Stable Model Semantics for Guarded Existential Rules and Description Logics

Georg Gottlob[1] **André Hernich**[2] **Clemens Kupke**[3] **Thomas Lukasiewicz**[1]

[1]Department of Computer Science, University of Oxford, UK
firstname.lastname@cs.ox.ac.uk
[2]Department of Computer Science, University of Liverpool, UK
andre.hernich@liverpool.ac.uk
[3]Department of Computer and Information Sciences, University of Strathclyde, Glasgow, UK
clemens.kupke@strath.ac.uk

Abstract

We tackle a long-standing open research problem and prove the decidability of query answering under the stable model semantics for guarded existential rules, where rule bodies may contain negated atoms, and provide complexity results. The results extend to guarded Datalog$^{\pm}$ with negation, and thus provide a natural and decidable stable model semantics to description logics such as $\mathcal{ELHI}$ and *DL-Lite*$_{\mathcal{R}}$.

1 Introduction

1.1 Existential Rules, Datalog$^{\pm}$, and DLs

Existential rules have attracted much recent interest for knowledge representation and reasoning (see (Calì et al. 2010) for an overview). An example for an existential rule is

$$\textit{EmpProj(x,y), BaseLevel(x)} \rightarrow \exists z\ \textit{IsManagedBy(x,z)}, \quad (1)$$

which states that every base-level employee working in a project must be managed by somebody. Such rules are also well-known as *tuple-generating dependencies (TGDs)*, see (Beeri and Vardi 1984). In the Datalog$^{\pm}$ languages (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012), TGDs are complemented by a few other types of rules, notably by *negative constraints (NCs)*, such as *Ceo(x), BaseLevel(x)* $\rightarrow \bot$, where $\bot$ stands for *false*.[1]

Query answering based on rule sets is the following decision problem: For a database D, a set of rules Σ, and a Boolean query Q, decide whether $D \cup \Sigma \models Q$. The queries considered in this paper are unions of Boolean conjunctive queries (UBCQs). Query answering with TGDs is undecidable (Beeri and Vardi 1981), even in the case of fixed sets Σ of TGDs (Calì, Gottlob, and Kifer 2013) and singleton sets of TGDs (Baget, Leclère, and Mugnier 2010). Therefore, to obtain decidability, restrictions need to be imposed on TGDs. Semantic conditions that lead to decidability were described in (Baget, Leclère, and Mugnier 2010; Baget et al. 2009; 2011; Thomazo 2011; Calì, Gottlob, and Pieris 2012; Gottlob, Manna, and Pieris 2013). Related syntactic conditions led to the Datalog$^{\pm}$ family (Calì, Gottlob, and Lukasiewicz 2012). A class of major relevance are *guarded* TGDs (GTGDs). A TGD is guarded if its body contains an atom, called *guard*, that covers all body variables. NCs may be safely used together with guarded TGDs; their bodies are not required to contain a guard atom.

A nice and important aspect of guarded rules is that they generalize well-known description logics (DLs). All the DLs of the *DL-Lite* family in (Calvanese et al. 2007; Poggi et al. 2008) and the DL $\mathcal{ELHI}$ (Baader, Brandt, and Lutz 2005) can be embedded into guarded Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012). In particular, this holds for *DL-Lite*$_{\mathcal{R}}$, the theoretical basis of the QL profile of the Web ontology language OWL 2. To appreciate how easily DL axioms can be transformed into Datalog$^{\pm}$, note that rule (1) can be written as $\exists\text{EmpProj} \sqcap \text{BaseLevel} \sqsubseteq \exists\text{IsManagedBy}$.

1.2 The Main Problem: Stable Negation

The main goal of this paper is to add non-monotonic negation under the *stable model semantics (SMS)* (Gelfond and Lifschitz 1988) (see also (Subrahmanian 1999; Dantsin et al. 2001) for an overview of the semantic and computational properties of the SMS) to guarded rules (possibly with NCs) where body atoms may be negated. This is considered a hot open research problem, and has, to our knowledge, not been solved previously. Rules that may contain negated atoms of the form $notR(\mathbf{x})$ in their bodies, are called *normal TGDs (NTGDs)*. Guarded such rules are *guarded NTGDs* (*GNTGDs*). The standard way of defining the SMS for them is inherited from the well-known stable model semantics of normal logic programming with function symbols (Gelfond and Lifschitz 1988). In fact, we can replace existentially quantified variables in program heads by Skolem terms.

Example 1 Let $D_a = \{Person(mary)\}$ be a database and let Σ_a be the following rule set expressing that each person has at least one parent, and that each person belongs to either an odd generation or to an even generation, and that odd and even alternate between one generation and the next.

$$\begin{aligned} \textit{Person(x)} &\rightarrow \exists y\ \textit{Parent(x,y)}, \\ \textit{Parent(x,y)} &\rightarrow \textit{Person(y)}, \\ \textit{Person(x), not Even(x)} &\rightarrow \textit{Odd(x)}, \\ \textit{Person(x), not Odd(x)} &\rightarrow \textit{Even(x)}, \end{aligned}$$

[1]Often also *equality-generating dependencies (EGDs)* are used, which may express key constraints (keys) or functional dependencies. Here, we do not consider EGDs. But, as will be explained in the full paper, all results of this paper carry over to the case where rule sets are complemented with a restricted class of EGDs, called *non-conflicting EGDs* (Calì, Gottlob, and Lukasiewicz 2012).

$$\textit{Parent(x,y),Odd(x)} \rightarrow \textit{Even(y)},$$
$$\textit{Parent(x,y), Even(x)} \rightarrow \textit{Odd(y)}.$$

After skolemization, the first rule becomes:

$$\textit{Person(x)} \rightarrow \textit{Parent(x,f(x))},$$

where f is a Skolem function. This program has exactly two stable models, each consisting of an infinite chain rooted in the constant *mary*, one containing (among others) the facts *Odd(mary), Even(f(mary)), Odd(f(f(mary)))*, and so on, the other containing the fact *Even(mary), Odd(f(mary)), Even(f(f(mary)))*. Let Q_a be the conjunctive query $\exists x, y, z$ *(Parent(x,y), Parent(y,z), Odd(x), Odd(z))*, and let Q'_a be the query $\exists x, y$ *(Parent(x,y), Odd(x), Odd(y))*. Then, under the SMS, $(D_a \cup \Sigma_a) \models Q_a$ but $(D_a \cup \Sigma_a) \not\models Q'_a$. ∎

Note that further examples are given in Section 7. It is easy to exhibit rule sets Σ that generate infinitely many infinite stable models, as illustrated by the following example.

Example 2 Let $D_b = D_a = \{Person(mary)\}$, and let Σ_b be obtained by taking just the first four rules of the rule set Σ_a of Example 1. Then, there is an uncountably infinite number of stable models of $D_b \cup \Sigma_b$, one for each possibility of having for each Skolem term $f^i(mary)$ either *Even(f^i(mary))* or *Odd(f^i(mary))*, where $1 \leqslant i \leqslant \infty$. For the query Q_a of Example 1, note that $(D_b \cup \Sigma_b) \not\models Q_a$. ∎

It seems that the interplay of nonmonotonic negation with these two types of infinity made it hard to understand how the SMS for GNTGDs could be brought under control. The main questions that we pose and tackle are thus: Is query answering under the stable model semantics for GNTGDs decidable? If so, what is the complexity of this problem and what kind of algorithms are appropriate for it?

1.3 Results

An insightful initial observation while addressing the above decidability problem was that if a guarded rule set with negated atoms admits a stable model, then it admits a tree-shaped one. In fact, each such stable model is a model of some Gelfond-Lifschitz transform of the ground version *ground*(Σ) of the original rule set. For the latter (which are guarded positive existential rule sets), we know that there are universal tree-shaped models (Calì, Gottlob, and Kifer 2013) that arise via the well-known chase procedure (Maier, Mendelzon, and Sagiv 1979; Beeri and Vardi 1984). We could thus encode query answering under the SMS for existential Datalog$^{\pm}$ in monadic second-order logic (MSO), making stable models correspond to binary trees. Decidability of this problem then follows from Rabin's Theorem (1969), by which satisfiability of MSO is decidable over the class of binary trees. This is very briefly described in Section 3.3 and fully proven in the extended paper (Gottlob et al. 2014). Related ideas for deciding other guarded logics or DLs via MSO can be found in (Eiter and Simkus 2010) and in (Motik, Horrocks, and Sattler 2009). We thus have shown:

Theorem. There is an algorithm that, given a database D, a finite set Σ of GNTGDs, and a BCQ Q, decides whether Q is true in all stable models for D and Σ.

Unfortunately, the MSO formula constructed for proving the above result does not give us a tight complexity upper bound for query evaluation with GNTGDs. To arrive at a matching bound, we pursued the following different ideas. Recall that finding a stable model essentially involves first guessing a model and then checking that this model is indeed the least fixpoint of the Gelfond-Lifschitz transform Σ^M of the original rule set Σ relative to the guessed model M. Given that query answering under the classical first-order (FO) semantics for guarded disjunctive TGDs (GDTGDs) has been recently defined and studied (Alviano et al. 2012; Gottlob et al. 2012b; Bourhis, Morak, and Pieris 2013), why not simulating the stable model semantics of GNTGDs by GDTGDs under the FO semantics? For the guessing phase, this turned out to be possible, as guessing the truth or falsehood of an atom can be encoded as a classical disjunction. But the checking phase requires computing the complement of Σ^M. This is an inherently nonmonotonic task not feasible in FO. But only one level of nonmotonic inference is needed. It would thus be sufficient to extend DGTDs under FO semantics by *stratified negation*, yielding the formalism of stratified sets of guarded disjunctive normal TGDs (GDNTGDs). In this setting, atom entailment within each stratum is the classical monotonic FO-entailment, but each stratum may access the lower strata nonmonotonically via the *not* operator, i.e., for each atom $\mathbf{a}$ of the Herbrand base, each stratum may assume that *not* $\mathbf{a}$ is true if the predicate of $\mathbf{a}$ belongs to a lower stratum, but $\mathbf{a}$ cannot be derived there.

We thus aimed at proving the decidability and determining the complexity of query answering with stratified GDNTGDs. This turned out to be much harder a challenge than the same problem for GTGDs (Calì, Gottlob, and Lukasiewicz 2012) or stratified GNTGDs. With GTGDs (or stratified GNTGDs), there is only a single, possibly infinite canonical model, and each query UBCQ Q is true in this model iff it is true in a finite part of this model. This finite part consists, for each stratum, of the atoms generated by the chase in that stratum that do not exceed a certain derivation depth. Unfortunately, this is no longer true for models generated by GDTGDs, and thus for stratified GDNTGDs. For a fixed rule set Σ and a fixed query Q, and any arbitrary positive integer d, in general, there may exist "tardive" models M such that $M \models Q$ but $M_d \not\models Q$, where M_d contains all atoms of M whose chase derivation depth is at most d. Not too surprisingly, tardive models may also arise with rules sets Σ of original formalism, namely, GNTGDs under the SMS. To illustrate this, consider the setting of Example 2, and let Q be the query $\exists x$*Even(x)*. Let d be an arbitrary positive integer. There are stable models M such that, for all $1 \leqslant i \leqslant d$, *Odd*$(f^i(mary))$ and *Even*(f^{d+1}). For such models M, it does not suffice to consider the initial part M_d containing all facts of M of chase derivation depth $\leqslant d$, for Q would be false in this initial part, while true in M.

Fortunately, we can cope with tardive models for a query Q due to the following observation. Whenever for a stratified GDNTGD set Σ and a query Q, a tardive model exists in which the query is not answered within a certain (doubly exponential) derivation depth, then there must exist *another model* in which Q is generally false, and thus Q must be

overall answered *false*, because Q is false in some model. It is thus still possible to reconduct query answering to a finitary setting. This insight led us to develop alternating algorithms for query answering under stratified sets of GDNTDGs, which give rise to the following results, where $\models_{\mathsf{strat}}$ is entailment under the described stratified semantics:

Theorem. Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of GDNTGDs on $\mathcal{R}$, and Q a union of CQs over $\mathcal{R}$. Deciding $D \cup \Sigma \models_{\mathsf{strat}} Q$ has the following complexity:

1. 2-EXPTIME-complete in general;
2. EXPTIME-complete in case the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed;
3. coNP-complete in case $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.

In Section 6, we present the desired polynomial-time translation from GNTGDs under the SMS to GDNTGDs under FO semantics with stratified negation. This solves the main problem attacked in this paper. From this translation, and from easily obtained lower bounds, for query-answering for GNTGDs, we get exactly the same complexity bounds as those in the above theorem. This solves our main complexity problem. Moreover, all the above decidability and complexity results also hold in case negative constraints are added to the rule sets. (This is trivial: To simulate NCs, it suffices to add the body of each NC as a disjunct to the UCQ.)

In Section 7, we finally apply our results to DLs. We immediately obtain a stable model semantics for these DLs, as well as decidability results and complexity upper bounds. In particular, this provides a stable model semantics for the well-known DLs $\mathcal{ELHI}$ and *DL-Lite*$_{\mathcal{R}}$.

In the sequel, we use the letters "G", "D", and "N" in front of TGDs to abbreviate that the TGDs are "guarded", "disjunctive", and "normal", respectively. Other properties of TGDs are written out in textual form whenever necessary. Note that proofs of all results are given in the extended version of this paper (Gottlob et al. 2014).

1.4 Related Work on Rules with Negation

We now discuss a number of previous approaches to extend guarded TGDs with negation.

Stratified Negation. Stratified negation for guarded Datalog$^{\pm}$, and thus automatically for a number of important DLs was introduced in (Calì, Gottlob, and Lukasiewicz 2012) and further extended to the more expressive formalism of *weakly guarded Datalog*$^{\pm}$ in (Arenas, Gottlob, and Pieris 2014). These papers show that stratified negation is very well-behaved in the sense that its addition does not endanger decidability, and actually does not augment the complexity of reasoning and query answering. But this semantics does not apply to the (unstratified) rules in Examples 1 and 2.

Well-Founded Negation. Negation under the well-founded semantics (WFS) for guarded Datalog$^{\pm}$ and covered DLs was studied in (Hernich et al. 2013; Gottlob et al. 2012a) for two variants of the WFS. The version of (Hernich et al. 2013) studies the so-called *standard WFS*, which extends the well-known WFS for logic programming with function symbols, where it is assumed (as here) that different Skolem terms are not unifiable, which means that the *unique name assumption (UNA)* is applied to Skolem terms. The second variant, called *equality-friendly WFS* (Gottlob et al. 2012a) does not use the UNA. Note that both variants would be unsatisfactory for Examples 1 and 2, as they would simply leave the predicates *Odd* and *Even* undefined.

Stable Model Negation. In (Magka, Krötzsch, and Horrocks 2013), new acyclicity and stratification conditions are presented for existential rules with negation in rule bodies, which identify classes of rule sets that have finite and/or unique stable models, and constraints are added on the input facts to further extend these classes. Our work here, in contrast, is not restricted to finite and/or unique stable models, and thus much more general. The only syntactic restriction necessary here is guardedness (or the weaker weakly guardedness). The $\mathbb{FDNC}$ programs in (Eiter and Simkus 2010) combine nonmonotonic negation and rules, allowing also the use of function symbols. Decidability is obtained by restricting the structure of rules to one of seven predefined forms.

Hybrid Approaches. Less closely related approaches are loosely and tightly coupled dl-programs (Eiter et al. 2008; Lukasiewicz 2010), as well as the hybrid MKNF knowledge bases (Motik and Rosati 2010). More precisely, the former loosely and tightly, respectively, combine a description logic knowledge base L and a logic program P. Rule bodies in P may contain queries to L, which may also contain facts as additional input to L, in the loosely coupled case, while concepts and roles from L are used as predicates in P in the tightly coupled case. Decidability is based on the finiteness of stable models. The hybrid MKNF knowledge bases allow for querying a description logic knowledge base L via the operators $\mathbf{K}$ and $\mathbf{not}$. Decidability is obtained via the so-called DL-safety condition, which makes the rules applicable only to explicitly known individuals. In our approach here, in contrast, stable models may be infinite, and no restrictive DL-safety of rules is assumed.

In summary, none of the above approaches allows one to use the stable model semantics for the general class of guarded normal TGDs.

2 Preliminaries

We now briefly recall some basics on Datalog$^{\pm}$ (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012), namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple-generating dependencies (TGDs), and negative constraints. For equality-generating dependencies (EGDs), in particular, non-conflicting keys, we refer to (Calì, Gottlob, and Lukasiewicz 2012).

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries and constraints). We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *relation names* (or *predicate symbols*, or simply *predicates*).

A *position* $P[i]$ identifies the i-th argument of a predicate P. A *term* t is a constant or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms. A conjunction of atoms is often identified with the set of all its atoms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. We denote by $dom(D)$ the set of all elements of Δ that occur in D. 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 the variables $\mathbf{X}$ and $\mathbf{Y}$ (and eventually constants). Note that $\Phi(\mathbf{X}, \mathbf{Y})$ may also contain equalities but no inequalities. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$. We often write a BCQ as the set of all its atoms, having constants and variables as arguments, and omitting the quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \mathcal{V} \to \Delta \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, and (ii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \to \Delta$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q() = \exists \mathbf{Y}\, \Phi(\mathbf{Y})$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \varnothing$, i.e., there is a homomorphism $\mu\colon \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{Y})) \subseteq D$.

Tuple-Generating Dependencies (TGDs). Tuple-generating dependencies (TGDs) describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads; their syntax and semantics are as follows. Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there is a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, then h is extendable to a homomorphism h' that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. As TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard* of σ.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \varnothing$.

Negative Constraints. Another crucial ingredient of Datalog$^{\pm}$ for ontological modeling are *negative constraints* (or simply *constraints*), which are first-order formulas of the form $\forall \mathbf{X}\ \Phi(\mathbf{X}) \to \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded), called its *body*.

We usually omit the universal quantifiers and implicitly assume all sets of dependencies/constraints to be finite here.

3 Stable Models for Normal Datalog$^{\pm}$

In this section, we recall normal TGDs and BCQs (Calì, Gottlob, and Lukasiewicz 2012), as well as stable models of normal (logic) programs (Gelfond and Lifschitz 1988), and we define the stable model semantics of guarded normal TGDs.

3.1 Normal TGDs and BCQs

Normal TGDs may also contain (default-)negated atoms in their bodies: Given a relational schema $\mathcal{R}$, a *normal TGD (NTGD)* σ has the form $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathcal{R}$, and $\Psi(\mathbf{X}, \mathbf{Z})$ is a conjunction of atoms over $\mathcal{R}$. It is also abbreviated as $\Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. We denote by $head(\sigma)$ the atom in the head of σ, and by $body^+(\sigma)$ and $body^-(\sigma)$ the sets of all positive and negative ("$\neg$"-free) atoms in the body of σ, respectively. We say σ is *guarded* iff it contains a positive body atom, denoted $guard(\sigma)$, that contains all universally quantified variables of σ. W.l.o.g., every NTGD has a single atom in its head, and constants in the body of guarded σ occur only in guards. We say σ is *linear* iff σ is guarded and has exactly one positive atom in its body. As for the semantics, an NTGD σ is *satisfied* in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h for all the variables and constants in the body of σ that maps (i) all atoms of $body^+(\sigma)$ to atoms of D and (ii) no atom of $body^-(\sigma)$ to atoms of D, then there exists an extension h' of h that maps all atoms of $head(\sigma)$ to atoms of D.

We next add negation to BCQs. A *normal Boolean conjunctive query (NBCQ)* Q is an existentially closed conjunction of atoms and negated atoms of the form

$$\exists \mathbf{X}\, p_1(\mathbf{X}) \wedge \cdots \wedge p_m(\mathbf{X}) \wedge \neg p_{m+1}(\mathbf{X}) \wedge \cdots \wedge \neg p_{m+n}(\mathbf{X}),$$

where $m \geqslant 1$, $n \geqslant 0$, and the variables of the p_i's are among $\mathbf{X}$. Let $Q^+ = \{p_1(\mathbf{X}), \ldots, p_m(\mathbf{X})\}$ and $Q^- = \{p_{m+1}(\mathbf{X}), \ldots, p_{m+n}(\mathbf{X})\}$. That is, Q^+ (resp., Q^-) is the set of all positive (resp., negative) ("$\neg$"-free) atoms of Q. We say Q is *safe* iff every variable in a negative atom in Q also occurs in a positive atom in Q. We say Q is *covered* iff for every negative atom α in Q, there is a positive atom β in Q such that every argument in α occurs in β. Note that the coveredness of Q implies also the safeness of Q, but not vice versa. In the sequel, w.l.o.g., NBCQs contain no constants. Note that disjunctive normal non-Boolean CQs over finite databases can be reduced to constant-free NBCQs, by first reducing them to NBCQs with constants, and then moving their constants into the TGDs by introducing new predicate symbols.

3.2 Stable Models of Normal Programs

A *normal rule* (or simply *rule*) r is of the form

$$\alpha \leftarrow \beta_1, \ldots, \beta_n, not\ \beta_{n+1}, \ldots, not\ \beta_{n+m}\,, \qquad (1)$$

where $\alpha, \beta_1, \ldots, \beta_{n+m}$ are atoms and $k, m, n \geqslant 0$. We call α the *head* of r, denoted *head*(r), while the conjunction

$\beta_1, \ldots, \beta_n, not\ \beta_{n+1}, \ldots, not\ \beta_{n+m}$ is its *body*. We define $body(r) = body^+(r) \cup body^-(r)$, where $body^+(r) = \{\beta_1, \ldots, \beta_n\}$ and $body^-(r) = \{\beta_{n+1}, \ldots, \beta_{n+m}\}$. We say r is *positive* iff $m = 0$. We say r is a *fact* iff $m = n = 0$. A *normal* (resp., *positive*) *program* P is a finite set of normal (resp., positive) rules.

Let P be a normal program. We denote by HU_P and HB_P the *Herbrand universe* and the *Herbrand base* for P, respectively. A *ground instance* of a rule $r \in P$ is obtained from r by replacing each variable in r by an element from HU_P. We denote by $ground(r)$ (resp., $ground(P)$) the set of all ground instances of r (resp., rules in P). An *interpretation* I for P is a subset of HB_P. Such I is a *model* of a ground rule r iff $\alpha \in I$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \varnothing$. We say I is a *model* of a normal program P iff I is a model of every $r \in ground(P)$.

The *Gelfond-Lifschitz reduct* of a normal program P relative to $I \subseteq HB_P$, denoted P^I, is the (possibly infinite) ground positive program obtained from $ground(P)$ by

1. deleting every rule r such that $B^-(r) \cap I \neq \varnothing$, and
2. deleting the negative body from each remaining rule.

Since P^I is positive, it has a unique minimal model. A *stable model* of a normal program P is an interpretation $I \subseteq HB_P$ such that I is the minimal model of P^I. Note that every stable model of P is also a minimal model of P.

3.3 Stable Models for Guarded Normal TGDs

Let Σ be a finite set of guarded NTGDs, and let D be a database over the same schema as Σ. Given an NTGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, the *functional transformation* of σ, denoted σ^f, is the normal rule $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \Psi(\mathbf{X}, \mathbf{f}_\sigma(\mathbf{X}, \mathbf{Y}))$, where $\mathbf{f}_\sigma$ is a vector of function symbols $f_{\sigma,Z}$ for σ, one for every variable Z in $\mathbf{Z}$. Given a set Σ of NTGDs, the *functional transformation* of Σ, denoted Σ^f, is obtained from Σ by replacing each TGD σ in Σ by σ^f. Note that the functional transformation of a finite set of guarded TGDs is a positive program.

Definition 1 Let Σ be a finite set of guarded NTGDs, and let D be a database over the same schema as Σ. Then, a *stable model* for D and Σ is a stable model of the normal program $P = D \cup \Sigma^f$, having D as a collection of ground facts.

Non-constant terms (i.e., those involving function symbols) that occur in stable models for D and Σ are henceforth considered as *nulls*. Let Σ^+ be the finite set of TGDs obtained from Σ by dropping all negative literals from the rules' bodies. Recall the definition of the guarded chase forest for D and Σ^+ from (Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012). The *upper guarded chase forest* for D and Σ, denoted *g-chase*(D, Σ), is the guarded chase forest for D and Σ^+. The following lemma shows that the set of atoms occurring in *g-chase*(D, Σ) provides an overapproximation for any stable model for D and Σ. From this, we infer below that query answering under guarded normal Datalog$^\pm$ with the stable models semantics is decidable. For the proof, we refer to (Gottlob et al. 2014).

Lemma 2 *Let Σ be a finite set of guarded NTGDs, and let D be a database over the same schema as Σ. If S is a stable model for D and Σ, then all atoms of S occur as the label of some node in g-chase(D, Σ).*

Decidability of query answering relative to the stable model semantics can be obtained as a consequence of Rabin's Theorem (1969) via an encoding into monadic second-order logic over the infinite m-ary tree (for suitable $m \in \mathbb{N}$).

Theorem 3 *There is an algorithm that, given a database D, a finite set Σ of guarded NTGDs, and a covered UNBCQ Q, decides whether Q is true in all stable models for D and Σ.*

Proof (Sketch). Given D, Σ, and Q, compute an MSO sentence $\varphi_{D,\Sigma,Q}$ such that Q is true in all stable models for D and Σ iff $T \models \varphi_{D,\Sigma,Q}$. The basic idea for constructing the MSO sentence is to label a subset of the nodes of the full m-ary tree T with a *type* (α, N) consisting of an atom α and a set N of atoms whose arguments occur in α, and then to verify that the subtree T' of T induced by all labelled nodes corresponds to a stable model. Due to space limitations, and because we obtain stricter complexity bounds for query answering in the coming sections, we omit the details of the MSO-translation, and refer the reader to the extended version of this paper (Gottlob et al. 2014). □

4 Disjunctive Rules with Stratified Negation

Disjunctive Normal TGDs and BCQs. Given a relational schema $\mathcal{R}$, a *disjunctive normal TGD (DNTGD)* σ has the form $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathcal{R}$, and $\Psi(\mathbf{X}, \mathbf{Z})$ is a disjunction of atoms over $\mathcal{R}$. We denote by $head(\sigma)$ the set of atoms in the head of σ. All other notions like $body^+(\sigma)$, $body^-(\sigma)$, guardedness, and $guard(\sigma)$ are defined as for NTGDs. A DNTGD σ is *satisfied* in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h for all the variables and constants in the body of σ that maps (i) all atoms of $body^+(\sigma)$ to atoms of D and (ii) no atom of $body^-(\sigma)$ to atoms of D, then there exists an extension h' of h that maps some atoms of $head(\sigma)$ to atoms of D.

A *union of normal Boolean conjunctive queries (UNBCQ)* Q is a disjunction of $k > 0$ NBCQs Q_i. It is *safe* (resp., *covered*) iff all its NBCQs Q_i are safe (resp., covered).

We assume the reader to be familiar with the chase procedure. The *disjunctive normal chase* is an extension of the chase. It differs to the chase in that an application of a DNTGD σ to a database D via a homomorphism $h\colon body^+(\sigma) \rightarrow D$ (that maps atoms $\beta \in body^-(\sigma)$ to non-atoms in D) generates more than one possible successor database—one database $D \cup \{h(\psi)\}$ for each disjunct $\psi \in head(\sigma)$. The set of all databases obtained using this version of the chase is denoted by $chase(D, \Sigma)$. Note that DNTGDs may be applied several times with the same homomorphism, to introduce several of the head atoms.

Canonical Model Semantics. A *stratification* of a set Σ of disjunctive normal TGDs on $\mathcal{R}$ is a mapping $\mu\colon \mathcal{R} \rightarrow \{0, 1, \ldots, k\}$ such that for all $\sigma \in \Sigma$:

- for all positive literals $R(\bar{u})$ in the body of σ, and for all atoms $R'(\bar{u}')$ in the head of σ, we have $\mu(R) \leqslant \mu(R')$;

- for all negative literals $\neg R(\bar{u})$ in the body of σ, and for all atoms $R'(\bar{u}')$ in the head of σ, we have $\mu(R) < \mu(R')$.

We call k the *length* of μ. For each $i \in \{0, 1, \ldots, k\}$, let Σ_i be the set of all $\sigma \in \Sigma$ such that i is the minimum of $\mu(R)$, where R ranges over all the predicates in the head of σ. We say that Σ is *stratified* if there is a stratification of Σ.

Definition 4 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a set of DNTGDs on $\mathcal{R}$, and Q a UNBCQ over $\mathcal{R}$. Suppose that Σ has a stratification of length k. Define:*

- $\mathcal{S}^0(D, \Sigma) := \{D\}$;
- $\mathcal{S}^{i+1}(D, \Sigma) := \bigcup_{S \in \mathcal{S}^i(D,\Sigma)} chase^S(S, \Sigma_i)$, *if* $i \leqslant k$;
- $\mathcal{S}(D, \Sigma) := \mathcal{S}^{k+1}(D, \Sigma)$.

We write $D \cup \Sigma \models_{\mathsf{strat}} Q$ iff $S \models Q$ for all $S \in \mathcal{S}(D, \Sigma)$.

5 Query Answering under Disjunctive Rules with Stratified Negation

This section presents our results on answering queries over disjunctive guarded Datalog$^{\pm}$ programs with stratified negation. Our main result is:

Theorem 5 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of guarded DNTGDs on $\mathcal{R}$, and Q a covered UNBCQ over $\mathcal{R}$. Deciding $D \cup \Sigma \models_{\mathsf{strat}} Q$ has the following complexity:*

1. 2-EXPTIME-*complete in general;*
2. EXPTIME-*complete in case the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed;*
3. coNP-*complete in case $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.*

Thus, extending guarded Datalog$^{\pm}$ by the feature of disjunction and stratified negation—even allowing negative literals in the query—does not change the complexity of query answering in the general and in the bounded arity case, while it increases the data complexity. The data complexity is coNP already for the extension of guarded Datalog$^{\pm}$ by disjunction (Bourhis, Morak, and Pieris 2013), which proves coNP-hardness in Theorem 5. The other hardness results of Theorem 5 follow from answering BCQs under guarded TGDs being 2-EXPTIME-hard in general and EXPTIME-hard for bounded arities (Calì, Gottlob, and Kifer 2013).

For the upper bounds, we devise an algorithm to answer UNBCQs over stratified sets of guarded DNTGDs. The algorithm extends the QCHECK algorithm in (Calì, Gottlob, and Kifer 2013), but requires significantly new ideas. Our exposition follows the one of QCHECK in that paper in that we first explain how to decide entailment or non-entailment of atoms, and then describe the general algorithm.

5.1 Checking Entailment of Ground Atoms

One of the first steps of our algorithm is to decide if certain ground atoms are true in some "stratified model" or false in all "stratified models" $S \in \mathcal{S}(D, \Sigma)$. Here, an atom α is *ground* (relative to a database D, which, however, will always be clear from the context) iff $dom(\alpha) \subseteq dom(D)$.

Proposition 6 below states that (non-)entailment of ground atoms can be checked within the desired complexity bounds. For stating and proving it, we need the following definitions. For atoms α, let $dom(\alpha) := dom(\{\alpha\})$. The databases D_1 and D_2 are *consistent* iff, for all α with $dom(\alpha) \subseteq dom(D_1) \cap dom(D_2)$, we have that $\alpha \in D_1$ iff $\alpha \in D_2$. We say that D_2 *embeds* D_1 iff $D_1 \subseteq D_2$, and D_1 and D_2 are consistent. Let $\mathcal{S}^*(D, \Sigma)$ be the set of all $S \in \mathcal{S}(D, \Sigma)$ that embed D.

Proposition 6 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of guarded DNTGDs on $\mathcal{R}$, and α a ground atom over $\mathcal{R}$. Deciding whether some model in $\mathcal{S}^*(D, \Sigma)$ contains α, or if none of the models in $\mathcal{S}^*(D, \Sigma)$ contains α is in:*

- 2-EXPTIME *in general;*
- EXPTIME *in case the maximum arity w of a predicate in $\mathcal{R}$ is bounded;*
- PTIME *in case w and $|\mathcal{R}|$ are bounded.*

Proof (Sketch). We extend the ACHECK algorithm from (Calì, Gottlob, and Kifer 2013). ACHECK checks if an atom (atomic BCQ, in general) α is entailed by a database D and a set Σ of guarded TGDs. To do this, it guesses a sequence $\alpha_1, \ldots, \alpha_n$ of atoms such that $\alpha_1 \in D$, $\alpha_n = \alpha$, and each α_i, $i > 1$, can be obtained by a TGD σ_i in Σ whose guard is matched to α_{i-1}. To ensure that the *side atoms* β of each σ_i (i.e., the non-guard atoms in the body of σ_i required to "fire" σ_i) can be derived as well, it launches side computations that check (similarly as for α) if there is a similar path for β etc.

Our algorithm, called $\mathsf{ACHECK}_{\vee,\mathsf{strat}}$, generalizes this approach. The main difficulty is that under a stratified set Σ of guarded DNTGDs, there is not a single model (namely, the chase model) to explore, but many models $S \in \mathcal{S}^*(D, \Sigma)$. However, every such model can be constructed on the fly by applying the DNTGDs in Σ, which is enough to search for a model that contains α, or—by duality—to check that no model contains α. To be a bit more precise, each disjunct in the head of a DNTGD represents a possible way of extending the part of a model derived so far. Thus, when faced with a disjunction in the head of a DNTGD, it suffices to guess the disjunct β that leads to a model S containing α (and the set of all atoms in such an S with $dom(\gamma) \subseteq dom(\beta)$); this guess is verified by appropriate subcomputations. Negative literals in the bodies of DNTGDs pose no problem, since negation is stratified in Σ. We only have to ensure that the stratification is respected when applying the DNTGDs. For details, see the extended paper (Gottlob et al. 2014).

As in (Calì, Gottlob, and Kifer 2013), it is possible to reduce the space required to represent the configurations of $\mathsf{ACHECK}_{\vee,\mathsf{strat}}$ to $|\mathcal{R}| \cdot 2^{O(w)} \cdot \log |D|$, where w is the maximum arity of a predicate in $\mathcal{R}$, via *canonization*. As alternating logarithmic (resp., polynomial, exponential) space corresponds to polynomial (resp., exponential, doubly exponential) time, this yields the complexity bounds. □

5.2 Main Algorithm

We now complete the proof of Theorem 5 by proving the corresponding upper bounds.

Proposition 7 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a stratified set of guarded DNTGDs on $\mathcal{R}$, and Q a covered UNBCQ over $\mathcal{R}$. Deciding $D \cup \Sigma \models_{\mathsf{strat}} Q$ can be done within the upper bounds indicated in Theorem 5.*

Proof (Sketch). First, w.l.o.g., Q is a union of BCQs. Indeed, if $\neg\beta = \neg R(x_1, \ldots, x_k)$ occurs in Q, and α is the positive literal in Q covering β, then $\neg\beta$ can be simulated by $\alpha \wedge \neg\beta \rightarrow \bar{R}(x_1, \ldots, x_k)$, where $\bar{R}$ is a fresh predicate. This increases the length of a stratification of Σ only by one. In Q, we replace $\neg\beta$ by the atom $\bar{R}(x_1, \ldots, x_k)$.

To obtain the complexity bounds, we extend the QCHECK algorithm in (Calì, Gottlob, and Kifer 2013). Handling stratified DNTGDs requires significantly new ideas, though. The original QCHECK algorithm checks if a BCQ is entailed by a database and a set of (weakly) guarded TGDs. It starts by computing the extension $\hat{D}$ of D with all ground atoms entailed by D and the TGDs. Next, it guesses a *squid decomposition* δ of Q. Intuitively, a squid decomposition of Q describes how Q may be mapped to a model S in $\mathcal{S}(D, \Sigma)$. For our purposes, a squid decomposition of Q may be defined as a tuple (Q^+, h, H, T, V), where:

- Q^+ is a *cover* of Q (a BCQ that contains all atoms of Q and at most $|Q|$ other atoms);
- $h\colon var(Q^+) \rightarrow var(Q^+)$,
- $H \subseteq Q^+$ and $T := Q^+ \setminus H$ are the *head* and *tentacles*,
- and T is *acyclic* (i.e., T has a join tree).

It can be shown (Calì, Gottlob, and Kifer 2013) that Q is true in the chase model iff there is a squid decomposition whose head can be mapped to the ground part of the database, and whose tentacles can be mapped to the non-ground part. Thus, the QCHECK algorithm also guesses a homomorphism μ from the head of δ to $\hat{D}$. In the rest of the computation, it checks that μ can be extended to a homomorphism from the tentacles to the result of the chase with D and the TGDs. In our case, we do not have a single "chase model", but many such models $S \in \mathcal{S}(D, \Sigma)$ into which we have to map Q. In particular, there are many extensions $\hat{D}$ of D with ground atoms such that some $S \in \mathcal{S}(D, \Sigma)$ embeds $\hat{D}$. Our algorithm thus starts by universally branching over all such extensions $\hat{D}$ of D with ground atoms, and checks that there is a model $S \in \mathcal{S}(D, \Sigma)$ that embeds $\hat{D}$. The latter can be done within the desired complexity bounds by using the algorithm as guaranteed by Proposition 6. In a parallel subcomputation, the algorithm then checks that Q is true in all models $S \in \mathcal{S}^*(\hat{D}, \Sigma)$. Clearly, this algorithm decides $D \cup \Sigma \models_{\mathsf{strat}} Q$ (i.e., if Q is true in all $S \in \mathcal{S}(D, \Sigma)$).

It remains to explain how truth of Q in all $S \in \mathcal{S}^*(\hat{D}, \Sigma)$ can be decided within the desired complexity bounds. To this end, we extend the TCHECK algorithm in (Calì, Gottlob, and Kifer 2013). We highlight only the main changes to TCHECK; (Gottlob et al. 2014) contains a more detailed description. For details regarding TCHECK, see the proof of Theorem 5.9 in (Calì, Gottlob, and Kifer 2013).

The idea of TCHECK is as follows. Given a squid decomposition δ (for which we have found a homomorphism h from its head to the ground part of the chase in the initialization phase of QCHECK), a maximal subtree t of the tentacles of δ, the root v of t, and the assignment h restricted to the *exported variables* of v (i.e., those variables that occur in v but also outside the subtree below v), TCHECK tries to extend the assignment to a homomorphism from the subtree rooted at v to the chase model by deriving an atom where v can be matched to, then deriving the atoms where the children of v can be matched to, and so on. In our case, each model $S \in \mathcal{S}(\hat{D}, \Sigma)$ might satisfy Q via a different embedding or squid decomposition of Q. Thus, rather than maintaining information about the progress of extending the initial homomorphism to a single subtree of a squid decomposition's tentacles throughout the computation, we store information about all possible squid decompositions of Q, and the parts of the squid decompositions that still need to be matched (i.e., have not been mapped into the part of the model constructed so far). Once we are in a state where one squid decomposition is complete (i.e., no part of it is left to be checked), we accept. Thus, if all computation branches accept, we have successfully checked that each $S \in \mathcal{S}^*(\hat{D}, \Sigma)$ satisfies Q. □

6 Simulating Normal Datalog$^{\pm}$ under Stable Models with Stratified Disjunctive Rules

The following theorem shows that NBCQ answering over a database and a finite set of guarded NTGDs under the stable model semantics can be translated in polynomial time into NBCQ answering over the same database and a finite set of guarded DNTGDs with stratified negation.

Theorem 8 *Let D be a database for some schema $\mathcal{R}$, let Σ be a finite set of guarded NTGDs over $\mathcal{R}$, and let Q be an NBCQ over $\mathcal{R}$. Then, Σ and Q can be translated in polynomial time into a stratified finite set of guarded DNTGDs Σ' and an NBCQ Q' such that Q is true in all stable models of D and Σ iff Q' is true in all canonical models of D and Σ'.*

Proof. We define the stratified finite set of guarded DNTGDs $\Sigma' = \Sigma'_1 \cup \Sigma'_2 \cup \Sigma'_3 \cup \Sigma'_4$ and the NBCQ Q' as follows:
• Σ'_1 generates the potential atoms that may occur in a stable model for D and Σ. To this end, it "executes" the program $P^+ = (D \cup \Sigma^f)^+$. For each $R \in \mathcal{R}$, let R^* be a fresh predicate of the same arity as R. Then, Σ'_1 consists of all TGDs that are obtained from an NTGD in Σ by dropping all negative literals from its body, and replacing each predicate R by R^*. Furthermore, for each k-ary predicate $R \in \mathcal{R}$, the set Σ'_1 contains the TGD $R(x_1, \ldots, x_k) \rightarrow R^*(x_1, \ldots, x_k)$.
• Σ'_2 "guesses" a partition of the set of all atoms derivable from P^+ into two parts. For each predicate $R \in \mathcal{R}$, let R^+ and R^- be fresh predicates of the same arity as R. The idea is that R^+ and R^--atoms are those belonging to the first and second part of the partition, respectively. To "guess" the partition, for each k-ary predicate $R \in \mathcal{R}$, we add the following disjunctive TGD and negative constraint to Σ'_2:

$$R^*(x_1, \ldots, x_k) \rightarrow R^+(x_1, \ldots, x_k) \vee R^-(x_1, \ldots, x_k),$$
$$R^+(x_1, \ldots, x_k) \wedge R^-(x_1, \ldots, x_k) \rightarrow \bot.$$

• Σ'_3 generates all consequences of the program P^S, where S is the set of all the R^+-atoms (with R^+ renamed to R). More precisely, for each predicate $R \in \mathcal{R}$, let $\hat{R}$ be a predicate of the same arity as R. Consider an NTGD in Σ:

$$R_1(\mathbf{x}_1) \wedge \cdots \wedge R_k(\mathbf{x}_k) \wedge \neg S_1(\mathbf{y}_1) \wedge \cdots \wedge \neg S_\ell(\mathbf{y}_\ell) \rightarrow \exists \mathbf{z}\, T(\mathbf{u}).$$

• Then, Σ'_3 contains the following NTGD:

$$\hat{R}_1(\mathbf{x}_1) \wedge \cdots \wedge \hat{R}_k(\mathbf{x}_k) \wedge S_1^-(\mathbf{y}_1) \wedge \cdots \wedge S_\ell^-(\mathbf{y}_\ell) \rightarrow \exists \mathbf{z}\, \hat{T}(\mathbf{u}).$$

Furthermore, for each k-ary predicate $R \in \mathcal{R}$, the set Σ'_3 contains the TGD $R(x_1, \ldots, x_k) \rightarrow \hat{R}(x_1, \ldots, x_k)$.

• Σ'_4 verifies that the consequences of P^S (represented by all the $\hat{R}$-atoms) coincide with the set S (represented by all the R^+-atoms). To this end, for each k-ary predicate $R \in \mathcal{R}$, it contains the following negative constraints:

$$\hat{R}(x_1, \ldots, x_k) \wedge R^-(x_1, \ldots, x_k) \rightarrow \bot \qquad (\text{“}\hat{R} \subseteq R^+\text{”})$$
$$R^+(x_1, \ldots, x_k) \wedge \neg \hat{R}(x_1, \ldots, x_k) \rightarrow \bot \qquad (\text{“}R^+ \subseteq \hat{R}\text{”}).$$

• Finally, Q' is obtained from Q by replacing every positive (resp., negative) literal $R(\mathbf{x})$ (resp., $\neg R(\mathbf{x})$) by $R^+(\mathbf{x})$ (resp., $\neg R^+(\mathbf{x})$).

We now show that Q is false in some stable model for D and Σ iff Q' is false in some canonical model for D and Σ'. We here consider only the case that Q is a BCQ; the line of argumentation can easily be generalized to NBCQs.

“$\Rightarrow$” Let S be a stable model for D and Σ such that $S \not\models Q$. Let T be the least fixpoint of P^+. Define:

$$S' = \{R^*(\mathbf{a}) \mid R(\mathbf{a}) \in T\} \cup \{R^+(\mathbf{a}) \mid R(\mathbf{a}) \in S\} \cup \{R^-(\mathbf{a}) \mid R(\mathbf{a}) \in T \setminus S\} \cup \{\hat{R}(\mathbf{a}) \mid R(\mathbf{a}) \in S\}.$$

It is then not difficult to verify that S' is a model of D and Σ' under stratified semantics, and $S' \not\models Q'$ (since $\{\mathbf{a} \mid R(\mathbf{a}) \in S\} = \{\mathbf{a} \mid R^+(\mathbf{a}) \in S'\}$).

“$\Leftarrow$” Let S' be a model for D and Σ' obtained using the chase (for disjunctive TGDs with stratified negation) such that $S' \not\models Q'$. For a predicate R and a database D, let $D(R)$ be the set of all the tuples $\mathbf{a}$ with $R(\mathbf{a}) \in D$. Then, $T := \{R(\mathbf{a}) \mid \mathbf{a} \in S'(R^*)\}$ is the unique minimal model for P^+, and $\{S'(R^+), S'(R^-)\}$ forms a partition of $S'(R^*)$. Moreover, the NTGDs in Σ'_4 enforce $S'(\hat{R}) \cap S'(R^-) = \varnothing$ and $S'(R^+) \subseteq S'(\hat{R})$. Since $S'(\hat{R}) \subseteq S'(R^*)$ (only a subset of the rules which generate R^* are triggered), and $S'(R^-) = S'(R^*) \setminus S'(R^+)$, we thus have $S'(\hat{R}) = S'(R^+)$. By the construction of the rules Σ'_3, it follows that $S := \{R(\mathbf{a}) \mid \mathbf{a} \in S'(R^+)\}$ is a stable model for D and Σ. Moreover, since $S' \not\models Q'$, and $S(R) = S'(R^+)$, we have $S \not\models Q$. □

The following example illustrates the above translation.

Example 3 Consider the database $D = \{P(a)\}$ and the following finite set of NTGDs Σ:

$$P(x) \wedge \neg R(x) \rightarrow Q(x), \qquad P(x) \wedge \neg Q(x) \rightarrow R(x).$$

Then, Σ' consists of the following DNTGDs (note that some of the rules (like $R^*(x) \rightarrow R^*(x)$) are omitted, because they would add nothing to Σ' under D):

$$P(x) \rightarrow P^*(x), \quad P^*(x) \rightarrow Q^*(x), \quad P^*(x) \rightarrow R^*(x),$$
$$P^*(x) \rightarrow P^+(x) \vee P^-(x), \qquad P^+(x) \wedge P^-(x) \rightarrow \bot,$$
$$Q^*(x) \rightarrow Q^+(x) \vee Q^-(x), \qquad Q^+(x) \wedge Q^-(x) \rightarrow \bot,$$
$$R^*(x) \rightarrow R^+(x) \vee R^-(x), \qquad R^+(x) \wedge R^-(x) \rightarrow \bot,$$
$$P(x) \rightarrow \hat{P}(x),$$
$$\hat{P}(x) \wedge R^-(x) \rightarrow \hat{Q}(x), \quad \hat{P}(x) \wedge Q^-(x) \rightarrow \hat{R}(x),$$
$$\hat{P}(x) \wedge P^-(x) \rightarrow \bot, \qquad P^+(x) \wedge \neg \hat{P}(x) \rightarrow \bot,$$
$$\hat{Q}(x) \wedge Q^-(x) \rightarrow \bot, \qquad Q^+(x) \wedge \neg \hat{Q}(x) \rightarrow \bot,$$
$$\hat{R}(x) \wedge R^-(x) \rightarrow \bot, \qquad R^+(x) \wedge \neg \hat{R}(x) \rightarrow \bot.$$

To obtain a proper stratified program, we could replace falsum ($\bot$) in the head of $P^+(x) \wedge \neg \hat{P}(x) \rightarrow \bot$ and the corresponding rules for Q and R by an atom $F(x)$ over a fresh predicate F, and put $F(x)$ as a disjunct to our query. Then, F would be in the upper stratum, and all other predicates would be in the lower stratum. ■

The following complexity results for UNBCQ answering in guarded normal Datalog$^\pm$ under the stable model semantics are an immediate corollary of Theorems 8 and 5.

Corollary 9 *Let $\mathcal{R}$ be a relational schema, D a database for $\mathcal{R}$, Σ a finite set of guarded NTGDs on $\mathcal{R}$, and Q a covered UNBCQ over $\mathcal{R}$. Then, deciding whether Q is true in all stable models of D and Σ has the following complexity:*

1. 2-EXPTIME-*complete in general;*
2. EXPTIME-*complete in case the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed;*
3. coNP-*complete in case $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.*

7 Stable Model Semantics for DLs

We now discuss how our decidability results for the stable model semantics can directly be applied to obtain extensions of DLs with non-monotonic negation. The proposed logics are essentially the same as in (Gottlob et al. 2012a), but the crucial difference is that non-monotonic negation is now interpreted via the stable model semantics. This can lead to better query answers, as Example 5 below demonstrates.

We extend *DL-Lite*$_\mathcal{R}$ (which is underlying the OWL 2 QL profile) and *DL-Lite*$_{\mathcal{R},\sqcap}$ (Calvanese et al. 2007; Poggi et al. 2008), and $\mathcal{ELHI}$ (Baader, Brandt, and Lutz 2005) with nonmonotonic negation under the stable model semantics.

Definition 10 Recall that a *DL-Lite*$_{\mathcal{R},\sqcap}$ knowledge base consists of a pair $(\mathcal{T}, \mathcal{A})$, where the TBox $\mathcal{T}$ is a finite set of concept and role inclusion axioms $U_1 \sqcap \cdots \sqcap U_n \sqsubseteq V$, and the ABox $\mathcal{A}$ is a finite set of concept and role membership axioms $C(a)$ and $R(a, b)$, respectively. A *DL-Lite*$_{\mathcal{R},\sqcap,\text{not}}$ knowledge base $(\mathcal{T}, \mathcal{A})$ consists of a finite set of inclusion axioms $\mathcal{T}$ and a finite set of membership axioms $\mathcal{A}$, where:

- Any *DL-Lite*$_{\mathcal{R},\sqcap}$ inclusion axiom is a *DL-Lite*$_{\mathcal{R},\sqcap,\text{not}}$ inclusion axiom.
- If $U_1 \sqcap \cdots \sqcap U_n \sqsubseteq V$ and $U'_1 \sqcap \cdots \sqcap U'_m \sqsubseteq V$ with $n, m > 0$ are both either concept or role inclusion axioms in *DL-Lite*$_{\mathcal{R},\sqcap}$, and V is positive (i.e., not of the form $V = \neg V'$), then $U_1 \sqcap \cdots \sqcap U_n \sqcap \text{not}\, U'_1 \sqcap \cdots \sqcap \text{not}\, U'_m \sqsubseteq V$ is

a *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$ concept or role inclusion axiom, respectively. Here, the U_i's and U_i''s contain no conjunction, and $\mathrm{not}\, U_i'$ denotes the negation as failure (as opposed to the classical negation "$\neg$" in *DL-Lite*).

- For any concept A, any role R, and any individuals a, b, the expressions $A(a)$ and $R(a, b)$ are concept and role membership axioms, respectively.

A *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$ knowledge base $(\mathcal{T}, \mathcal{A})$ is a *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ knowledge base iff all inclusion axioms in $\mathcal{T}$ contain precisely one positive atom on the left-hand side.

Finally, we define $\mathcal{ELHI}_{\mathrm{not}}$ as the extension of $\mathcal{ELHI}$ that allows formulas of the form $\mathrm{not}\, C$ for atomic concepts $C = A$ and for concepts $C = \exists R.B$ to occur in top-level conjunctions on the left-hand side of concept inclusions and formulas $\mathrm{not}R$ and $\mathrm{not}R^-$ for any role R on the left hand side of role inclusions (in addition to at least one positive concept or one positive role, respectively, to ensure guardedness). ■

The semantics of *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) is defined by translating a given *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) knowledge base KB into a normal guarded Datalog$^{\pm}$ program P_{KB} and by computing the stable model semantics of P_{KB}. More details on the translation into normal Datalog$^{\pm}$ can be found in (Gottlob et al. 2012a). Similarly it is straightforward to translate $\mathcal{ELHI}_{\mathrm{not}}$ knowledge bases into normal guarded Datalog$^{\pm}$-programs in order to obtain a stable semantics for $\mathcal{ELHI}_{\mathrm{not}}$.

The following example demonstrates how Example 1 from the introduction can be formalized in $\mathcal{ELHI}_{\mathrm{not}}$.

Example 4

$$\begin{aligned} \mathrm{Person} \sqcap \mathrm{notEven} &\sqsubseteq \mathrm{Odd} \\ \mathrm{Person} \sqcap \mathrm{notOdd} &\sqsubseteq \mathrm{Even} \\ \mathrm{Person} \sqcap \mathrm{Even} &\sqsubseteq \exists \mathrm{hasParent.Odd} \\ \mathrm{Person} \sqcap \mathrm{Odd} &\sqsubseteq \exists \mathrm{hasParent.Even}. \end{aligned}$$

If we consider the ABox $\{\mathrm{Person}(\mathrm{mary})\}$, we obtain two (infinite) stable models for the knowledge base that look very similar to the models obtained in Example 1, the two models correspond to the case in which $\mathrm{Odd}(\mathrm{mary})$ and to the case in which $\mathrm{Even}(\mathrm{mary})$ holds. ■

In the next example, the stable model semantics leads to better query answers than the well-founded semantics.

Example 5

$$\begin{aligned} \mathrm{Hotel} \sqcap \mathrm{Perfect} &\sqsubseteq \exists \mathrm{Beach} \sqcap \exists \mathrm{Pool} \\ \mathrm{FiveStar} &\sqsubseteq \mathrm{Hotel} \\ \mathrm{FiveStar} \sqcap \mathrm{not}\exists \mathrm{Pool} &\sqsubseteq \exists \mathrm{Beach} \\ \mathrm{FiveStar} \sqcap \mathrm{not}\exists \mathrm{Beach} &\sqsubseteq \exists \mathrm{Pool} \\ \exists \mathrm{Beach} &\sqsubseteq \exists \mathrm{SwimOpp} \\ \exists \mathrm{Pool} &\sqsubseteq \exists \mathrm{SwimOpp} \\ \mathrm{Hotel} \sqcap \exists \mathrm{SwimOpp} &\sqsubseteq \mathrm{Excellent}. \end{aligned}$$

Given the ABox $\{\mathrm{FiveStar}(\mathrm{ritz})\}$, the well-founded semantics for the knowledge base would only contain the atoms $\{\mathrm{FiveStar}(\mathrm{ritz}), \mathrm{Hotel}(\mathrm{ritz})\}$. In contrast to that, all stable models of the same knowledge base contain in addition the atoms $\exists \mathrm{SwimOpp}(\mathrm{ritz})$ and $\mathrm{Excellent}(\mathrm{ritz})$. This correctly reflects the fact that, while we are unable to know whether a given five star hotel has a swimming pool or a beach, we can be sure that one of these facts is true. ■

The decidability results for query answering relative to the stable model semantics can directly be applied to DLs.

Theorem 11 *Let $\mathcal{L}$ be DL-Lite*$_{\mathcal{R},\mathrm{not}}$*, DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$ *or $\mathcal{ELHI}_{\mathrm{not}}$. Then, given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a covered UNBCQ[2] Q, deciding whether Q is true under the stable model semantics of $\mathcal{K}$ is in EXPTIME (resp., coNP) in the combined (resp., data) complexity. For $\mathcal{ELHI}_{\mathrm{not}}$ the combined complexity is EXPTIME-complete. In all cases, the data complexity is coNP-complete.*

Proof (Sketch). The upper bounds for the complexity are an immediate consequence of Corollary 9. The lower bound for combined complexity of query answering over a $\mathcal{ELHI}_{\mathrm{not}}$ knowledge base is a consequence of the EXPTIME lower bound for subsumption checking in $\mathcal{ELI}$ (Baader, Brandt, and Lutz 2008, Theorem 3), as a subsumption $C \sqsubseteq D$ can be checked by answering the query $C \sqcap \mathrm{not}D$. The lower bound for the data complexity follows from a reduction of the UNSAT-problem; see (Gottlob et al. 2014). □

8 Summary and Outlook

This paper has provided the first fully general stable model semantics for guarded existential rules with negation. We proved that query answering is decidable under this semantics, and we determined the complexity of this problem for various settings. On the way to these results, we defined disjunctive rule sets under FO semantics augmented by stratified negation, and also studied the complexity of query answering based on this formalism, which is of interest in its own right. Finally, we showed how our results can be used for adorning well-known DLs with stable model negation.

For space reasons, many aspects and possible extensions of our work had to remain undiscussed and will be given more attention in the full paper. One feature that we already mentioned are EGDs. Another issue is the unique name assumption (UNA). Our approach for defining the SMS follows the standard semantics for logic programming with function symbols, where different Skolem terms are considered to be different elements that cannot be unified. In a precise sense, this means that our semantics applies the UNA not only to database constants, but also to invented null values, i.e., Skolem terms. As long as negation is not used, this does not matter. But with negation, this may become an issue in some cases. Reconsider Example 1 and let $\Sigma_c = \Sigma_a \cup \{$*Person(x), not Parent(x,x)*$\rightarrow ok\}$. Under our current semantics, $\{$*Person(mary)*$\} \cup \Sigma_c \models ok$. However, if one does not apply the UNA, then ok would not be derivable, because there would exist models in which, e.g., *mary = f(mary)*. We are able to define a version of the stable model semantics without the UNA, too, and we call this the *equality-friendly SMS*, in analogy to (Gottlob et al. 2012a). This semantics is defined in a totally analogous way to the one defined here, except that instead of using stratified GDNTGDs, we use guarded fixpoint logic. As will be shown in the full paper, decidability and the same complexity results hold.

[2] See (Gottlob et al. 2014) for a definition.

Ongoing work includes in particular the SMS for disjunctive guarded TGDs with negation (GDNTGDs). The semantics for this is, again, directly inherited from the corresponding SMS for disjunctive logic programming with function symbols. As this can be encoded into MSO in a similar way as the normal stable semantics, query answering is decidable. By a similar encoding into MSO, we have also obtained decidability for the SMS for disjunctive weakly guarded and weakly frontier-guarded TGDs with negation. We are currently still exploring the precise complexity of all these problems. We also plan to extend suitable versions of the interesting MKNF approach (Motik and Rosati 2010) by our techniques, so to achieve larger decidable classes.

Acknowledgments. This work was supported by the EPSRC under the grant EP/H051511/1 "ExODA", by the European Research Council under the FP7/2007-2013/ERC grant 246858 "DIADEM", and by a Yahoo! Research Fellowship. Georg Gottlob is a James Martin Senior Fellow, and also gratefully acknowledges a Royal Society Wolfson Research Merit Award. The work was carried out in the context of the James Martin Institute for the Future of Computing.

References

Alviano, M.; Faber, W.; Leone, N.; and Manna, M. 2012. Disjunctive Datalog with existential quantifiers: Semantics, decidability, and complexity issues. *TPLP* 12(4/5):701–718.

Arenas, M.; Gottlob, G.; and Pieris, A. 2014. Expressive languages for querying the Semantic Web. In *Proc. of PODS*. In press.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, 364–369.

Baader, F.; Brandt, S.; and Lutz, C. 2008. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED*.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2009. Extending decidable cases for rules with existential variables. In *Proc. of IJCAI*, 677–682.

Baget, J.-F.; Mugnier, M.-L.; Rudolph, S.; and Thomazo, M. 2011. Walking the complexity lines for generalized guarded existential rules. In *Proc. of IJCAI, Volume II*, 712–717.

Baget, J.-F.; Leclère, M.; and Mugnier, M.-L. 2010. Walking the decidability line for rules with existential variables. In *Proc. of KR*, 466–476.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Proc. of ICALP*, 73–85.

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

Bourhis, P.; Morak, M.; and Pieris, A. 2013. The impact of disjunction on query answering under guarded-based existential rules. In *Proc. of IJCAI*, 796–802.

Calì, A.; Gottlob, G.; Lukasiewicz, T.; Marnette, B.; and Pieris, A. 2010. Datalog+/–: A family of logical knowledge representation and query languages for new applications. In *Proc. of LICS*, 228–242.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48:115–174.

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

Calì, A.; Gottlob, G.; and Pieris, A. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193:87–128.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning* 39(3):385–429.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33(3):374–425.

Eiter, T., and Simkus, M. 2010. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.* 11(2).

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.* 172(12/13):1495–1539.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP/SLP*, 1070–1080.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2012a. Equality-friendly well-founded semantics and applications to description logics. In *Proc. of AAAI*, 757–764.

Gottlob, G.; Manna, M.; Morak, M.; and Pieris, A. 2012b. On the complexity of ontological reasoning under disjunctive existential rules. In *Proc. of MFCS*, 1–18.

Gottlob, G.; Hernich, A.; Kupke, C.; and Lukasiewicz, T. 2014. Stable model semantics for guarded existential rules and description logics. Technical report, Department of Computer Science, University of Oxford.

Gottlob, G.; Manna, M.; and Pieris, A. 2013. Combining decidability paradigms for existential rules. *TPLP* 13(4/5):877–892.

Hernich, A.; Kupke, C.; Lukasiewicz, T.; and Gottlob, G. 2013. Well-founded semantics for extended Datalog and ontological reasoning. In *Proc. of PODS*, 225–236.

Lukasiewicz, T. 2010. A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Trans. Knowl. Data Eng.* 22(11):1577–1592.

Magka, D.; Krötzsch, M.; and Horrocks, I. 2013. Computing stable models for nonmonotonic existential rules. In *Proc. of IJCAI*, 1031–1038.

Maier, D.; Mendelzon, A. O.; and Sagiv, Y. 1979. Testing implications of data dependencies. *ACM Trans. Database Syst.* 4(4):455–469.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *J. ACM* 57(5).

Motik, B.; Horrocks, I.; and Sattler, U. 2009. Bridging the gap between OWL and relational databases. *J. Web Sem.* 7(2):74–89.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. Data Semantics* 10:133–173.

Rabin, M. O. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. Am. Math. Soc.* 141:1–35.

Subrahmanian, V. S. 1999. Nonmonotonic logic programming. *IEEE Trans. Knowl. Data Eng.* 11(1):143–152.

Thomazo, M. 2011. *Conjunctive Query Answering Under Existential Rules — Decidability, Complexity, and Algorithms*. Ph.D. Dissertation, Univ. of Montpellier, France.