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 Wang3
1School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, P.R.China
2KRDB Research Centre, Free University of Bozen-Bolzano, Piazza Domenicaní 3, 39100 Bolzano, Italy
3School of Information and Communication Technology, Griffith University, Brisbane, QLD 4111, Australia
wanhai@mail.sysu.edu.cn, xiao@inf.unibz.it, {wangchlin, liuxq35, chenhj46}@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. Unfortu-

Example 1. (Gottlob et al. 2014, Example 1) Let \( D = \{ \text{Person(mary)} \} \) be a database and \( \Sigma \) 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:

\[
\begin{align*}
\text{Person}(x) & \rightarrow \exists y \, \text{Parent}(x,y), \\
\text{Parent}(x,y) & \rightarrow \text{Person}(y), \\
\text{Person}(x), \text{not Even}(x) & \rightarrow \text{Odd}(x), \\
\text{Person}(x), \text{not Odd}(x) & \rightarrow \text{Even}(x), \\
\text{Parent}(x,y), \text{Odd}(x) & \rightarrow \text{Even}(y), \\
\text{Parent}(x,y), \text{Even}(x) & \rightarrow \text{Odd}(y).
\end{align*}
\]

The Skolemization of rule (1) is

\[
\text{Person}(x) \rightarrow \text{Parent}(x, f(x)).
\]

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 \( Q = \exists x_1 x_2 x_3 \, \text{Parent}(x_1,x_2) \land \text{Parent}(x_2,x_3) \land \neg \text{Parent}(x_1,x_3) \). 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
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, *e.g.* 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 $\Delta$ of constants, an infinite set $\Delta_0$ of (labeled) nulls (used as fresh Skolem terms), and an infinite set $\Delta_v$ 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 (function) of arity $n$, then $f(t_1, \ldots, t_n)$ is a functional term. We denote by $x$ a sequence of variables $x_1, \ldots, x_k$ with $k \geq 0$. An atom $a$ is of the form $\neg a$, where $a$ is an $n$-ary relation symbol (predicate) and $t_1, \ldots, t_n$ are terms. We denote by $\text{pred}(a)$ its predicate and $\text{dom}(a)$ the set of all its arguments. For a set $A$ of atoms, $\text{dom}(A) = \bigcup_{a \in A} \text{dom}(a)$. 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 $R$ 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 $R$ is a finite instance with relation symbols from $R$ and with arguments only from $\Delta$. A conjunctive query (CQ) over $R$ has the form $Q(x) = \exists y \phi(x, y)$, where $\phi(x, y)$ is a conjunction of atoms with variables $x$ and $y$. An atomic query is a CQ with only one atom. A Boolean CQ (BCQ) is a CQ of the form $\phi \lor \psi$, where $\phi$ and $\psi$ are conjunctive queries. We denote the conjunction of conjunctive queries $Q_1 \land Q_2$ by $Q_1 \land Q_2$. The query answer of $Q$ over database $D$ 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) $P$ is a finite set of rules $r$ of the form $\beta_1, \ldots, \beta_n \land \neg \beta_{n+1}, \ldots, \neg \beta_m \rightarrow \alpha_1 \lor \cdots \lor \alpha_k$ where $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_n$ ($k \geq 1$, $m \geq n \geq 0$) are atoms. A positive literal is an atom (e.g., $\beta$); a negative literal is the negation of an atom (e.g., $\neg \beta_{n+1}$). We write $\text{head}(r) = \{\alpha_1, \ldots, \alpha_k\}$ for the head, and $\text{body}^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $\text{body}^-(r) = \{\beta_{n+1}, \ldots, \beta_m\}$ 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 $P$, facts $\Pi$ and heads $\Pi$ denote the set of facts and heads in $P$, respectively. We denote the Herbrand universe by $\mathcal{U}(\Pi)$ and the Herbrand base by $\mathcal{B}(\Pi)$. A variable-free rule $r'$ is called an instance (or ground rule) of some rule $r \in \Pi$ if there is a function $h : \Delta_v \rightarrow \mathcal{U}(\Pi)$ s.t. $h(r) = r'$. The grounding (or instantiation) of $P$, $\text{ground}(P)$, is the set of all instances of all rules $r$ from $P$.

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

**Normal TGDs and BNCQ.** A normal tuple generating dependency (NTGD) is a first-order sentence of form $\forall x \forall y \varphi(x, y) \rightarrow \exists \psi(x, z)$, where $\varphi$ is a conjunction of literals, $\psi$ is a conjunction of atoms, and each universally quantified variable appears in at least one positive conjunct of $\varphi$. 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 $\sigma$ as $\beta_1, \ldots, \beta_n, \not\beta_{n+1}, \ldots, \not\beta_m \rightarrow \exists x \alpha$, where $\beta$s and $\alpha$ are the atoms in $\varphi$ and $\psi$, respectively. The notions of head and body atoms are defined as in DLP. $\sigma^-$ denotes the rules obtained by dropping all negative literals from $\sigma$, and $\Sigma^+ = \bigcup_{\sigma \in \Sigma} \sigma^-$ for a set $\Sigma$ of NTGDs.

An NTGD $\sigma$ is a TGD if $\text{body}^-(\sigma) = \emptyset$. We say $\sigma$ is a guarded NTGD (GNTGD) if it contains an atom in its body that covers all body variables of $\sigma$. The leftmost such atom is called the guard atom of $\sigma$, and other body atoms are the side atoms of $\sigma$. 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) $\sigma$ is a formula $\forall x \forall y \varphi(x, y) \rightarrow \exists \psi(x, z)$, where $\psi$ is a disjunction of atoms and $\varphi$ is a conjunction of literals.

The stable models of a finite set $\Sigma$ of DGNNTGDs and a database $D$ is simply the stable models of the program $P = D \cup \text{skd}(\Sigma)$, where $\text{skd}(\Sigma)$ is the Skolemization of $\Sigma$ (Gottlob et al. 2014). A Boolean normal conjunctive query (BNCQ) $Q$ is an existentially closed conjunction of atoms and negated atoms of the form $\exists x_1 \land \cdots \land \exists x_m \land \neg p_{m+1}(x_{m+1}) \land \cdots \land \neg p_{m+n}(x_{m+n})$ ($m \geq 1$, $n \geq 0$). 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 $x_m+1, \ldots, x_{m+n} \subseteq x_1, \ldots, x_k$; $Q$ is covered if for every $i \in \{m+1, \ldots, m+n\}$, there exists $j \in \{1, \ldots, m\}$ s.t. $x_i \subseteq x_j$. Coveredness implies safeness, but not vice versa.

**Chase.** Let $\sigma = \phi(x, y) \rightarrow \exists \psi(x, z)$ be a TGD and $I$ an instance. We say that $\sigma$ is applicable w.r.t. $I$ if there exists a
homomorphism $h$ s.t. $h(\phi(x, y)) \subseteq I$. The result of applying $\sigma$ over $I$ with $h$ is the instance $J = I \cup \{h(\psi(x, z))\}$, where $h(z)$ is a fresh null, for every $z \in \mathcal{Z}$. Such a single chase step is denoted by $I(\sigma, h)J$. For a set $\Sigma$ of TGDs and an instance $I$, a chase sequence for $I$ under $\Sigma$ is a sequence $(I_i(\sigma_i, h_i)I_{i+1})_{i \geq 0}$ of chase steps s.t. $(1) I_0 = I$; $(2)$ for each $i \geq 0$, $\sigma_i \in \Sigma$; and $(3) \bigcup_{i \geq 0} I_i = \Sigma$. We call $\bigcup_{i \geq 0} I_i$ the result of this chase sequence. We denote by $\text{chase}(I, \Sigma)$ the result of an arbitrary chase sequence for $I$ under $\Sigma$.

The guarded chase forest (GCF) for a finite set $\Sigma$ of TGDs and a database $\mathcal{D}$, $\text{GCF}(\mathcal{D}, \Sigma)$, is a directed edge-labeled graph $(V, E, \lambda)$, where $V = \text{chase}(\mathcal{D}, \Sigma)$, $\lambda$ is the labeling function, and an edge $e = (a, b)$ labeled with $\sigma$ (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 (Cali, Gottlob, and Kifer 2013). The guarded depth of an atom $a$ in GCF for $\mathcal{D}$ and $\Sigma$, $\text{depth}(a)$, is the smallest length of a path from some $d \in D$ to $a$ in GCF.

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

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 $\Pi$, the set of its predicates is split into sets $C_1, \ldots, C_n$. Each $C_i$ is called a component and the sequence $\gamma = (C_1, \ldots, C_n)$ 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 $\Pi$ and its component ordering $(C_1, \ldots, C_n)$, 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 $\text{comp}(p)$. 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 $\text{Simpl}(S_i, R)$, is obtained from $S_i$ by:

1) deleting each rule $r$ s.t. $a \in \text{body}^-(r)$ or $a \in \text{head}(r)$ for some $a \in \text{facts}(R)$,
2) eliminating each literal $l$ from the remaining rules $r$.

\begin{itemize}
    \item $l = a$, $a \in \text{body}^+(r)$, $a \in \text{facts}(R)$, or
    \item $l = \text{not} a$, $a \in \text{body}^-(r)$, $\text{comp}(\text{pred}(a)) = C_j$ with $j < i$, and $a \notin \text{heads}(R)$.
\end{itemize}

For a set $X$ of ground rules of $M_j$ and a set $H$ of ground rules belonging only to $M_j$ with $j < i$, let $\Phi_{M_j, H}(X) = \text{Simpl}(\text{In}_A(M_j), H)$, where $\text{In}_A(M_j) = \{r \leftarrow r' \mid r \in M_j, r' \text{ is a ground instance of } r, \text{body}^+(r') \subseteq A\}$. The intelligent instantiation of $\Pi$ for $\gamma$, denoted by $\Pi^\gamma$, is the last element $S_0$ of the sequence s.t. $S_0 = \text{facts}(P)$, $S_i = S_{i-1} \cup \Phi_{M_j, S_{i-1}}^\infty(\emptyset)$, where $\Phi_{M_j, S_{i-1}}^\infty(\emptyset)$ is the least fixed point of $\Phi_{M_j, S_{i-1}}$. If $\Pi^\gamma$ is finite for every component ordering, $\Pi$ is said to be finitely-ground, and $\Pi$ and $\Pi^\gamma$ 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 $\Sigma$ of GNTGDs and a database $\mathcal{D}$, let $P = D \cup sk(\Sigma)$ and $P^\gamma$ an intelligent instantiation of $P$ with $\gamma$ as its component ordering, the guarded instantiation forest (GIF) of $P$, denoted by $\text{GIF}(\mathcal{D}, \Sigma)$, is a directed graph $(V, E, \lambda)$ where $V = P^\gamma$ is the set of nodes, $\lambda$ is the labeling function, and an edge $e = (r, r')$ labeled with $\sigma$ (i.e., $\lambda(e) = \sigma$) belongs to $E$ if $r'$ is obtained from $\text{head}(r)$ and possibly other atoms by one-step application of a GNTGD $\sigma \in \Sigma$ with $\text{head}(r)$ as the guarded atom.

As in GCF, the guarded depth of a rule $r$ in $\text{GIF}(\mathcal{D}, \Sigma)$, denoted by $\text{depth}(r)$, 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 $\text{GIF}(\mathcal{D}, \Sigma)$, there exists a corresponding atom in $\text{GCF}(\mathcal{D}, \Sigma^+)$ with the same guarded depth.

Lemma 1. Given a finite set $\Sigma$ of GNTGDs and a database $\mathcal{D}$, let $\text{GIF}(\mathcal{D}, \Sigma) = (V_1, E_1, \lambda_1)$ and $\text{GCF}(\mathcal{D}, \Sigma^+) = (V_2, E_2, \lambda_2)$, then for every rule $r \in V_1$, there exists an atom $a \in V_2$ with $\text{head}(r) = a$ and $\text{depth}(r) = \text{depth}(a)$.

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 $\text{depth}(r) = k$. Then we consider a ground rule $r_{k+1} \in V_1$ with $\text{depth}(r_{k+1}) = k + 1$, there is a rule $r_{k+1}$ s.t. $r_{k+1}$ is obtained from $r_k$ by the intelligent instantiation. Since the $\text{body}^+(r_{k+1}) \subseteq \text{heads}(V_1)$, thus $\text{body}^+(r_{k+1}) \subseteq V_2$. Thus, there is a rule $r''_{k+1}$ in $V_2$ s.t. $r''_{k+1} = r'_{k+1}$. Let $a = \text{head}(r''_{k+1})$, then $a \in V_2$, $\text{head}(r_{k+1}) = a$ and $\text{depth}(r_{k+1}) = 1 = \text{depth}(a)$.

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 $\Sigma$ of GNTGDs and a database $\mathcal{D}$, a set of rules $\{r_1, \ldots, r_n\}$ in $\text{GIF}(\mathcal{D}, \Sigma)$ 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 $\text{GIF}(\mathcal{D}, \Sigma)$ if (i) the subtree rooted at $r_1$ is isomorphic to the subtree rooted at $r_n$; and (ii) every pair of rules in $\{r_1, r_2, \ldots, r_n\}$ 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 (Cali, 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 $\beta: A_1 \cup dom(A_1) \to A_2 \cup dom(A_2)$ exists, s.t. (i) $\beta$ and $\beta^{-1}$ 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 $\Sigma$ of GNTGDs and a database $D$, a set of rules $\{r_1, \ldots, r_{n+1}\}$ 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 $r' \in \{r_1, \ldots, r_n\}$ s.t. $r'$ is $dom(r_1)$-isomorphic to $r_{n+1}$, and (iii) every pair in $\{r_1, \ldots, r_n\}$ 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.

**Example 2** (Example 1 continued). Consider two TGDs: $Person(x) \rightarrow F(x)$, $F(x) \rightarrow \exists y F(y)$.

Let $\Sigma_1 = \{(1), (2), (8), (9)\}$, $D_1 = \{Person(a)\}$. Fig. 1 shows the basic blocks and prime blocks of GIF($D_1, \Sigma_1$)

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.

**Proposition 2.** The length of a prime block in GIF($D, \Sigma$) (resp., GCF($D, \Sigma$)) is less than or equal to $|R|(2\omega)^{|\omega|^2}$, where $|R|$ denotes the numbers of predicates and $\omega$ 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 Cali, Gottlob, and Lukasiewicz (2012), we can similarly obtain that, let $S$ be a finite subset of $\sum \cup \Delta_n$, 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 $r_1$ isomorphic to $r_{n+1}$, and (iii) every pair in $\{r_1, \ldots, r_n\}$ are not $dom(r_1)$-isomorphic.

Ground Termination for GNTGDs

The ground termination problem for GNTGDs, i.e., given a finite set $\Sigma$ 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 $\Sigma$ of GTGDs, the problem of deciding whether the chase on $D$ and $\Sigma$ 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 GFC is infinite.

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

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

**Lemma 3.** Given a finite set $\Sigma$ 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 $\Sigma$ 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 $\Sigma$ of GNTGDs (resp., GTGDs) and a database $D$, then $GIF_1(D, \Sigma)$ (resp., $GCF_1(D, \Sigma)$) $(i \geq 1)$ 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 $P^i_k$ be the set of rules in $GIF_1(D, \Sigma)$ and rules$(P^i_k, i)$ be the set of rules in the $i$th layer of prime blocks, i.e., rules$(P^i_k, i) = P^i_k \backslash P^i_{k-1}$, for $i > 1$, and rules$(P^i_k, 1) = P^i_k$. 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.

Cali, 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** (Cali, Gottlob, and Lukasiewicz, 2012). Given a finite set $\Sigma$ of GTGDs and a database $D$, let $GCF(D, \Sigma) = (V, E, \lambda)$, then there is a constant $k$, depending only on $R$, s.t. $\forall a \in V$, 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 $i$th 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 $\Sigma$ of GNTGDs (resp., GTGDs) and a database $D$, for any rule $r$ (resp., atom $a$) in $GIF_1(D, \Sigma)$ (resp., $GCF_1(D, \Sigma)$), the proof of $r$ (resp., $a$) in $GIF_{i+1}(D, \Sigma)$ (resp., $GCF_{i+1}(D, \Sigma)$).

Thus, to obtain some rules (resp., atoms) in $GIF_1(D, \Sigma)$ (resp., $GCF_1(D, \Sigma)$), 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 $\Sigma$ of GNTGDs and a database $D$, let NLP $P = D \cup sk(\Sigma)$ and $\gamma$ its component ordering, let $P' = \bigcup_{i=0}^{\infty}$ rules$(P^i_k, i)$, 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 $\Sigma$ 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 $GCF_1(D, \Sigma)$.

Proof. We show only the “($\Rightarrow$)” direction. Let $GCF(D, \Sigma) = (V, E, \lambda)$ and $GCF_1(D, \Sigma) = (V_1, E_1(Q), \lambda_1(Q))$. Suppose that there exists a homomorphism from $Q$ to $V$, and $\beta$ is the one s.t. depth$(\beta) = \Sigma_{Q \subseteq depth(\beta(Q))}$ is minimal. Now we show that $\beta(Q) \subseteq Q_{\Sigma}$. Suppose that $\beta(Q) \not\subseteq Q_{\Sigma}$, then there exists a layer of prime blocks in $V_{\Sigma}$ 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 $\mu$ be the homomorphism mapping the subtree rooted at $r'$ to that rooted at $r$. Let $\beta' = \beta \circ \mu$, then $\beta'$ is a homomorphism mapping $Q$ to $V$. But depth$(\beta') = \Sigma_{Q \subseteq depth(\beta(Q))}$ is less than depth$(\beta)$, which contradicts to the assumption that depth$(\beta)$ is minimal. Therefore, $\beta(Q) \subseteq Q_{\Sigma}$.

**Proof.**

Lemma 6 provides us an algorithm for query answering with GNTGDs, which is in general more efficient than the approach based on the bound of guarded depth (Cali, Gottlob, and Lukasiewicz 2012, Theorem 5). Recall that the upper bound of the length of a prime block is double-exponential in $|R|$, because the number of all guarded TGDs (up to isomorphism) which are generated according to $R$ is at most double-exponential in $|R|$ in the worst case. However, given a set $\Sigma$ of GNTGDs over $R$, 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 $\Sigma$ of GDNTGDs with stratified negation, a database $D$, and a safe BNCQ $Q$, then $Q$ is true in all canonical models of $D$ and $\Sigma$ iff $Q$ is true in all GCFs of $GCF_1(D, \Sigma)$.

**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 $GIF_1(D, \Sigma)$ as in Lemma 6. But Example 4 shows this does not work.

**Example 4.** Consider two additional two rules:

\[
\begin{align*}
\text{Parent}(x, y), \neg S(y) & \rightarrow S(x) \\
\text{Parent}(x, y), S(y) & \rightarrow S(x)
\end{align*}
\]

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

Definition 5. Given two ground programs \( P \) and \( P' \) with \( P \subseteq P' \), we say \( m \in \text{SM}(P) \) is a semi-stable model of \( P' \) if there exists \( m' \in \text{SM}(P') \) s.t. \( \text{rules}_m(P) \subseteq \text{rules}_m(P') \).

We write \( \text{SSM}(P, P') \) 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 \( \Sigma \) 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 \( \text{SSM}(P^P_d, P^P) \) where \( NLP = D \cup \Sigma \).

The proof of Theorem 5 requires the following lemma.

Lemma 7. Given a finite set \( \Sigma \) of GNTGDs, a database \( D \), let \( NLP = D \cup \Sigma \) and \( \gamma \) its component ordering.

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

Definition 5. Given two ground programs \( P \) and \( P' \) with \( P \subseteq P' \), we say \( m \in \text{SM}(P) \) is a semi-stable model of \( P' \) if there exists \( m' \in \text{SM}(P') \) s.t. \( \text{rules}_m(P) \subseteq \text{rules}_m(P') \).

We write \( \text{SSM}(P, P') \) 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 \( \Sigma \) 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 \( \text{SSM}(P^P_d, P^P) \) where \( NLP = D \cup \Sigma \).

The proof of Theorem 5 requires the following lemma.

Lemma 7. Given a finite set \( \Sigma \) of GNTGDs, a database \( D \), let \( NLP = D \cup \Sigma \) and \( \gamma \) its component ordering.

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

Definition 5. Given two ground programs \( P \) and \( P' \) with \( P \subseteq P' \), we say \( m \in \text{SM}(P) \) is a semi-stable model of \( P' \) if there exists \( m' \in \text{SM}(P') \) s.t. \( \text{rules}_m(P) \subseteq \text{rules}_m(P') \).

We write \( \text{SSM}(P, P') \) 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 \( \Sigma \) 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 \( \text{SSM}(P^P_d, P^P) \) where \( NLP = D \cup \Sigma \).

The proof of Theorem 5 requires the following lemma.

Lemma 7. Given a finite set \( \Sigma \) of GNTGDs, a database \( D \), let \( NLP = D \cup \Sigma \) and \( \gamma \) its component ordering.

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

Definition 5. Given two ground programs \( P \) and \( P' \) with \( P \subseteq P' \), we say \( m \in \text{SM}(P) \) is a semi-stable model of \( P' \) if there exists \( m' \in \text{SM}(P') \) s.t. \( \text{rules}_m(P) \subseteq \text{rules}_m(P') \).

We write \( \text{SSM}(P, P') \) for the set of stable models of \( P \) that are semi-stable models of \( P' \).
the assumption that \( m_k \) is not a semi-stable model of \( P_{r_k}^{0} \). So \( m_k \) is not a semi-stable model of \( P^r \). Due to the arbitrariness of \( P^r \), \( m_k \) is not a stable model of \( P^r \), contradicting the assumption that \( m_k \) is a semi-stable model of \( P^r \). □

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 \( P_1^s \) has the answer set \( m_1 = \{ S(mary), S(mary_{f1}) \} \) and \( P_2^s \) has the answer set \( m_2 = \{ S(mary), S(mary_{f1}), S(mary_{f2}), S(mary_{f3}) \} \). Also \( \text{rules}_{m_1}(P_{1}^s) = \{ 1, 2, 3, 4, 5, 6 \} \), \( \text{rules}_{m_2}(P_{2}^s) = \{ 1, 2, 3, 4, 5, 6, 7, 8 \} \). We obtain \( P_2' \) from \( P_2^s \) by subtracting basic block \( Bb_{21} \). Then program \( P_2' \) has one stable model \( m'_{2} = \{ S(mary), S(mary_{f1}), S(mary_{f2}) \} \), and \( \text{rules}_{m_2}(P_{2}^s) = \{ 1, 2, 3, 4, 5, 6, 7, 8 \} \). We can verify that \( \text{rules}_{m_1}(P_{1}^s) \nsubseteq \text{rules}_{m_2}(P_{2}^s) \) and \( \text{rules}_{m_1}(P_{1}^s) \nsubseteq \text{rules}_{m_2}(P_{2}^s) \), and we know that \( m_1 \) is not a semi-stable model of \( P^r \) 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 \( \Sigma \) of (multi-)linear NTGDs, a database \( D \) over schema \( R \), a safe BNCQ \( Q \), then \( D \cup \Sigma \models Q \) if \( Q \) is true in all stable models of \( \Sigma \cup \{ Q \} \), where \( NLP \) \( P = D \cup sk(\Sigma) \) and \( \gamma \) is its component ordering.

**Proof.** (Sketch) By Theorem 5, it suffices to show that each stable model \( m_k \) of \( P_{r_k}^{0} \) is also a semi-stable model of \( P^r \). Given such a model \( m_k \), we can inductively construct stable models \( m_{k+j} \) of \( P_{r_k}^{0+j} \) 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 \( P^r \). □

**Experimental Evaluation**

We implemented a prototype system in Python for query answering with GNTGDs. 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: LUBM3, GeoConcepts3, and Vicodi3 which (multi-)linear TGDS: DEEP-100k(200)8 as benchmarks, which are modified by changing atoms and adding negations to make sure modified ontologies are definitely-ground and have full negation. We designed different safe BNCQs and translated into constraints.

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

**Evaluation1.** The results are summarized in Table 1. We note that \( M_{GCF} \) can only handle small queries and requires 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, \( M_{GIF} \) 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 NODE, by the Italian PRIN project HOPE, by the Free University of Bozen-Bolzano through the projects QUADRO, KGID and GeoVKG.

---

1Code and datasets. https://github.com/sysulic/GNTGDs
2LUBM. http://swat.cse.lehigh.edu/projects/lubm/
5DEEP-100k(200). https://github.com/dbunibas/chasebench
6All experiments run in 64-bit Linux on a machine with 2.10GHz Intel Xeon and 128G 1333 MHz memory.
References


