I' \models f$. Since $\text{sig}(I' \cup \{f\}) \cap \Sigma = \emptyset$, $\text{body}(r')$ is over $\text{sig}(\Pi) \setminus \Sigma$. Thus, there is an aggregated rule $r''$ of $\Pi^k |_{\Sigma}$ that can be obtained from $r'$ by eliminating head atoms s.t. $\{r''\} \cup I' \models f$. That is, by Proposition 2 (2), $\Pi^k |_{\Sigma} \cup I' \models f$. To show $\Pi^k |_{\Sigma} \cup D \models f$, we only need to show that $\Pi^k |_{\Sigma} \cup D \models f$. We have reduced the proof to showing $\Pi^k |_{\Sigma} \cup D \models g$ for some nodes $(g, j)$ with $j < i$. By recursively applying the reduction, it finally reduces to $\Pi^k |_{\Sigma} \cup D \models h$ for some nodes $(h, j)$, which clearly holds. ($\Leftarrow$): Again, we consider the derivations forest $F$ for $Q$, and w.l.o.g., we assume all the facts $f \in I$ occur in $Q$. Note that the roots (leaves) of $F$ consist of facts in $Q \sigma'$ (resp., $D$) and thus are all outside $\Sigma$. Take the sub-forest $F'$ of $F$ where

- the roots consist of all the nodes $(f, i)$ outside $\Sigma$ 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 $\Sigma$ 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 $n$ the minimum $j$. We want to show that $m - n \leq k$ for the bound $k$. It is clear that $\Pi \models I' \models Q'$. As $\text{sig}(I' \cup Q') \cap \Sigma = \emptyset$, by the definition of forgetting, forget($\Pi, \Sigma) \cup I' \models Q'$. By Theorem 1, unfold($\Pi |_{\Sigma} \cup D \models Q'$. From the assumption that $\Pi$ is unfolding bounded w.r.t. $\Sigma$, there is $l \geq 0$ (independent of $D$ and $Q$) s.t. $\Pi^l |_{\Sigma} \equiv \text{unfold}(\Pi) |_{\Sigma}$. That is, $\Pi^l |_{\Sigma} \cup I' \models Q'$. Since that every internal node in $F'$ are over $\Sigma$, there is an aggregated rule $r'$ of $\Pi^l |_{\Sigma}$ s.t. $(r') \cup I' \models Q'$ (as otherwise $(\Pi^l |_{\Sigma})(I')$ would be outside $\Sigma$ and have smaller indexes than $Q'$, leading to a contradiction). Thus, there is an aggregated rule $r''$ of $\Pi^k$ 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 $\Pi^k |_{\Sigma} \models Q'$. From the construction of $Q'$, $\Pi^k |_{\Sigma} \models I$.

Further, we show that program boundedness can be reduced to the expressibility of forgetting, thus establishing its undecidability. For an ontology $\Pi$, $\Pi^k$ is obtained from $\Pi$ by adding two rules $A_1(\vec{x}) \to A_2(\vec{x})$ and $A_2(\vec{x}) \to A_3(\vec{x})$ for each predicate $A$ occurring in $\Pi$, where $A_1$ and $A_3$ are fresh predicates and have the same arity as $A$.

**Proposition 3.** An ontology $\Pi$ is program bounded iff a result of forgetting about $\text{sig}(\Pi) \setminus I$ is expressible.

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

**Proposition 4.** For an aGRD ontology $\Pi$ and a signature $\Sigma$, a result of forgetting about $\Sigma$ in $\Pi$ 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
unfolding sequence of an ontology $\Pi$ is inductively defined as follows: each rule $r \in \Pi$ is an unfolding sequence and the result of the sequence is $r$; and if $\alpha$ is an unfolding sequence whose result is $r$ and $r$ is unfoldable by a rule $r' \in \Pi$ with piece unification $\mu$, then $\alpha \mu r'$ is an unfolding sequence, and the result of the sequence is $r \alpha \mu r'$.

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

Proof Sketch: We only need to show that for each rule $r$ as a result of $r_1 \mu_1 (r_2 \phi \mu_2 r_3)$, where $r_1 \in \text{unf}(\Pi)$ $(i = 1, 2, 3)$, $r$ 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 $\mu_j$ $(j = 1, 2)$. Let $r' = r_2 \phi \mu_2 r_3$, then head($r'$) $\subseteq$ (head($r_2$) $\cup$ head($r_3$))$r_2$. Suppose $\mu_j = (B_j, H_j, r_j)$ and $\mu'_j = (B'_j, H'_j, r'_j)$, and for simplicity we use $r_j$ and $r_j'$ to denote their safe extensions in unfolding. $B_1 \subseteq \text{body}(r_1)$ and $H_1 \subseteq \text{head}(r_1)$, and thus $H_1 \subseteq (\text{head}(r_2) \cup \text{head}(r_3))r_2$. If $H_1 \subseteq \text{head}(r_2)r_2$, it is not hard to see that $r$ is a result of $r_1 \mu_1 r_2 \mu_2 r_3$; and if $H_1 \subseteq \text{head}(r_2)r_2$ then $r$ is a result of $r_1 \mu_1 r_3 \mu_2 r_2$. Otherwise, if $B_2$ does not contain a piece of body($r_1$) w.r.t. $r_1$ such that $I$ unifies (by $\mu_1$) with atoms from both head($r_2$) $\cup$ head($r_3$) $\cup$ head($H_1$) $\cap$ head($r_3$), then by taking $H'_1 = H_1 \cap$ (head($r_2$) $\cup$ 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 $Z = Z_r \cap \text{var}(\text{head}(r_2) \cup \text{head}(r_3))$. 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 $Z_{r_2} = \emptyset$ 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, $Z \subseteq \text{var}(\text{head}(r_2) \cup \text{head}(r_3))$ implies that $Z \subseteq \text{var}(B_2 \cup \text{var}(\text{head}(r_3) \cup \text{var}(\text{head}(r_2) \cup \text{var}(\text{head}(r_3) \cup \text{head}(r_3)))$. That is, $Z \subseteq \text{var}(\text{body}(r_2))$. Thus, $I$ is not a piece w.r.t. $r_2$. By taking $H'_2 = H_1 \cap$ head($r_2$) and $H'_2 = (H_1 \cap$ head($r_2$)), $r$ is a result of $r_1 \mu_1 r_2 \mu_2 r_3$.

To compute forgetting, we can generate rules by searching all valid unfolding sequences. However, the search space can be still 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 data-log 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 $\Pi$ into a datalog program $\Pi^d$ 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 $\Gamma$ such that $D^* = \{ A(\ldots, +) \mid A \in \Gamma \}$. It then applies the rules in $\Pi^d$ (through forward chaining) on $D^*$ and keeps track of the rule application. The module $\Pi^d$ consists of all the rules $r$ from $\Pi$ such that $r^d$ is involved in the derivation of some facts over $\Gamma$. It is shown that $\Pi \cup D \models Q$ iff $\Pi^d \cup D \models Q$ for each BCQ $Q$ and dataset $D$ over $\Gamma$ (Romero et al. 2016). Clearly, by taking $\Gamma = \text{sig}(\Pi) \cap \Sigma$, forgetting about $\Sigma$ in $\Pi$ can be computed via unfolding $\Pi^d$ instead of $\Pi$.

Moreover, we define the GRA of $\Pi$ w.r.t. $\Sigma$ based on the forward chaining of $\Pi^d$ on $D^*$, denoted $I_0, \ldots, I_n$. We say a pair $(r_1, r_2)$ triggers a (not necessarily different) pair $(r_3, r_2)$ if (i) $r_2$ depends on $r_1$, (ii) $r_1^d$ is applicable to some $I_i (0 \leq i < n)$ with homomorphism $\sigma_i$, and (iii) $r_3^d$ is applicable to $I_j$ for some $i < j \leq n$ with a homomorphism $\sigma_2$ such that $\text{body}(r_2^d) \sigma_2 \cap \text{head}(r_3^d) \sigma_1 \neq \emptyset$. The graph of rule application (GRA) of $\Pi$ w.r.t. $\Sigma$ is defined as a directed graph $G^\Sigma_{\Pi} = (N, E)$ with the nodes $N = \{(r, \sigma) \mid r \in \Pi, \sigma\}$ applicable to some $I_i$ with homomorphism $\sigma$, and the edges $E = \{(\sigma, \sigma') \mid (r, \sigma) \text{ triggers } (r', \sigma')\}$. That is, a node of the GRA is a rule in $\Pi$ coupled its instantiation that is involved in the forward chaining, and the edges in the GRA capture the triggering relationships.

Example 3. Let $\Pi_1$ be in Example 1 be extended with one more rule $r_5 = E(y, z) \rightarrow C(y, z)$. Then, $\Pi_2$ can be obtained by replacing $r_1$ and $r_2$ with the following two data-log rules

\[
\begin{align*}
& r_1^d = A(x) \rightarrow B(x, c_1) \land C(x, c_1), \\
& r_2^d = C(x, y) \land D(x) \rightarrow E(y, c_2).
\end{align*}
\]

Take $\Sigma = \{B, D\}$ and thus $\Gamma = \{A, C, E, F\}$. Then, $D^* = \{A(\sigma), C(\sigma, \sigma), E(\sigma, \sigma), F(\sigma)\}$. New facts derived in the forward chaining of $\Pi_1^d$ on $D^*$ include $B(\sigma, c_1)$, $C(\sigma, c_1, D(\sigma), E(c_1, c_2), E(c_2, c_2), C(c_1, c_2), C(c_2, c_2)$, $C(c_2, c_2)$, and $C(c_2, c_2)$. Let $\sigma_1 = \{x/y, c_1, z/c_2\}$, $\sigma_2 = \{x/y, z/c_2\}$, $\sigma_3 = \{x/y, z/c_2\}$, and $\sigma_4 = \{x/y, c_2, z/c_2\}$. The GRA of $\Pi_2$ w.r.t. $\Sigma$ is as follows.

![Figure 1: The GRA of $\Pi_2$ w.r.t. $\Sigma = \{B, D\}$](image)

A GRA is always finite. If $\Pi$ has $\Pi$ rules and each rule has at most $m$ occurrences of variables, then the number of nodes (i.e., rule instantiations) in $G^\Sigma_{\Pi}$ 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 $(T, \lambda)$ where $T$ is a tree and $\lambda$ 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 $\Pi$ and a signature $\Sigma$, if $r \in \text{unf}(\Pi) \cap \Sigma$ is obtained from the result of an unfolding sequence $r_1 \mu_1 r_2 \mu_2 \ldots r_{n-1} \mu_n r_n$ (by eliminating head atoms), then there is an unravelling tree of $G^\Sigma_{\Pi}$ 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 $(r_1, \sigma_1)$.

3We 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 $(r_i, \sigma_i)$ in $G_{\Pi}^\Sigma$ (for some $\sigma_i$), and if $i > 1$, there is a node $(r_j, \sigma_j)$ with $1 \leq j < i$ (and some $\sigma_j$) and an edge from $(r_i, \sigma_i)$ to $(r_j, \sigma_j)$ in $G_{\Pi}^\Sigma$.

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

We have shown that $(r_i, \sigma_i)$ is in $G_{\Pi}^\Sigma$ for each $1 \leq i \leq n$. If $i > 1$, we want to show that there are a node $(r_j, \sigma_j)$ with $1 \leq j < i$ (and some $\sigma_j$) and an edge from $(r_i, \sigma_i)$ to $(r_j, \sigma_j)$ in $G_{\Pi}^\Sigma$. In the unfolding sequence, $r_i$ unfolds the result of $r_{j_1} \ldots r_{j_k-1}$, and thus some head atoms of $r_i$ unifies with some body atoms from $r_{j_1} \ldots r_{j_k-1}$. Suppose $H \subseteq \text{head}(r_j)$ unifies with $B \subseteq body(r_i)$ with $1 \leq j < i$. From the above discussion, $r_j^\Sigma$ and $r_i^\Sigma$ are applicable to respectively some $I_{l_i}$ and $I_{l_j}$ with homomorphisms $\sigma_l$ and $\sigma_j$, s.t. $l_i < l_j$. Also, $body(r_j^\Sigma) \cap head(r_i^\Sigma) = H \sigma_l \neq \emptyset$. Taking $D = body(r_j^\Sigma) \cap \text{head}(r_i^\Sigma)$, it is clear that $D$ depends on $r_j$. By the definition of rule triggering, $(r_i, \sigma_i)$ connects to $(r_j, \sigma_j)$ in $G_{\Pi}^\Sigma$.

For ontology $\Pi_\ell$, in Example 3, an unfolding sequence is $r_3 \mu r_4 \mu r_2 \mu r_3 \mu r_4$ with $\mu = \emptyset$ and $\mu' = \{x'/x\}$. A corresponding unravelling tree has four nodes $n_1$, $n_2$, $n_3$, and $n_4$, labelled with respectively $(r_3, \sigma_1)$, $(r_4, \sigma_1)$, $(r_2, \sigma_1)$ and $(r_4, \sigma_1)$, 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 $\text{pre}_{\Sigma}(N)$ consist of all the nodes $(r, \sigma)$ in $G$ such that there are a node $(r', \sigma') \in N$ and an edge from $(r, \sigma)$ to $(r', \sigma')$ in $G$ with $A$ occurring in $\text{head}(r') \cap body(r')$.

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, $A_{r_3} = \{B(x, y)\}$ and $N_{r_3} = \{(r_3, \sigma_1)\}$. At line 7, $B(x, y)$ is popped, and $\text{pre}_{\Sigma}(N_{r_3})$ consists of only $(r_3, \sigma_1)$. Yet $r_3$ is not applicable by $r_4$. At line 14, $E(y, z)$ is pushed into $A_{r_3}$. Again, at line 7, $E(y, z)$ is popped, and $\text{pre}_{\Sigma}(N_{r_3})$ consists of only $(r_2, \sigma_1)$. $r_3$ is not applicable by $r_2$, which results in a rule $r = B(x', y) \land C(x, y) \land D(x) \implies \exists z. [E(x, y) \land E(x', z)]$ pushed into $R$, with $N_r = \{(r_3, \sigma_1), (r_2, \sigma_1)\}$ and $A_r = \{B(x, y), D(x)\}$.

In the next iteration, line 3, $r$ is popped, and at line 7, $B(x, y)$ is popped with $\text{pre}_{\Sigma}(N_r)$ consisting of only $(r_1, \sigma_1)$. This time, $r$ is not applicable by $r_1$ which results in a rule $r' = A(x) \land D(x) \implies \exists y, z. [B(x, y) \land C(x, y) \land E(y, z) \land E(x, z)]$ pushed into $R$, with $N_{r'} = \{(r_3, \sigma_1), (r_2, \sigma_1), (r_1, \sigma_1)\}$ and $A_{r'} = \{\}$.

Algorithm 1 Compute forgetting via GRA-guided unfolding

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

A_{r'} = \{D(x)\}.

In a third iteration, at line 3, $r'$ is popped, and at line 7, $D(x)$ is popped, and $\text{pre}_{\Sigma}(N_{r'})$ consists of only $(r_1, \sigma_1)$. $r'$ is not applicable by $r_4$, which results in a rule $r'' = A(x) \land E(x, z) \implies \exists y, z. [B(x, y) \land C(x, y) \land E(y, z)]$ to be pushed into $R$. Finally, $A(x) \land F(x) \implies \exists y, z. [C(x, y) \land E(y, z)]$ is added into $\Pi'$ at line 5.

For a signature $\Sigma$, we say an ontology $\Pi$ is $\Sigma$-aGRA if $G_{\Pi}^\Sigma$ is acyclic. The class of $\Sigma$-aGRA ontologies strictly expresses 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 $\Sigma$-aGRA ontologies are not aGRD. The ontology $\Pi_\ell$ in Example 3 is such a case.

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

Theorem 3. For a signature $\Sigma$ and an ontology $\Pi$ satisfying $\Sigma$-aGRA, Algorithm 1 terminates and computes a result of forgetting about $\Sigma$ in $\Pi$.

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

Corollary 1. For a signature $\Sigma$, a result of forgetting about $\Sigma$ in a $\Sigma$-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**


