Query Answering with Guarded Existential Rules under Stable Model Semantics

Hai Wan,[1] **Guohui Xiao,**[2*] **Chenglin Wang,**[1] **Xianqiao Liu,**[1] **Junhong Chen,**[1] **Zhe Wang**[3]
[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, P.R.China
[2]KRDB Research Centre, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
[3]School of Information and Communication Technology, Griffith University, Brisbane, QLD 4111, Australia
wanhai@mail.sysu.edu.cn, xiao@inf.unibz.it, {wangchlin, liuxq35, chenjh46}@mail2.sysu.edu.cn, zhe.wang@griffith.edu.au

Abstract

In this paper, we study the problem of query answering with guarded existential rules (also called GNTGDs) under stable model semantics. Our goal is to use existing answer set programming (ASP) solvers. However, ASP solvers handle only finitely-ground logic programs while the program translated from GNTGDs by Skolemization is not in general. To address this challenge, we introduce two novel notions of (1) guarded instantiation forest to describe the instantiation of GNTGDs and (2) prime block to characterize the repeated infinitely-ground program translated from GNTGDs. Using these notions, we prove that the ground termination problem for GNTGDs is decidable. We also devise an algorithm for query answering with GNTGDs using ASP solvers. We have implemented our approach in a prototype system. The evaluation over a set of benchmarks shows encouraging results.

Introduction

Existential rules (Calì, Gottlob, and Pieris 2010; Calì, Gottlob, and Lukasiewicz 2012), also well-known as *tuple-generating dependencies (TGDs)* (Beeri and Vardi 1984), is a powerful rule-based logic formalism for query answering. Since query answering with TGDs is undecidable in general (Beeri and Vardi 1981), one important research direction is to identify decidable fragments of TGDs. A major decidable class of TGDs is *guarded* TGDs *(GTGDs)* whose body contains an atom that covers all body variables (Calì, Gottlob, and Lukasiewicz 2012). Guardedness is a well-accepted paradigm because it captures important databases constraints such as inclusion dependencies, and lightweight description logics such as DL-Lite.

Adding non-monotonic negation in TGDs under the *stable model semantics (SMS)* (Gelfond and Lifschitz 1988), as in answer set programming (ASP), called *normal* TGDs *(NTGDs)*, has drawn much attention. The most notable work is by Gottlob et al. (2014), who showed the decidability and complexity of query answering under SMS for GNTGDs and extended the QCHECK algorithm (Calì, Gottlob, and Kifer 2013) to answer *covered* queries (i.e. queries in which the variables in each negative atom is covered by a positive atom) by transforming the GNTGDs into disjunctive rules with stratified negation. Other significant decidable fragments of NTGDs under SMS include Magka, Krötzsch, and Horrocks (2013), Zhang, Zhang, and You (2015), and Alviano, Morak, and Pieris (2017).

Due to the availability of efficient ASP solvers, such as clingo (Gebser et al. 2012) and DLV (Leone et al. 2002), it is natural to consider reusing ASP solvers for query answering with GNTGDs, which correspond to ASP programs with function symbols. However, only a restricted subset of ASP programs, e.g. *finitely-ground* ones (Calimeri et al. 2008), can be handled by existing ASP solvers. Unfortunately, as shown in the next motivating example, a logic program translated from GNTGDs can be infinitely-ground.

Example 1. *(Gottlob et al. 2014, Example 1) Let* $D = \{Person(mary)\}$ *be a database and* Σ *be GNTGDs expressing that each person has at least one parent, each person belongs to either an odd generation or an even, and odd and even alternate between one generation and the next:*

$$Person(x) \rightarrow \exists y\, Parent(x, y), \tag{1}$$
$$Parent(x, y) \rightarrow Person(y), \tag{2}$$
$$Person(x), \mathsf{not}\ Even(x) \rightarrow Odd(x), \tag{3}$$
$$Person(x), \mathsf{not}\ Odd(x) \rightarrow Even(x), \tag{4}$$
$$Parent(x, y), Odd(x) \rightarrow Even(y), \tag{5}$$
$$Parent(x, y), Even(x) \rightarrow Odd(y). \tag{6}$$

The Skolemization of rule (1) *is*

$$Person(x) \rightarrow Parent(x, f(x)). \tag{7}$$

Let $\Sigma_0 = \Sigma \setminus \{(1)\} \cup \{(7)\}$. *Since program* $\Sigma_0 \cup D$ *is infinitely-ground, ASP solvers are not able to handle it. Moreover, consider a boolean conjunctive query* $Q = \exists x_1 x_2 x_3\ Parent(x_1, x_2) \wedge Parent(x_2, x_3) \wedge \neg Parent(x_1, x_3)$. *Since* Q *is not covered, the technique in Gottlob et al. (2014) does not help either. To the best of knowledge, no existing method can handle such queries over GNTGDs.*

To address this challenge, we rely on two techniques: the first one is *guarded chase forest (GCF)* (Calì, Gottlob, and Lukasiewicz 2012; Calì, Gottlob, and Kifer 2013), in which the nodes are derived atoms, and the edges encode the application of GTGDs. The second is *intelligent instantiation* (Calimeri et al. 2008), which has been used for characterizing finitely-grounded logic programs. In order to tame the

*Corresponding author

negation in GNTGDs, we have proposed a novel notion of *guarded instantiation forest (GIF)*, in which the nodes are ground rules, and the edges encode the procedure of intelligent instantiation. Another is the *prime block* for characterizing the repeated structures of GCF and GIF, an extension of *basic block* of GCF (Calautti, Gottlob, and Pieris 2015).

These two notions of GIF and prime blocks are quite powerful for GNTGDs. We first investigate the *ground termination* problem for GNTGDs, *i.e.*, deciding whether the program translated from GNTGDs is a finitely-ground program. Recall that this problem is undecidable for ASP (Calimeri et al. 2008). Fortunately, for GNTGDs, we show that this problem boils down to the existence of prime blocks, whose complexity is the same as the chase termination problem (Calautti, Gottlob, and Pieris 2015).

Next, we turn our attention to the problem of query answering, *esp.* for *infinitely-ground* GNTGDs. We show that this can be done by considering only finite fragments of the GIFs. Also, we show that prime block is a more fine-grained bound (thus more efficient) than guarded depth for query answering with guarded existential rules (Gottlob et al. 2014).

Finally, we develop a prototype and conduct experiments on a set of benchmarks. The results confirm that the our approach is scalable for query answering with GNTGDs.

Preliminaries

We briefly recall some basic notions for the rest of the paper.

Databases and Queries. We assume an infinite set Δ of *constants*, an infinite set Δ_n of *(labeled) nulls* (used as fresh Skolem terms), and an infinite set Δ_v of *variables*. A term is either a simple term or a functional term. A *simple term* is a constant, a null, or a variable. If $t_1, \ldots, t_n$ are terms and f is a function symbol (functor) of arity n, then $f(t_1, \ldots, t_n)$ is a functional term. We denote by $\mathbf{x}$ a sequence of variables $x_1, \ldots, x_k$ with $k \geq 0$. An *atom* a is of the form $R(t_1, \ldots, t_n)$, where R is an n-ary *relation symbol* (or *predicate*) and $t_1, \ldots, t_n$ are terms. We denote by $pred(a)$ its predicate and $dom(a)$ the set of all its arguments. For a set A of atoms, $dom(A){=}\bigcup_{a \in A} dom(a)$. An atom is *ground* if it contains no variables or nulls. A conjunction of atoms is identified with the set of all its atoms. A *relational schema* $\mathcal{R}$ is a finite set of relation symbols. An *instance* I over R is a (possibly infinite) set of variable free atoms over R. A *position* $P[i]$ in a relational schema is identified by a relational predicate P and its ith attribute. A *database* D over a relational schema $\mathcal{R}$ is a finite instance with relation symbols from $\mathcal{R}$ and with arguments only from Δ. A *conjunctive query (CQ)* over $\mathcal{R}$ has the form $Q(\mathbf{x}) = \exists \mathbf{y}\ \phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms with variables $\mathbf{x}$ and $\mathbf{y}$. An *atomic query* is a CQ with only one atom. A *Boolean CQ (BCQ)* is a CQ of the form $Q()$, written as Q. Given a D and a BCQ Q, the *query answer* of Q over D is *Yes*, denoted by $D \models Q$, if there is a *homomorphism* from Q to D.

Logic Programs and Stable Models. A *disjunctive logic program (DLP)* Π is a finite set of rules r of the form $\beta_1, \ldots, \beta_n, \mathsf{not}\, \beta_{n+1}, \ldots, \mathsf{not}\, \beta_m \rightarrow \alpha_1 \vee \ldots \vee \alpha_k$ where $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_m$ $(k{\geq}1,\ m{\geq}n{\geq}0)$ are atoms. A positive literal is an atom (e.g., β_1); a negative literal is the negation of an atom (e.g., $\mathsf{not}\, \beta_{n+1}$). We write $head(r) = \{\alpha_1, \ldots, \alpha_k\}$ for the head, and $body^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $body^-(r) = \{\beta_{n+1}, \ldots, \beta_m\}$ for the positive and negative body of r, respectively. A rule r is a *fact* if $k = 1$ and $m = 0$; r is a *constraint* if $k = 0$; r is normal if $k = 1$. A normal logic program (NLP) is a set of normal rules.

For a DLP Π, $facts(\Pi)$ and $heads(\Pi)$ denote the set of facts and heads in Π, respectively. We denote the *Herbrand universe* by $HU(\Pi)$ and the *Herbrand base* by $HB(\Pi)$. A variable-free rule r' is called an *instance* (or *ground rule*) of some rule $r \in \Pi$ if there is a function $h\colon \Delta_v \rightarrow HU(\Pi)$ *s.t.* $h(r) = r'$. The *grounding* (or *instantiation*) of Π, $ground(\Pi)$, is the set of all instances of all rules r from Π.

The *Gelfond-Lifschitz (GL) reduct* of a DLP Π *w.r.t.* a set $M \subseteq HB(\Pi)$, denoted by Π^M, is the (possibly infinite) ground positive program obtained from $ground(\Pi)$ by *(1)* deleting every rule r such that $body^-(r) \cap M \neq \emptyset$, and *(2)* deleting all negative literals from each remaining rule. M is a *stable model* of Π if M is a minimal model of Π^M (Gelfond and Lifschitz 1988; Ferraris, Lee, and Lifschitz 2011). The set of stable models of Π is denoted by $\mathsf{SM}(\Pi)$.

Normal TGDs and BNCQ. A *normal tuple generating dependency (NTGD)* σ is a first-order sentence of form $\forall \mathbf{x} \forall \mathbf{y} \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \psi(\mathbf{x}, \mathbf{z})$, where φ is a conjunction of literals, ψ is a conjunction of atoms, and each universally quantified variable appears in at least one positive conjunct of φ. W.l.o.g. we assume that each rule has a single atom in its head. For simplicity, we omit the universal quantifiers and write σ as:

$$\beta_1, \ldots, \beta_n,\ \mathsf{not}\, \beta_{n+1}, \ldots,\ \mathsf{not}\, \beta_m \rightarrow \exists \mathbf{z} \alpha$$

where β's and α are the atoms in ϕ and ψ, respectively. The notions of head and body atoms are defined as in DLP. σ^+ denotes the rules obtained by dropping all negative literals from the σ, and $\Sigma^+ = \bigcup_{\sigma \in \Sigma} \sigma^+$ for a set Σ of NTGDs.

An NTGD σ is a *TGD* if $body^-(\sigma) = \emptyset$. We say σ is a *guarded NTGD (GNTGD)* if it contains an atom in its body that covers all body variables of σ. The leftmost such atom is called the *guard atom* of σ, and other body atoms are the *side atoms* of σ. The notions of *stratified* and *full* negation are defined as usual.An NTGD is *linear* if there is only one positive atom in its body; it is *multi-linear* if all body atoms have the same variables. A linear NTGD is also a multi-linear NTGD. A *disjunctive NTGD (DNTGD)* σ is a formula $\forall \mathbf{x} \forall \mathbf{y} \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \psi(\mathbf{x}, \mathbf{z})$, where ψ is a disjunction of atoms and φ is a conjunction of literals.

The *stable models* of a finite set Σ of DGNTGDs and a database D is simply the stable models of the program $P{=}D{\cup}sk(\Sigma)$, where $sk(\Sigma)$ is the Skolemization of Σ (Gottlob et al. 2014). A *Boolean normal conjunctive query* (BNCQ) Q is an existentially closed conjunction of atoms and negated atoms of the form $\exists \mathbf{x} p_1(\mathbf{x_1}) \wedge \ldots \wedge p_m(\mathbf{x_m}) \wedge \neg p_{m+1}(\mathbf{x}_{m+1}) \wedge \ldots \wedge \neg p_{m+n}(\mathbf{x}_{m+n})$ $(m{\geq}1,\ n{\geq}0)$. We write $\Sigma \cup D {\models} Q$ if $M {\models} Q$ for all stable models M of $\Sigma {\cup} D$. $|Q|$ denotes the number of atoms in Q. A BNCQ Q is *safe* if $\mathbf{x}_{m+1} {\cup} \ldots {\cup} \mathbf{x}_{m+n} \subseteq \mathbf{x}_1 {\cup} \ldots {\cup} \mathbf{x}_m$; Q is *covered* if for every $i {\in} \{m{+}1, \ldots, m{+}n\}$, there exists $j {\in} \{1, \ldots, m\}$ *s.t.* $\mathbf{x}_i \subseteq \mathbf{x}_j$. Coveredness implies safeness, but not vice versa.

Chase. Let $\sigma = \phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z} \psi(\mathbf{x}, \mathbf{z})$ be a TGD and I an instance. We say that σ is *applicable w.r.t.* I if there exists a homomorphism h *s.t.* $h(\phi(\mathbf{x}, \mathbf{y})) \subseteq I$. The result of *applying* σ over I with h is the instance $J = I \cup \{h'(\psi(\mathbf{x}, \mathbf{z}))\}$, where $h'(z)$ is a fresh null, for every $z \in \mathbf{z}$. Such a single chase step is denoted by $I\langle\sigma, h\rangle J$. For a set Σ of TGDs and an instance I, a *chase sequence* for I under Σ is a sequence $(I_i\langle\sigma_i, h_i\rangle I_{i+1})_{i\geq 0}$ of chase steps *s.t. (1)* $I_0 = I$; *(2)* for each $i \geq 0$, $\sigma_i \in \Sigma$; and *(3)* $\bigcup_{i\geq 0} I_i \models \Sigma$. We call $\bigcup_{i\geq 0} I_i$ the result of this chase sequence. We denote by $chase(I, \Sigma)$ the result of an arbitrary chase sequence for I under Σ.

The *guarded chase forest (GCF)* for a finite set Σ of TGDs and a database D, $GCF(D, \Sigma)$, is a directed edge-labeled graph (V, E, λ), where $V = chase(D, \Sigma)$, λ is the labeling function, and an edge $e = (a, b)$ labeled with σ (*i.e.*, $\lambda(e) = \sigma$) belongs to E if b is obtained from a and possibly other atoms by a *one-step application* of a TGD $\sigma \in \Sigma$ with a as guard (Calì, Gottlob, and Kifer 2013). The *guarded depth* of an atom a in GCF for D and Σ, $depth(a)$, is the smallest length of a path from some $d \in D$ to a in GCF.

Given a finite set Σ of NTGDs with stratified negation and a database D, let $\Sigma_0 \cup ... \cup \Sigma_k$ be the stratification of Σ. We define the sets S_i as follows: $S_0 = chase(D, \Sigma_0)$; if $i > 0$, then $S_i = chase(S_{i-1}, \Sigma_i^{S_{i-1}})$, where $\Sigma_i^{S_{i-1}}$ is the GL-reduct of Σ_i w.r.t. S_{i-1}. S_k is a *canonical model* of D and Σ, which corresponds to the stable model of $D \cup \Sigma$.

Finitely-Ground Program and Intelligent Instantiation. *Finitely-ground (FG) programs* are an important class of DLP, whose stable models are computable. FG programs are characterized by *intelligent instantiation*. Consider a DLP Π, the set of its predicates is split into sets $C_1, \ldots, C_n$. Each C_i is called a *component* and the sequence $\gamma = \langle C_1, \ldots, C_n\rangle$ *components ordering*. Then, according to the component ordering, the rules of NLP can be split into a number of sets, called *modules*. Finally, the program can be safely instantiated with module ordering. Given a DLP Π and its component ordering $\langle C_1, \ldots, C_n\rangle$, for each C_i, the *module* M_i is the set of rules whose head contains some predicate $p \in C_i$; if a rule can belong to multiple modules, it belongs to the lowest one. Given a predicate p, we denote its component by $comp(p)$. For a set S_i of ground rules for C_i, and a set of ground rules R for the components preceding C_i, the *simplification* of S_i *w.r.t.* R, denoted by $Simpl(S_i, R)$, is obtained from S_i by:

1) deleting each rule r s.t. $a \in body^-(r)$ or $a \in head(r)$ for some $a \in facts(R)$,
2) eliminating each literal l from the remaining rules r
   - $l = a$, $a \in body^+(r)$, $a \in facts(R)$, or
   - $l = \textbf{not}\ a$, $a \in body^-(r)$, $comp(pred(a)) = C_j$ with $j < i$, and $a \notin heads(R)$.

For a set X of ground rules of M_i, and a set H of ground rules belonging only to M_j with $j < i$, let $\Phi_{M_i,H}(X) = Simpl(In_A(M_i), H)$, where $A = heads(H \cup X)$, $In_A(M_i) = \{r' \mid r \in M_i,\ r'$ is a ground instance of r, $body^+(r') \subseteq A\}$. The *intelligent instantiation* of Π for γ, denoted by Π^γ, is the last element S_n of the sequence *s.t.* $S_0 = facts(P)$, $S_i = S_{i-1} \cup \Phi^\infty_{M_i,S_{i-1}}(\emptyset)$, where $\Phi^\infty_{M_i,S_{i-1}}(\emptyset)$ is the least fixed point of $\Phi_{M_i,S_{i-1}}$. If Π^γ is finite for every component ordering, Π is said to be *finitely-ground*, and Π and Π^γ have the same stable models. For more details, please refer to Calimeri et al. (2008).

Guarded Instantiation Forest & Prime Block

In this section, we introduce the notions of guarded instantiation forest and prime block, and then prove that the ground termination problem is decidable. Based on the prime block, we propose the prime block-bounded instantiation/chase.

Guarded Instantiation Forest

Based on the GCF, we now present guarded instantiation forest (GIF), the first novel notion of this paper.

Definition 1. Given a finite set Σ of GNTGDs and a database D, let $P{=}D \cup sk(\Sigma)$ and P^γ an intelligent instantiation of P with γ as its component ordering, the *guarded instantiation forest (GIF)* of P, denoted by $GIF(D, \Sigma)$, is a directed graph (V, E, λ) where $V = P^\gamma$ is the set of nodes, λ is the labeling function, and an edge $e = (r, r')$ labeled with σ (*i.e.*, $\lambda(e) = \sigma$) belongs to E if r' is obtained from $head(r)$ and possibly other atoms by one-step application of a GNTGD $\sigma \in \Sigma$ with $head(r)$ as the guarded atom.

As in GCF, the *guarded depth* of a rule r in $GIF(D, \Sigma)$, denoted by $depth(r)$, is the smallest length of a path from some $a \in D$ to r in GIF. The following Lemma shows that for each rule in $GIF(D, \Sigma)$, there exists a corresponding atom in $GCF(D, \Sigma^+)$ with the same guarded depth.

Lemma 1. *Given a finite set Σ of GNTGDs and a database D, let $GIF(D, \Sigma) = (V_I, E_I, \lambda_I)$ and $GCF(D, \Sigma^+) = (V_C, E_C, \lambda_C)$, then for every rule $r \in V_I$, there exists an atom $a \in V_C$ with $head(r) = a$ and $depth(r) = depth(a)$.*

Proof. We prove by induction on the guarded depth of the nodes. The base case is trivial. For the inductive case, suppose the result holds for $depth(r) = k$. Then we consider a ground rule $r_{k+1} \in V_I$ with $depth(r_{k+1}) = k + 1$, there is a rule r'_{k+1} *s.t.* r_{k+1} is obtained from r'_{k+1} by the intelligent instantiation. Since the $body^+(r'_{k+1}) \subseteq heads(V_I)$, thus $body^+(r'_{k+1}) \subseteq V_C$. Thus, there is a rule r''_{k+1} in V_C *s.t.* $r''_{k+1} = r'^{+}_{k+1}$. Let $a = head(r''_{k+1})$, then $a \in V_C$, $head(r_{k+1}) = a$ and $depth(r_{k+1}) = 1 = depth(a)$. ☐

Intuitively, the process of generating GCF or GIF is similar. The main difference between GCF and GIF is that the nodes in GCF are atoms while the nodes in GIF are rules obtained by the intelligent instantiation.

Prime Blocks for GCF and GIF

Calautti, Gottlob, and Pieris (2015) introduced the *basic block* for GCF, which is the repeated segment of an infinite path in GCF. We extend the basic block of GCF to GIF.

Definition 2. Given a finite set Σ of GNTGDs and a database D, a set of rules $\{r_1, ..., r_{n+1}\}$ in $GIF(D, \Sigma)$ *s.t.* r_{i+1} is derived from r_i $(1 \leq i \leq n)$, the path from r_1 to r_n is a *basic block* of $GIF(D, \Sigma)$ if *(i)* the subtree rooted at r_1 is isomorphic to the subtree rooted at r_{n+1}, and *(ii)* every pair of rules in $\{r_1, r_2..., r_n\}$ are not isomorphic.

Now we present the second central notion *prime block* of this paper, which is a refinement of basic block. To do so, we need the auxiliary notion of S-isomorphic (Calì, Gottlob, and Lukasiewicz 2012). Given a set S of terms, two sets of atoms A_1 and A_2 are *S-isomorphic* iff a bijection β: $A_1 \cup dom(A_1) \rightarrow A_2 \cup dom(A_2)$ exists, *s.t. (i)* β and β^{-1} are homomorphisms, and *(ii)* $\beta(c) = c = \beta^{-1}(c)$ for all $c \in S$. In other words, A_1 and A_2 are S-isomorphic if they are isomorphic and the bijection for the isomorphism restricted to S is the identity function.

Definition 3. Given a finite set Σ of GNTGDs and a database D, a set of rules $\{r_1, ..., r_{n+1}\}$ in $GIF(D, \Sigma)$ *s.t.* r_{i+1} derived from r_i $(1 \leq i \leq n)$, the path from r_1 to r_n is a *prime block* if *(i)* the subtree rooted at r_1 is isomorphic to the subtree rooted at r_{n+1}, *(ii)* there exists $r' \in \{r_1, ..., r_n\}$ *s.t.* r' is $dom(r_1)$-isomorphic to r_{n+1}, and *(iii)* every pair in $\{r_1, ..., r_n\}$ are not $dom(r_1)$-isomorphic.

Next we show prime blocks are larger than basic blocks.

Proposition 1. *For a GIF, a prime block consists of at least two basic blocks.*

By Definition 3, we must find a rule r' *s.t.* r_1 is isomorphic to r'. By Definition 2, all rules isomorphic to r_1 are the first rule in each basic block. If the first rule of the second basic block is r', then the first basic block is the prime block. However, it can not satisfy condition (ii) of the prime block. Hence r' must be the rule that is not the first rule of the second basic block, which means the second basic block is also a part of the prime block.

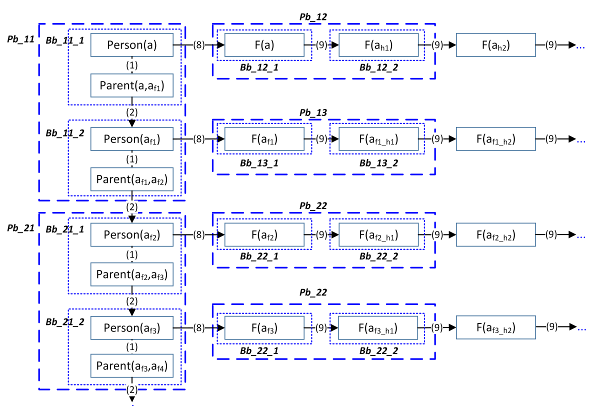

Figure 1: Basic blocks and prime blocks for $GCF(D_1, \Sigma_1)$.

Example 2 (Example 1 continued)**.** *Consider two TGDs:*

$$Person(x) \rightarrow F(x), \tag{8}$$

$$F(x) \rightarrow \exists y F(y). \tag{9}$$

Let $\Sigma_1 = \{(1), (2), (8), (9)\}$*,* $D_1 = \{Person(a)\}$*. Fig. 1 shows the basic blocks and prime blocks of* $GCF(D_1, \Sigma_1)$[1]*. Prime block* Pb_11 *has two basic blocks* Bb_11_1, Bb_11_2*.*

The notion of prime blocks of GIFs can be naturally extended to GCFs. The upper bound of the length of a prime block in GIF or GCF is characterized by Proposition 2.

[1]We use subscripted a to for nulls generated from a.

Proposition 2. *The length of a prime block in* $GIF(D, \Sigma)$ *(resp.,* $GCF(D, \Sigma)$*) is less than or equal to* $|R|(2\omega)^{\omega} 2^{|R|(2\omega)^{\omega}}$*, where* $|R|$ *denotes the numbers of predicates and* ω *is the maximal arity of a predicate in* R*.*

Proof. We give the proof in the case of $GIF(D, \Sigma)$, which can be applied in $GCF(D, \Sigma)$. Given an atom a, let $type(a)$ in $GIF(D, \Sigma)$ be the set of atoms in $heads(GIF(D, \Sigma))$ that only use constants and nulls from $dom(a)$ as arguments. Then, by Lemma 1 in Calì, Gottlob, and Lukasiewicz (2012), we can similarly obtain that, let S be a finite subset of $\Delta \cup \Delta_n$, and let r_1 and r_2 be rules from $GIF(D, \Sigma)$ *s.t.* the pairs $(head(r_1), type(r_1))$ and $(head(r_2), type(r_2))$ are S-isomorphic, then the subtree of r_1 (*i.e.*, all the nodes in $GIF(D, \Sigma)$ derived from r_1) and the subtree of r_2 are S-isomorphic. Towards a contradiction, suppose that there is a prime block with its head r_1 in $GIF(D, \Sigma)$ that its length is larger than $|R|(2\omega)^{\omega} 2^{|R|(2\omega)^{\omega}}$, then by Lemma 2 in (Calì, Gottlob, and Lukasiewicz 2012), there are at least two $dom(r_1)$-isomorphic pairs. Then, there are at least two rules that their subtree are $dom(r_1)$-isomorphic, which is contradict to the definition of prime block. Thus, the length of a prime block in $GIF(D, \Sigma)$ (*resp.*, $GCF(D, \Sigma)$) is less than or equal to $|R|(2\omega)^{\omega} 2^{|R|(2\omega)^{\omega}}$. □

Ground Termination for GNTGDs

The ground termination problem for GNTGDs, *i.e.*, given a finite set Σ of GNTGDs and a database D, deciding whether NLP $P = D \cup sk(\Sigma)$ is a finitely-ground program. Calautti, Gottlob, and Pieris (2015) proved that the chase termination problem for GTGDs is decidable and gave the complexity.

Theorem 1 (Calautti, Gottlob, and Pieris, 2015a)**.** *Given a finite set Σ of GTGDs, the problem of deciding whether the chase on D and Σ terminates for every database D is 2-EXPTIME-complete, and EXPTIME-complete for predicates of bounded arity.*

It is clear that if GIF is infinite then GCF is infinite.

Lemma 2. *Given a finite set Σ of GNTGDs and a database D, if $GIF(D, \Sigma)$ is infinite, then $GCF(D, \Sigma^+)$ is infinite.*

The following Lemma shows that prime blocks are sources of infinity GIFs.

Lemma 3. *Given a finite set Σ of GNTGDs and a database D, let NLP $P = D \cup sk(\Sigma)$, $GIF(D, \Sigma)$ is infinite iff there exist prime blocks in $GIF(D, \Sigma)$.*

Since the chase termination problem can be reduced to ground termination problem, we can obtain the lower bound of the ground termination problem. By Lemma 3 and Proposition 2, we can obtain the upper bound of the ground termination problem. Hence, we further prove that the ground termination problem for GNTGDs is also decidable and has the same complexity.

Theorem 2. *Given a finite set Σ of GNTGDs and a database D, the problem of deciding whether $GIF(D, \Sigma)$ is finite is 2-EXPTIME-complete, and EXPTIME-complete for predicates of bounded arity.*

Theorem 2 tells us whether a program translated from GNTGDs by Skolemization is a finitely ground logic program. If so, we can use ASP solvers directly. If not, however, we should find an approach to handle the infinite program.

Prime Block-bounded GIF/GCF

To handle the repeated infinite structures in GIF/GCF, we introduce the prime block-bounded instantiation/chase.

Definition 4. Given a finite set Σ of GNTGDs (*resp.*, GTGDs) and a database D, then $GIF_i(D, \Sigma)$ (*resp.*, $GCF_i(D, \Sigma)$) ($i \geq 1$) denotes the maximal subgraph of $GIF(D, \Sigma)$ (*resp.*, $GCF(D, \Sigma)$) with at most i prime blocks in every path starting from elements of D.

Example 3. For the GCF in Fig. 1, $GCF_1(D, \Sigma)$ consists of three prime blocks *Pb_11*, *Pb_12*, and *Pb_13*.

Let P_i^γ be the set of rules in $GIF_i(D, \Sigma)$ and $rules(P^\gamma, i)$ be the set of rules in the ith layer of prime blocks, *i.e.*, $rules(P^\gamma, i) = P_i^\gamma \backslash P_{i-1}^\gamma$, for $i > 1$, and $rules(P^\gamma, 1) = P_1^\gamma$. To obtain GIF/GCF, the nodes are not generated sequentially in order of the guarded depth, since the guarded depth of side atoms are possibly higher than that of guard atoms.

Calì, Gottlob, and Lukasiewicz (2012) presented the *proof* of an atom a in GCF is the minimal subgraph of GCF which contains the required nodes for generating a. Lemma 4 shows that for every atom a in GCF, there exists a bound for its proof in the bounded GCF.

Lemma 4 (Calì, Gottlob, and Lukasiewicz, 2012)**.** *Given a finite set Σ of GTGDs and a database D, let $GCF(D, \Sigma) = (V_C, E_C)$, then there is a constant k, depending only on R, s.t. $\forall a \in V_C$, the guarded depth of each atom in the proof of a is less than k.*

We then extend the proof to a rule in GIF and prove that given a node in the ith layer of prime blocks, all nodes in its proof are in the first $i + 1$ layers of prime blocks.

Lemma 5. *Given a finite set Σ of GNTGDs (resp., GTGDs) and a database D, for any rule r (resp., atom a) in $GIF_i(D, \Sigma)$ (resp., $GCF_i(D, \Sigma)$), the proof of r (resp., a) is in $GIF_{i+1}(D, \Sigma)$ (resp., $GCF_{i+1}(D, \Sigma)$).*

Thus, to obtain some rules (*resp.*, atoms) in $GIF_i(D, \Sigma)$ (*resp.*, $GCF_i(D, \Sigma)$), from Lemma 5, we can generate the forest within $i + 1$ layers prime blocks firstly, and only need to consider the nodes within i layers prime blocks.

We then extend the soundness and completeness proof of intelligent instantiation of DLP to the program translated from GNTGDs, which is possibly infinitely-ground.

Theorem 3. *Given a finite set Σ of GNTGDs and a database D, let NLP $P = D \cup sk(\Sigma)$ and γ its component ordering, let $P' = \bigcup_{i=0}^{+\infty} rules(P^\gamma, i)$, then P and P' have the same answer sets.*

Query Answering with GNTGDs

In this section, we first propose a method of query answering (QA) with GNTGDs by means of prime block-bounded chase. Then we propose another more efficient but involved method based on prime block-bounded instantiation.

QA via Prime Block-bounded Chases

We show that query answering with GTGDs can be handled via prime block-bounded chase.

Lemma 6. *Given a finite set Σ of GTGDs, a database D and a BCQ Q, then $D \cup \Sigma \models Q$ iff Q is true in the set of nodes in $GCF_{|Q|}(D, \Sigma)$.*

Proof. We show only the "($\Rightarrow$)" direction. Let $GCF(D, \Sigma) = (V, E, \lambda)$ and $GCF_{|Q|}(D, \Sigma) = (V_{|Q|}, E_{|Q|}, \lambda_{|Q|})$. Suppose that there exists a homomorphism from Q to V, and β is the one *s.t.* $depth(\beta) = \Sigma_{q \in Q} depth(\beta(q))$ is minimal. Now we show that $\beta(Q) \subseteq V_{|Q|}$. Suppose that $\beta(Q) \not\subseteq V_{|Q|}$, then there exists a layer of prime blocks in $V_{|Q|}$ that does not contain any atom in $\beta(Q)$. Consider a prime block P in the layer mentioned above. Let r_1 be the first atom in P. Let r be the atom in P $dom(r_1)$-isomorphic to the first atom r' of the next prime block. By definition 3, the subtree rooted at r is $dom(r_1)$-isomorphic to the subtree rooted at r'. Let μ be the homomorphism mapping the subtree rooted at r' to that rooted at r. Let $\beta' = \beta \circ \mu$, then β' is a homomorphism mapping Q to V. But $depth(\beta') = \Sigma_{q \in Q} depth(\beta'(q))$ is less than $depth(\beta)$, which contradicts to the assumption that $depth(\beta)$ is minimal. Therefore, $\beta(Q) \subseteq V_{|Q|}$. □

Lemma 6 provides us an algorithm for query answering with GTGDs, which is in general more efficient than the approach based on the bound of guarded depth (Calì, Gottlob, and Lukasiewicz 2012, Theorem 5). Recall that the upper bound of the length of a prime block is double-exponential in $|\mathcal{R}|$, because the number of all guarded TGDs (up to isomorphism) which are generated according to $\mathcal{R}$ is at most double-exponential in $|\mathcal{R}|$ in the worst case. However, given a set Σ of GNTGDs over $\mathcal{R}$, the size of a prime block is normally significantly less than its theoretical upper bound.

Then we can handle the QA with GNTGD via prime block-bounded chase by first transforming GNTGDs into GDNTGDs with stratified negation (Gottlob et al. 2014).

Theorem 4. *Given a finite set Σ of GDNTGDs with stratified negation, a database D, and a safe BNCQ Q, then Q is true in all canonical models of D and Σ iff Q is true in all GCFs of $GCF_{|Q|}(D, \Sigma)$.*

QA via Prime Block-bounded Instantiations

In this subsection, we develop a method of query answering with GNTGDs using prime block-bounded instantiation. Given an infinitely-ground program P translated from GNTGDs, our goal is to extract some finite fragments of $GIF(D, \Sigma)$ in order to employ ASP solvers for query answering. One natural attempt is to consider $GIF_{|Q|}(D, \Sigma)$ as in Lemma 6. But Example 4 shows this does not work.

Example 4. *Consider two additional two rules:*

$$Parent(x, y), \text{not } S(y) \rightarrow S(x) \quad (10)$$

$$Parent(x, y), S(y) \rightarrow S(x) \quad (11)$$

Let $\Sigma_2 = \{(1), (2), (10), (11)\}$*. Fig. 2 depicts* $GIF(D, \Sigma_2)$*. It is easy to see that the program* $D \cup \Sigma_2$ *has no answer sets, while for all* $k \geq 1$*, the* $GIF_k(D, \Sigma_2)$ *has answer sets.*

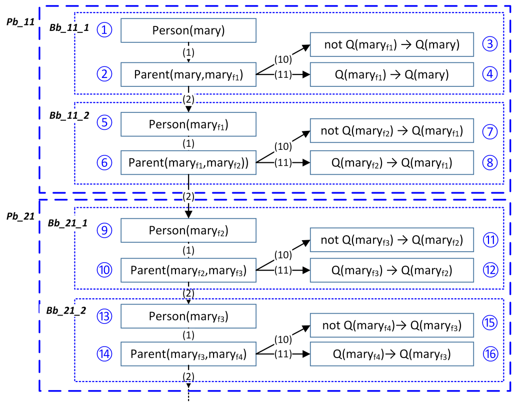

Figure 2: $GIF(D, \Sigma_2)$

To address this issue, we introduce a notion of *semi-stable model*. Given a ground program P, and a set m of atoms, let $rules_m(P) = \{r \mid r \in P,\ head(r) \in m\}$.

Definition 5. Given two ground programs P and P' with $P \subseteq P'$, we say $m \in \mathsf{SM}(P)$ is a *semi-stable model* of P' if there exists $m' \in \mathsf{SM}(P')$ *s.t.* $rules_m(P) \subseteq rules_{m'}(P')$. We write $\mathsf{SSM}(P, P')$ for the set of stable models of P that are semi-stable models of P'.

The following theorem gives a way of query answering under stable model semantics of GNTGDs by reducing to semi-stable models of prime block-bounded instantiation.

Theorem 5. *Given a finite set Σ of GNTGDs, a database D and a safe BNCQ Q, then $D \cup \Sigma \models Q$ iff Q is true in all semi-models of* $\mathsf{SSM}(P^{\gamma}_{|Q|}, P^{\gamma})$ *where NLP $P = D \cup sk(\Sigma)$ and γ its component ordering.*

The proof of Theorem 5 requires the following lemma.

Lemma 7. *Given a finite set Σ of GNTGDs, a database D, let NLP $P = D \cup sk(\Sigma)$ and γ its component ordering. If there exist $m \in \mathsf{SM}(P)$ and some $k > 0$, s.t. $m_k \not\subseteq m$ for all $m_k \in \mathsf{SM}(P^{\gamma}_k)$, then there exist $m_{k_0} \in \mathsf{SM}(P^{\gamma}_k)$ and $m' \in \mathsf{SM}(P)$ s.t. $m_{k_0} \subseteq m'$ and m' is isomorphic to m.*

Proof. (Sketch) First, it is easy to show that we can get a $P'_k \subseteq P^{\gamma}_k$ *s.t.* $\exists m'_k \in \mathsf{SM}(P'_k)$, $m'_k \subseteq m$. Also we know that $\forall m_k \in \mathsf{SM}(P^{\gamma}_k)$, $m'_k \not\subseteq m_k$. Let $hrules(P'_k, k)$ denote the set of rules in $rules(P'_k, k)$ that are in the highest basic blocks of the kth layer of prime blocks. Then $hrules(P'_k, k)$ is isomorphic to $hrules(P^{\gamma}_k, k)$. Also let $hrules_{m'_k}(P'_k, k)$ denote the set of rules in $hrules(P'_k, k)$ that are used to generate m'_k. We try to find an $m_{k_0} \in \mathsf{SM}(P^{\gamma}_k)$ *s.t.* $hrules_{m'_k}(P'_k, k)$ is isomorphic to $hrules_{m_{k_0}}(P^{\gamma}_k, k)$. If we cannot find such m_{k_0}, it is easy to show that m'_k is not a semi-stable model of P^{γ}, which contradicts to the assumption. If such m_{k_0} exists, then we can obtain an isomorphism h mapping $hrules_{m'_k}(P'_k, k)$ to $hrules_{m_{k_0}}(P^{\gamma}_k, k)$ and such an isomorphism h can be extended to m. Then we obtain an $m' \in \mathsf{SM}(P^{\gamma})$ *s.t.* $rules(P^{\gamma}, m)$ is isomorphic to $rules(P^{\gamma}, m')$. Thus, $m' \supseteq m_{k_0}$ is isomorphic to m. □

We proceed with the proof of Theorem 5: For any $m \in \mathsf{SM}(P^{\gamma})$, there are two cases (1) $\exists m_{|Q|} \in \mathsf{SM}(P^{\gamma}_{|Q|})$ and $m_{|Q|} \subseteq m$, or (2) $\forall m_{|Q|} \in \mathsf{SM}(P^{\gamma}_{|Q|})$, $m_{|Q|} \not\subseteq m$. In (2), by Lemma 7 we can obtain another model $m' \in \mathsf{SM}(P^{\gamma})$ *s.t.* $\exists m'_{|Q|} \in \mathsf{SM}(P^{\gamma}_{|Q|})$, $m'_{|Q|} \subseteq m'$ and m is isomorphic to m'. So $|Q|$ is a proper upper bound of prime block-instantiation which can be done analogously to Lemma 6. □

Example 5 illustrates how Lemma 7 plays a role.

Example 5. *Let Σ_3=$\{(1), (2), (10)\}$, BCQ Q=$\exists x S(x)$. Then $GIF(D, \Sigma_3)$ depicted in Fig. 3 has two stable models m and m', s.t. m includes $\{S(mary), S(mary_{f2}), ...\}$, and m' includes $\{S(mary_{f1}), S(mary_{f3}), ...\}$. $GIF_1(D, \Sigma_3)$*

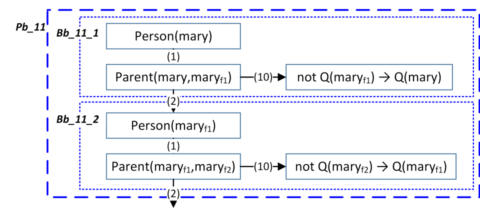

Figure 3: $GIF(D, \Sigma_3)$

has only one stable model m_1 that includes $S(mary_{f1})$. It is easy to see that $m_1 \in \mathsf{SSM}(P^{\gamma}_1, P^{\gamma}_2)$, but $m_1 \not\subseteq m'$. However, since m and m' are isomorphic, to check $D \cup \Sigma_3 \models Q()$, it suffices to check $m_1 \models Q()$, which is true.

We notice that our approach can handle safe query, while the QCHECK algorithm (Gottlob et al. 2014) can only handle covered query, *e.g.* , the safe BNCQ in Example 1, can be dealt analogously to the discussion in Example 5.

To apply Theorem 5, we still need an algorithm to verify whether a stable model of $P^{\gamma}_{|Q|}$ is a semi-stable model of P^{γ}. Theorem 6 serves as such an algorithm.

Theorem 6. *Given a finite set Σ of GNTGDs, a database D, let NLP $P = D \cup sk(\Sigma)$, γ its component ordering, $m_k \in \mathsf{SM}(P^{\gamma}_k)$ $(k > 0)$, $m_k \in \mathsf{SSM}(P^{\gamma}_k, P)$ iff there exists a P'_{k+1} s.t. $m_k \in \mathsf{SSM}(P^{\gamma}_k, P'_{k+1})$, where P'_{k+1} is obtained from P^{γ}_{k+1} by subtracting some or no basic blocks in the $(k{+}1)$th prime blocks.*

Proof. (Sketch) ($\Leftarrow$): If there exists a $P'_{k+1} \subseteq P^{\gamma}_{k+1}$ with basic block subtraction or $P'_{k+1} = P^{\gamma}_{k+1}$ *s.t.* m_k is a semi-stable model of P'_{k+1}. Because of the isomorphism of basic block, we can also add rules inductively to obtain P^{γ} and m_k is a semi-stable model of P^{γ}.
($\Rightarrow$): If there exists no $P'_{k+1} \subseteq P^{\gamma}_{k+1}$ with basic block subtraction nor $P'_{k+1} = P^{\gamma}_{k+1}$ *s.t.* , m_k is a semi-stable model of P'_{k+1}. Given $P'' \supseteq P^{\gamma}_{k+1}$, we can infer that m_k is not a semi-stable model of P''. We know that each rule in $rules(P^{\gamma}, k + j)$ is $dom(A)$-isomorphic to some rule in $rules(P^{\gamma}, k + 1)$, where $j > 1$ and A is the set of rules in P^{γ}_k. If m_k is a semi-stable model of P'', then there exists some $m'' \in SM(P'')$ *s.t.* $rules_{m_k}(P^{\gamma}_k) \subseteq rules_{m''}(P'')$. For any rule $r'' \in rules_{m'' \setminus m_k}(P'')$, there exists some $r \in rules(P^{\gamma}, k + 1)$ $dom(A)$-isomorphic to r''. Let R be the set of rules r mentioned above and $m'_k = m_k \cup heads(R)$. Then m'_k is a stable model of P'_{k+1}, and m_k is a semi-stable model of P'_{k+1} due to the existence of m'_k, which contradicts

the assumption that m_k is not a semi-stable model of P'_{k+1}. So m_k is not a semi-stable model of P''. Due to the arbitrariness of P'',m_k is not a stable model of P^γ, contradicting the assumption that m_k is a semi-stable model of P^γ. □

The following example shows how to apply Theorem 6.

Example 6. *(Example 4 continued) Let $P{=}D \cup sk(\Sigma_2)$. As shown in Fig. 2, the program P_1^γ has the answer set $m_1 \supseteq \{S(mary), S(mary_{f1})\}$ and P_1^γ has the answer set $m_2 \supseteq \{S(mary), S(mary_{f1}), S(mary_{f2}), S(mary_{f3})\}$. Also $rules_{m_1}(P_1^\gamma) = \{\textcircled{1}, \textcircled{2}, \textcircled{4}, \textcircled{5}, \textcircled{6}, \textcircled{7}\}$, $rules_{m_2}(P_2^\gamma) = \{\textcircled{1}, \textcircled{2}, \textcircled{4}, \textcircled{5}, \textcircled{6}, \textcircled{8}, \textcircled{9}, \textcircled{10}, \textcircled{12}, \textcircled{13}, \textcircled{14}, \textcircled{15}\}$.*

We obtain P'_2 from P_2^γ by subtracting basic block Bb_21_2 from prime block Pb_21. Then program P'_2 has one stable model $m'_2 \supseteq \{S(mary), S(mary_{f1}), S(mary_{f2}),$ and $rules_{m'_2}(P'_2) = \{\textcircled{1}, \textcircled{2}, \textcircled{4}, \textcircled{5}, \textcircled{6}, \textcircled{8}, \textcircled{9}, \textcircled{10}, \textcircled{12}\}$. We can verify that $rules_{m_1}(P_1^\gamma) \nsubseteq rules_{m_2}(P_2^\gamma)$ and $rules_{m_1}(P_1^\gamma) \nsubseteq rules_{m'_2}(P'_2)$, and we know that m_1 is not a semi-stable model of P^γ by Theorem 6. Then following Theorem 5, P does not have any answer set.

Finally, we show that query answering with (multi-)linear NTGD can be done more efficiently.

Theorem 7. *Given a finite set Σ of (multi-)linear NTGDs, a database D over schema $\mathcal{R}$, a safe BNCQ Q, then $D \cup \Sigma \models Q$ iff Q is true in all stable models of $P_{|Q|}^\gamma$, where NLP $P = D \cup sk(\Sigma)$ and γ its component ordering.*

Proof. (Sketch) By Theorem 5, it suffices to show that each stable model m_k of P_k^γ is also a semi-stable model of P^γ. Given such a model m_k, we can inductively construct stable models m_{k+j} of P_{k+j}^γ, for $j > 0$, by enlarging m_{k+j-1}. Let $m = \bigcup_{j=0}^{\infty} m_{k+j}$. Then m is a stable model of P^γ. □

Experimental Evaluation

We implemented a prototype system in Python for query answering with GNTGDs[2]. To the best of our knowledge, this is the first system with such functionality. The code and data to reproduce the experiments are in the online appendix.

Data. We considered three GNTGDs: LUBM[3], GeoConcepts[4], and Vicodi[5] and two (multi-)linear TGDs: DEEP-100(-200)[6] as benchmarks, which are modified by changing atoms and adding negations to make sure modified ontologies are infinitely-ground and have full negation. We designed different safe BNCQs and translated into constraints.

System. Our system has two modes: (1) the mode $\mathcal{M}_{GCF}$ first translates GNTGDs into GDNTGDs with stratified negation (Gottlob et al. 2014) and then uses Theorem 4. (2) the GIF mode $\mathcal{M}_{GIF}$ implements Theorems 5, 6, and 7. We use the ASP solver clingo-4.4.0.

Evaluation[7]**.** The results are summarized in Table 1. We note that $\mathcal{M}_{GCF}$ can only handle small queries and requires

[2]Code and datasets. https://github.com/sysulic/GNTGDs

[3]LUBM. http://swat.cse.lehigh.edu/projects/lubm/

[4]GeoConcepts. http://www.kr.tuwien.ac.at/research/projects/myits/

[5]Vicodi. http://www.vicodi.org

[6]DEEP-100(-200). https://github.com/dbunibas/chasebench

[7]All experiments run in 64-bit Linux on a machine with 2.10GHz Intel Xeon and 128G 1333 MHz memory.

Benchmark	$\|\Sigma\|$	$\|D\|$	$\|N\|$	$\|Q\|$	$\mathcal{M}_{GIF}$ Size	$\mathcal{M}_{GIF}$ Time$_S$	$\mathcal{M}_{GIF}$ Time$_T$	$\mathcal{M}_{GCF}$ Num$_F$	$\mathcal{M}_{GCF}$ Size	$\mathcal{M}_{GCF}$ Time$_T$
Modified LUBM	110	100	1%	2	1101	0	0.4	$1.2K$	$3.4M$	582.42
			3%	6	4953	43.7	49.9	—	—	—
Modified GeoConcepts	446	122	1%	2	808	1.2	1.8	$3.0K$	$24.1M$	3166.79
			3%	6	6064	80.4	92.5	—	—	—
Modified Vicodi	236	165	1%	2	1603	0	0.8	$3.5K$	$29.8M$	6737.74
			7%	6	18003	1040.7	1088.2	—	—	—
Modified DEEP-100	4241	1000	1%	2	6857	0	8.0	—	—	—
		10000	3%	5	69420	0	50.0	—	—	—
Modified DEEP-200	4539	1000	1%	2	9070	0	11.3	—	—	—
		10000	3%	5	108530	0	124.9	—	—	—

Table 1: Experimental Results of Query Answering with GNTGDs. $|D|$ (*resp.*, $|Q|$) stands for the size of database (*resp.*, query), $|\Sigma|$ the number of GNTGDs, Num$_F$ the number of GCFs generated, $|N|$ the ratio of the number of rules with negation and the total number of rules, Size the number of nodes in the forest, Time$_S$ the time to obtain semi-stable model, Time$_T$ the total time. Time out (7200 seconds) or out of memory are indicated as "–". All times are in seconds. K and M stand for 10^3 and 10^6.

more time and space in generating GCFs. The reason is that the number of GCFs grows exponentially since it enumerates all stable models. In contrast, $\mathcal{M}_{GIF}$ is more efficient in both space and time, although it has an extra step of checking semi-stable models. This justifies our claim that GIF is more suitable than GCF for query answering. For the last two benchmarks which are (multi-)linear and whose datasets are larger than those of the first three, GIF can process them in an acceptable period of time since (multi-)linear NTGDs does not need to check semi-stable models. However, GCF cannot handle any of them. We stress that our prototype is a proof-of-concept, and there are many possible optimizations but they are beyond the scope of this paper.

Conclusion and Future Work

We have proposed a framework of query answering with GNTGDs and safe queries using existing ASP solvers, By introducing guarded instantiation forest and prime block, we have proved the decidability and complexity of the ground termination problem for GNTGDs, and developed a query answering algorithm via prime block-bounded instantiation.

Future work will extend our approach to other fragments of GNTGDs, *e.g.*, weakly, *etc.* We also plan to develop optimization techniques to improve the performance.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China (No. 61573386, 61976232, 61972455), Guangdong Province Natural Science Foundation (No. 2016A030313292, 2017A070706010 (soft science), 2018A030313086), Guangdong Province Science and Technology Plan projects (No. 2016B030305007 and 2017B010110011), Guangzhou Science and Technology Project (No. 201804010435), the EU H2020 project INODE, by the Italian PRIN project HOPE, by the Free University of Bozen-Bolzano through the projects QUADRO, KGID and GeoVKG.

References

Alviano, M.; Morak, M.; and Pieris, A. 2017. Stable model semantics for tuple-generating dependencies revisited. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 377–388. ACM.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, 73–85.

Beeri, C., and Vardi, M. Y. 1984. A proof procedure for data dependencies. *Journal of the ACM* 31(4):718–741.

Calautti, M.; Gottlob, G.; and Pieris, A. 2015. Chase termination for guarded existential rules. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 91–103.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *Journal of Artificial Intelligence Research* 48:115–174.

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

Calì, A.; Gottlob, G.; and Pieris, A. 2010. Query answering under non-guarded rules in datalog+/-. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, 1–17.

Calimeri, F.; Cozza, S.; Ianni, G.; and Leone, N. 2008. Computable functions in ASP: theory and implementation. In *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, 407–424.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Stable models and circumscription. *Artificial Intelligence* 175(1):236–263.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, 1070–1080.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2014. Stable model semantics for guarded existential rules and description logics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*, 258–267.

Leone, N.; Pfeifer, G.; Faber, W.; Calimeri, F.; Dell'Armi, T.; Eiter, T.; Gottlob, G.; Ianni, G.; Ielpa, G.; Koch, C.; et al. 2002. The dlv system. In *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, 537–540.

Magka, D.; Krötzsch, M.; and Horrocks, I. 2013. Computing stable models for nonmonotonic existential rules. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 1031–1038.

Zhang, H.; Zhang, Y.; and You, J.-H. 2015. Existential rule languages with finite chase: Complexity and expressiveness. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 1678–1685.