Stable Model Semantics for Guarded Existential Rules and Description Logics: Decidability and Complexity

GEORG GOTTLOB, University of Oxford, UK
ANDRÉ HERNICH, Amazon Web Services, Germany
CLEMENS KUPKE, University of Strathclyde,UK
THOMAS LUKASIEWICZ, University of Oxford, UK

This work investigates the decidability and complexity of database query answering under guarded existential rules with nonmonotonic negation according to the classical stable model semantics. In this setting, existential quantification is interpreted via Skolem functions, and the unique name assumption is adopted. As a first result, we show the decidability of answering first-order queries based on such rules by a translation into the satisfiability problem for guarded second-order formulas having the tree-model property. To obtain precise complexity results for unions of conjunctive queries, we transform the original problem in polynomial time into an intermediate problem that is easier to analyze: query answering for guarded disjunctive existential rules with stratified negation. We obtain precise bounds for the general setting and for various restricted settings. We also consider extensions of the original formalism with negative constraints, keys, and the possibility of negated atoms in queries. Finally, we show how the above results can be used to provide decidability and complexity results for a natural adaptation of the stable model semantics to description logics such as $\mathcal{ELHI}$ and the *DL-Lite* family.

CCS Concepts: • **Computing methodologies** → **Logic programming and answer set programming**; **Description logics**; • **Information systems** → **Query languages**; • **Theory of computation** → **Logic**; **Theory and algorithms for application domains;**

Additional Key Words and Phrases: Answer set programming, stable models, complexity, decidability

ACM Reference format:
Georg Gottlob, André Hernich, Clemens Kupke, and Thomas Lukasiewicz. 2021. Stable Model Semantics for Guarded Existential Rules and Description Logics: Decidability and Complexity. *J. ACM* 68, 5, Article 35 (October 2021), 87 pages.
https://doi.org/10.1145/3447508

35

Georg Gottlob is a Royal Society Research Professor and acknowledges support by the Royal Society under this scheme (Project "RAISON DATA" RP/R1/201074). Georg Gottlob and Thomas Lukasiewicz were also supported by the EPSRC grants EP/M025268/1 "VADA" and EP/R013667/1 "RealPDBs." Thomas Lukasiewicz was also supported by the AXA Research Fund and by the Alan Turing Institute under the EPSRC grant EP/N510129/1. Clemens Kupke was supported by the EPSRC grant EP/N015843/1.
Authors' addresses: G. Gottlob and T. Lukasiewicz, Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom; emails: {georg.gottlob, thomas.lukasiewicz}@cs.ox.ac.uk; A. Hernich, Amazon Web Services, Krausenstr. 38, 10117 Berlin, Germany; email: andre.hernich@gmail.com; C. Kupke, Department of Computer and Information Sciences, University of Strathclyde, Livingstone Tower, 16 Richmond Street, Glasgow, G1 1XQ, Scotland, United Kingdom; email: clemens.kupke@strath.ac.uk.

0004-5411/2021/10-ART35 $15.00
https://doi.org/10.1145/3447508

Corrected Version of Record. V.1.1. Published November 12, 2021.

1 INTRODUCTION

Existential rules have attracted much interest in both the database and the knowledge representation and reasoning communities (see Reference [34] for an overview). These are essentially universally quantified Horn clauses (or Datalog rules) extended by the possibility of existential quantification in rule heads. An example for an existential rule is

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

which states that every base-level employee working on a project must be managed by somebody. Existential rules of the most basic form are also well-known as *tuple-generating dependencies (TGDs)*; see Reference [15]. In the Datalog$^{\pm}$ family of ontology languages [32, 33], TGDs are complemented by a number of other types of rules, notably, by ***negative constraints (NCs)***, such as $\forall x \, (Ceo(x), BaseLevel(x) \rightarrow \bot)$, which means that nobody can simultaneously be a CEO and a base-level employee), where $\bot$ stands for *false*, and a restricted class of ***equality-generating dependencies (EGDs)***, called *non-conflicting EGDs* [33], which may express key constraints (keys) and functional dependencies in relational data, for example, the rule $\forall x \forall y \forall z (IsManagedBy(x, y), IsManagedBy(x, z) \rightarrow y = z)$, which expresses in database terms that the first column (or attribute) of the *IsManagedBy* relation is a key. At the same time, the resulting language is syntactically restricted to achieve decidability and even tractability. In this article, we often omit the universal quantifiers when writing a rule.

Existential rules under various names are an extension of Datalog and have many applications both in database management and knowledge representation and reasoning. A first important application in the database area is *data exchange* [7, 54, 55, 58, 71], which consists of the problem of computing an instance of a target schema, given an instance of a source schema and a specification, in form of TGDs and possibly also EGDs of the relationship between the source and target data, plus a set of TGDs (or EGDs) to hold on the target relation. Data exchange, and the related issue of *schema mapping*, are mostly used in case of data integration, for example, after a merger of two companies. An important experimental system built by IBM Watson was CLIO [53], which has not only influenced IBM products but also many more recent systems such as, for example, the integrated VADA data wrangling system [79].

An area that extends classical database systems by rule-based knowledge, where TGDs play a central role is ***ontology-based data access (OBDA)*** [23, 103, 116], which enriches a database D by a so-called ontology Σ, which contains (possibly recursive) axioms or rules whereby (possibly infinitely many) new facts can be inferred or integrity constraints enforced. OBDA allows one to query a database enriched by an ontology without the necessity of materializing all facts that are entailed by D and Σ. Originally, these ontologies were based exclusively on **description logics (DLs)**. In this setting, they gave rise to the Mastro system [37], which has been successfully applied to data integration, database quality improvement, and knowledge sharing in various contexts, for example, in a major bank [106]. The *DL-Lite* axioms used in these ontologies can be reformulated as TGDs and EGDs [33]. More recently, OBDA frameworks that directly use the more powerful setting of TGDs and other constraints were introduced, for example, the Datalog$^{\pm}$ framework [33, 34].

There are literally hundreds of papers on data exchange and OBDA. However, there are many other applications where existential rules play an important role. We briefly mention a few. *Knowledge graph management*: In References [17–19], the Vadalog system for knowledge graph management is described. This system uses the formalism of TGDs as a main language construct. Some of its commercial applications are described in References [16, 41]. *Data distribution*: In Reference [61], a declarative framework based on TGDs and EGDs for reasoning about data distribution issues is presented. *Bag semantics*: It was shown in Reference [22] that the relational bag

semantics (where relations may contain duplicates) for Datalog can be defined and directly used via TGDs. *Querying the Semantic Web and SPARQL*: In Reference [8], a powerful semantic web query and reasoning language based on TGDs (with stratified negation) is presented. This language covers SPARQL with entailment regimes for RDFS and OWL vocabularies, but is more expressive. *Data quality assessment and data cleaning*: TGDs and versions of Datalog$^{\pm}$ have been used for detecting data inconsistencies, assessing data quality, repairing databases, and data cleaning [30, 94]. *Reasoning and classifying in chemistry:* A TGD-based system performing automatic classification of chemical compounds is presented in Reference [89]. *Verification and model checking:* Finally, let us remark, that the language Datalog LITE [65], which is essentially guarded Datalog without existential rules, can be expressed by the class of guarded TGDs that we study in the present article. It was shown in Reference [65] that Datalog LITE naturally encompasses popular modal and temporal logics, such as CTL or the alternation-free μ-calculus, and is equivalent to the alternation-free portion of guarded fixpoint logic. These logics are thus all covered by guarded normal TGDs with stratified negation. Note that ***normal TGDs (NTGDs)*** are existential rules that may contain negated atoms of the form $\neg R(\mathbf{x})$ in their bodies. However, guarded NTGDs with stratified negation, let alone with stable negation (as studied here), are a much more expressive formalism than Datalog LITE. This inspires possible future research on wider applications to model checking and system verification using guarded TGDs with the stable model semantics. Next to the technical challenge, all these applications motivated our research on whether a significant class of TGDs, for which query answering is decidable, can be married to the stable model semantics without spoiling decidability and complexity.

We now provide an overview of key technical details and underlying ideas of our work. Query answering based on sets of existential rules is the following decision problem. For a database D, a set of existential rules Σ, and a Boolean query Q, decide whether Q evaluates to true on $D \cup \Sigma$. As will be discussed in more detail in Section 2, Boolean query answering is computationally equivalent to the ***query of tuple (QOT)*** *problem* of checking whether for a possible answer tuple $(a_1, \ldots, a_n)$, $Q(a_1, \ldots, a_n)$ is indeed an answer for D and Σ. For this reason, while we concentrate on Boolean queries, our results are equally valid for the QOT problem, which is the relevant decision problem for answering non-Boolean queries.

The queries mostly considered in the related literature are **conjunctive queries (CQs)**, but we here also deal with various extensions, such as the well-known class of **unions** (i.e., disjuncts) **of conjunctive queries (UCQs)**, and with it extensions: **unions of normal CQs (UNCQs)**, which are allowed to contain negative literals, and *covered* UNCQs, which are like UNCQs but where in each query-disjunct, for each negated atom α, there must exist a positive atom β containing all variables of α. These classes will be defined more precisely in Sections 2 and 3. Their Boolean variants are abbreviated using the letter "B" in an appropriate position; so, for example, BCQ means Boolean conjunctive query, and UNBCQ stands for union of normal Boolean conjunctive queries.

Query answering with TGDs is undecidable [14, 39], even in the case of fixed sets of plain TGDs, as arbitrary TGDs allow to simulate a universal Turing machine [32] and singleton sets of TGDs [11]. Therefore, to obtain decidability, restrictions need to be imposed on TGDs.

One of the largest, most fundamental, and most significant decidable classes of first-order logic formulas is the *guarded fragment* [6], which we consider here in the context of TGDs. A TGD is *guarded* if its body contains an atom, called *guard*, that covers all body variables. Guarded TGDs were studied in References [32, 33], where the query-answering problem for guarded TGDs and for various extensions was shown to be decidable, and where the complexity of this problem was determined. To enhance their expressiveness, guarded TGDs were extended by stratified negation [33], with the same decidability and complexity results.

To further improve the suitability of TGDs for knowledge representation, there have been efforts by other authors to adapt the more powerful ***stable model semantics (SMS)*** [62] to guarded rules, but this has succeeded only for restricted fragments of guarded TGDs, sometimes enriched by other (orthogonal) features. However, it was not clear whether an extension of the full language of guarded TGDs to the SMS would be decidable at all, and if so, what the complexity of query answering would then be. For example, in Reference [51], the formalism $\mathbb{FNC}$ was defined, featuring very restricted types of guarded rules, for which query answering under the SMS is decidable. However, the authors of Reference [51] say that it is "problematic" to extend the formalism with other types of guarded rules under the SMS, and thus, to cover the full class of guarded TGDs. It is precisely the main goal of the present article to clarify this, not only for the full basic formalism of TGDs, but also for various extensions ((negative) constraints, **equality-generating dependencies (EGDs)**, and keys). In particular, we prove the decidability of query answering based on TGDs under the SMS and derive complexity results.

Some readers may experience a *déjà vu* at this point. After the original SMS was proposed in 1988, there have, in fact, been two very appealing more recent approaches that define the stable models semantics for full first-order logic. These two approaches actually semantically coincide. One is the model-theoretically defined *equilibrium logic* approach introduced by Pearce in 1996 [98], based on the so-called here-and-there logic, further studied in References [98–100, 102], and the second-order semantics for SMS, here abbreviated by *SOS semantics*, by Ferraris, Lee, and Lifschitz [56, 57], which is based on model minimization techniques similar to circumscription expressed in second-order logic. Given that TGDs are first-order formulas, why do we not just apply, say, the SOS here. Are we reinventing the wheel? We do not. We here base our work on the classical SMS, which is the same **logic programming (LP)** paradigm also used in major software systems such as DLV [84] or Clasp and Potassco [59, 60]. We thus use the classical SMS setting for LP with Herbrand interpretations and with the adoption of the ***unique name assumption (UNA)***. Existential quantifiers are here interpreted by Skolem functions and the UNA states that different elements of the Herbrand universe are to be considered different objects and thus cannot be unified. The SOS instead (just as Equilibrium Logic) is closer to classical logic and does not forbid such unifications. Interestingly, it was recently shown that for guarded TGDs, answering UBCQs under the SOS is undecidable [4]. In Section 3.4, we compare the SOS to the SMS and give examples where these semantics significantly differ. However, we also show that the UNA alone is responsible for the difference, and that when adding the UNA in form of axioms (expressible as unguarded TGDs) to the SOS, then the SOS (and thus also Pearce's approach) coincides with the SMS. Our decidability result and our complexity results thus carry over to the SOS under the UNA. In this sense, we actually derive positive results for the SOS as well.

The way in which we define the SMS for NTGDs is, as already explained, inherited from the well-known stable model semantics of normal logic programs with function symbols [62]. In fact, we interpret existential quantification in rule heads by Skolem terms, which means that the SMS for guarded NTGDs is already defined via the SMS for normal logic programs with function symbols. Thus, we simply use the SMS semantics on the skolemized version of the NTGDs.

The following simple examples and their discussion shall illustrate various aspects of the so-defined SMS and highlight the intrinsic difficulties that one is confronted with when searching for algorithmic solutions. Many further examples will be given in the following sections.

Example 1.1. 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 (who is a person), 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:

$$\begin{aligned} \mathit{Person}(x) &\rightarrow \exists y\, \mathit{Parent}(x,y), \\ \mathit{Parent}(x,y) &\rightarrow \mathit{Person}(y), \\ \mathit{Person}(x), \neg \mathit{Even}(x) &\rightarrow \mathit{Odd}(x), \\ \mathit{Person}(x), \neg \mathit{Odd}(x) &\rightarrow \mathit{Even}(x), \\ \mathit{Parent}(x,y),\ \mathit{Odd}(x) &\rightarrow \mathit{Even}(y), \\ \mathit{Parent}(x,y),\ \mathit{Even}(x) &\rightarrow \mathit{Odd}(y). \end{aligned}$$

After Skolemization, the first rule becomes:

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

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, and the other containing (among others) the facts *Even(mary), Odd(f(mary)), Even(f(f(mary)))*, and so on. Let Q_a be the conjunctive query (CQ) $\exists x, y, z$ *(Parent(x,y), Parent(y,z), Odd(x), Odd(z))*, asking whether there exist members x and z of odd generations, such that z is a grandparent of x. Also, let Q'_a be the CQ $\exists x, y$ *(Parent(x,y), Odd(x), Odd(y))*, asking nearly the same, except for using the parent relation instead of grandparent, and let Q''_a be the conjunctive query $\exists x$ *(Odd(x), Even(x))*, asking if there exists a member of both odd and even generation. Such queries are evaluated by *certain answers* (see Definition 3.6), which, in AI, corresponds to skeptical entailment. Then, by the SMS, Q_a evaluates to true on $D_a \cup \Sigma_a$, while both Q'_a and Q''_a evaluate to false on $D_a \cup \Sigma_a$ (see Section 3).

Note that it is easy to exhibit rule sets Σ that generate infinitely many infinite stable models, as illustrated by the following example:

Example 1.2. Let $D_b = D_a = \{\mathit{Person}(\mathit{mary})\}$, and let Σ_b be obtained by taking just the first four rules of the rule set Σ_a of Example 1.1. Then, there is an infinite number of stable models of $D_b \cup \Sigma_b$, one for each possibility of having for each Skolem term $f^i(\mathit{mary})$ either $\mathit{Even}(f^i(\mathit{mary}))$ or $\mathit{Odd}(f^i(\mathit{mary}))$, where $1 \leqslant i \leqslant \infty$. Hence, the conjunctive query Q_a of Example 1.1 evaluates to false on $D_b \cup \Sigma_b$ under the SMS.

It seems that the interplay of nonmonotonic negation with these two types of infinity made it hard to understand how the SMS for guarded NTGDs could be brought under control. In particular, as the above examples show, the following three simultaneous features are tightly interwoven: (i) recursive predicate definitions, (ii) existential quantifiers appearing in recursive cycles, possibly enforcing infinite models, and (iii) negation appearing within such recursive cycles. Accordingly, previous solutions for decidable query answering with guarded TGDs under the SMS either drop one of these features right away or significantly limit a feature or soften the interaction of at least two features. For example, the SMS for stratified NTGDs [33] forbids the use of negation within recursive cycles. A similar type of stratification has also been studied in Reference [88] under the name R-stratification. Stratified sets of NTGDs have at most one preferred model, which is also a stable model. In case the NTGDs are either guarded or fulfill acyclicity constraints, query answering is decidable. A nontrivial generalization of stratification (based on odd cycles) was given in References [25, 26].

Approaches where the existential quantifier is prevented to interact with recursive loops are based on the concept of weakly acyclic TGDs [47, 54] or their generalizations, e.g., broader criteria such as super-weak acyclicity [93], R-acyclicity [88], or joint acyclicity [80, 82], see also References [75, 81]. A very sophisticated method of interaction prevention was developed in

Reference [10]. In such approaches, the stable models are finite. An extreme case is the Datalog LITE [65] language, where guarded Datalog is considered without existential quantifiers. Even this language has a host of applications, some of which were mentioned earlier in this section. Finally, the already mentioned language $\mathbb{FNC}$ from Reference [51] and its generalization to the more expressive non-disjunctive fragment of the $\mathbb{BD}$ language in Reference [113] are most congenial to our endeavor, as these formalisms allow both existential quantification and negation to jointly appear within recursive cycles. This is achieved by a very strict rule syntax and a strong limitations on the predicate argument positions that are allowed to contain Skolem functions. Guarded rules significantly relax these restrictions.

In this article, after giving basic definitions in Section 2, we define, in Section 3 the SMS and compare it to the SOS. The SOS is an elegant formalism, which to a first-order theory T associates a **second-order (SO)** formula $SOS(T)$ whose models are defined to be the (SOS) stable models of T. In our setting, the original theory $T = (D, \Sigma)$ consists of a database D of ground facts and a set of TGDs Σ, and we write $SOS(D, \Sigma)$ for the resulting SOS second-order formula, whenever convenient. We prove that for databases D and arbitrary sets Σ of NTGDs, the SMS can be defined in terms of the SOS by adding UNA axioms, yielding an augmented set Σ^D_{UNA} of TGDs, such that the models of the SO formula $SOS(D, \Sigma^D_{UNA})$ are (up to isomorphism) precisely the SMS models of D with Σ.

Towards our decidability results, an insightful initial observation (which will be formally shown in Section 3) is that all stable models of a database D together with a set Σ of guarded NTGDs are *tree-shaped*, which means that they have bounded treewidth. We will show that this is due to the interplay between guardedness and the UNA. This means, in particular, that also the models of the second-order formula $SOS(D, \Sigma^D_{UNA}) \wedge \neg Q$ are all tree-shaped when Q is a Boolean first-order query. Moreover, we note that this formula actually belongs to ***guarded second-order logic (GSO)***, which is essentially the fragment of SO where quantified predicate variables are restricted to range over subsets of extensions of the free predicate variables. By a well-known meta-theorem stating that the satisfiability problem for GSO formulas having models of bounded treewidth is decidable [42, 72], we conclude that the satisfiability of $SOS(D, \Sigma^D_{UNA}) \wedge \neg Q$ is decidable, which means that also $SOS(D, \Sigma^D_{UNA}) \models Q$ is, and, in turn, query answering based on the SMS.

Theorem 1.3. *There is an algorithm that, given a database D, a finite set Σ of guarded NTGDs, and a Boolean first-order query Q, decides whether Q is true in all stable models for D and Σ.*

Unfortunately, the GSO formula constructed for proving the above result does not give us a tight upper bound for the complexity of query evaluation with guarded NTGDs. To arrive at such a bound, we pursued the following different ideas: Recall that finding a stable model essentially involves first guessing a model M 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. When infinite models are involved, guessing a model at once clearly becomes problematic. The idea, is therefore to translate the query answering problem to an equivalent auxiliary formalism with a more algorithmic flavor, where the guessing is divided into small nondeterministic steps each only guessing a single atom via a rule application, thereby successively constructing a potential stable model. A further idea is that this guessing might be stopped as soon as a sufficiently large initial part of a model has been developed, which (i) guarantees that it can be expanded to a full model and (ii) can be used for correct query answering.

We identified the formalism of stratified (sets of) guarded disjunctive normal TGDs (guarded DNTGDs) as a suitable auxiliary formalism. Stratified guarded DNTGDs generalize plain guarded DTGDs, which were studied in References [3, 28, 29, 68], by adding stratified (nonmonotonic) negation. The formalism of stratified DNTGDs will be defined in Section 4, where we will also present

a specific stratified and disjunctive variant of the well known *chase* procedure [15] for generating (possibly infinite) models from a database and a set of stratified DNTGDs. Essentially, this chase fires the DNTGDs stratum by stratum, and whenever the body of a DNTGD is matched, a nondeterministically chosen head atom is derived. A DNTGD may fire several times with different head atoms, thus, the heads represent inclusive disjunctions. This differs from the classical disjunctive ASP setting, where minimal models only are considered. While the so-obtained models are not necessarily minimal, it will become clear in Section 4, that by the use of stratified negation and constraints, minimal models can be simulated when we need them. Let us note that encoding the SMS via disjunctions has been done in the specific context of **answer set programming (ASP)** solvers. For example, modern ASP solvers for (extensions of) Datalog with negation under the SMS, such as Clasp [60] or DLV [2, 90], transform ground instances of a program to large sets of disjunctive clauses, which are then solved using fast SAT solving methods. Here, instead, we use a genuine encoding that works at the non-ground NTGD/DNTGD level and that works uniformly for each input database.

Our transformation from a set Σ of NTGDs to a set Σ' of stratified DNTGDs, which preserves guardedness, will be defined in full detail in Section 4. Σ' will use auxiliary predicates, among which, for each predicate symbol R of the original set Σ, predicates $\hat{R}$, R^+, R^-, and R^*, where the $\hat{R}$ predicates are computed by the TGDs $\hat{\Sigma}$ arising from Σ by eliminating all negative atoms from the rule bodies. The unique minimal model of $\hat{\Sigma}$ is tree-shaped and overapproximates all stable models of D and Σ. The extension of this overapproximation $\hat{R}$ is then partitioned via a disjunctive rule and a constraint into relations R^+ and R^- where the extension of R^+ is a candidate for a stable model. Then, the predicates R^* are computed by rules that simulate the **Gelfond-Lifschitz (GL)** transform of Σ relative to the predicates R^+, and thus, the computed extension of R^* is the minimal model of D and this GL transform. The set of atoms given by the extensions of the relations R^+ is then a stable model of Σ iff it corresponds with the atoms in the extensions of R^* (i.e., $R^+ = R^*$, for each R). The formulation of the GL-transform and the check $R^+ = R^*$ use stratified negation. Moreover, constraints (i.e., rules implying *Fail*) are used for the latter. All details and a comprehensive example (Example 4.3) are given in Section 4. This way, we obtain, via a polynomial transformation, a set Σ' from Σ whose stable models (when restricted to the predicates of Σ) essentially correspond to those of Σ. Therefore, answering conjunctive queries for NTGDs Σ under the SMS can be transformed into query-answering for stratified DNTGDs Σ' under the defined disjunctive chase semantics. Still, these models can be infinite, and it will be the task of Section 5 to show that only an initial part of these models is needed.

The overall goal of Section 5 is to determine the complexity for answering UBCQs and UNBCQs for NTGDs under the SMS. The lower bounds, that is, the hardness results, are proven in Section 5.1, and the upper bounds in Section 5.2. To better explain the role of guardedness and thus of the tree-shaped models (the technical term here is *[D]-acyclic* models) for obtaining upper bounds, we start in Section 5.2 with an analysis of the stratified chase, when a set Σ of guarded DNTGDs is chased over a database D. We show that due to guardedness, this chase essentially develops a forest "growing out" from the database, producing possibly infinitely many new atoms containing null values (or, equivalently, Skolem terms). We show that there is an integer d^*, which depends only on the number and maximum arity of the relation symbols in D and Σ, and on the number of atoms in a query Q, such that the trees do not need to be developed below depth d^* for answering Q. The reason is that due to some combinatorial isomorphism properties, below depth d^* of the chase forest, no relevant new atoms for answering Q can be derived, because whenever Q cannot been satisfied within the initial fragment of depth d^*, then either the chase has already stopped, or there is a way to continue the chase via subtrees isomorphic to already existing subtrees, arriving

at an infinite model that does not satisfy Q either. It follows that a UNBCQ Q is satisfied by all chase models of (D, Σ) iff it is satisfied by all initial fragments of depth at most d^* of such chase models. There are clearly only finitely many such initial fragments, and we can thus compute all of them via the chase (using backtracking instead of nondeterminism). This is how we get to our first algorithm and complexity upper bounds. In particular, we derive the following upper bounds: Answering UNBCQs for a database D and a set Σ of stratified DNTGDS under the disjunctive chase semantics (or, equivalently, for D and a set Σ of NTGDs under the SMS) is (1) in co-3-NEXPTIME in combined complexity, i.e., when D, Σ, and Q are given in input, (2) in co-2-NEXPTIME when the arities are fixed, and (3) in co-NP in data complexity, i.e., when only D is given in input, and both Σ and Q are fixed. The latter bound is a matching one, and thus we get:

Theorem. *UNBCQs answering for NTGDs under the SMS is co-NP complete.*

One nice aspect of the above theorem is that this data complexity is not harder than the data complexity of (normal) Datalog under the SMS, which is also the same as the complexity of propositional reasoning, when the program and the goal are given in input (see Reference [44] for an overview of previous complexity results).

The bounds under points (1) and (2) above are rather high. However, we succeeded to obtain significantly better tight bounds for the class of *covered UNBCQs*, which is a very large class of queries and a nice generalization of UBCQs, which are the queries mostly treated in the knowledge representation and database literature. To obtain these bounds, we develop alternating algorithms for UBCQs in Section 5.2, which, rather than computing and storing the chase tree up to depth d^*, compute the (almost) independent subtrees of this fragment of the chase tree independently, using "only" exponential space in case of combined complexity, and polynomial space in case of bounded arities. We show that these algorithms carry over to answering also covered UNBCQs. These alternating algorithms are very technical. They are informally explained and sketched in Section 5.2, while a detailed treatment is given in Appendix B.

Let us summarize our main complexity results for answering covered UNBCQs for databases D, and sets Σ of guarded NTGDs as follows, where $\models_{stable}$ denotes entailment under the SMS.

Theorem (Main Complexity Theorem for covered UNBCQs). *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$ (to be interpreted as a set of logical ground facts), Σ a set of guarded DNTGDs on $\mathcal{R}$, and Q be a covered UNBCQ over $\mathcal{R}$. Then, deciding $D \cup \Sigma \models_{stable} Q$ has the following complexity:*

(1) 2-EXPTIME*-complete in combined complexity, where* 2-EXPTIME*-hardness holds even for fixed Σ with conjunctive queries.*
(2) EXPTIME*-complete, if the maximum arity w of the relation symbols in $\mathcal{R}$ is fixed, and Q is acyclic, where* EXPTIME*-hardness holds even for atomic queries.*
(3) co-NP*-complete in data complexity (i.e., for fixed Q and fixed Σ), or if $|\mathcal{R}|$, w, and the number of atoms in Q are fixed.*

Let us briefly highlight how these results compare with complexity results on Datalog **query answering (QA)** with negation under the SMS by various authors summarized in Reference [44] and on QA with plain TGDs by various authors summarized in Reference [44] without negation (or TGDs with stratified negations, where QA has the same complexity as for plain TGDs), more recently obtained in References [32, 33]. Datalog QA under the SMS is co-NEXPTIME, while it is in co-NP for bounded arities as well as for data complexity. Instead for QA with plain TGDs, all complexity results of points (1)–(3) coincide with one exception in point (1): For fixed Σ, the complexity of Datalog-based UBCQ answering (and one can see as well, of UNBCQ answering) is "only" EXPTIME-complete.

The same results hold for covered UBCQ answering with guarded stratified DNTGDs under the disjunctive chase semantics.

In Section 6, we show that all the above decidability and complexity results also hold in case **negative constraints (NCs)** of the form $body(\mathbf{x}) \rightarrow \bot$, where $body(\mathbf{x})$ does not need to be guarded, are added to rule sets of NTGDs under the SMS, because NCs can be simulated by adding the body of each NC as a disjunct to the UBCQ. Of course, the standard trick to simulate such an NC by a rule $body(\mathbf{x}) \wedge \neg p \rightarrow p$ does not work, because this would, in general, not be a guarded rule. In Section 6, we also show that EGDs and keys can be added to the existential rule sets, as long as they are strongly non-conflicting with the existential rules (see Section 6).

Finally, in Section 7, we apply our results to **description logics (DLs)**, studying DLs with non-monotonic negation under the SMS. Again, we obtain decidability and complexity results. In particular, we adopt and study the SMS for the well-known DLs *DL-Lite*$_{\mathcal{F}}$, *DL-Lite*$_{\mathcal{R}}$, and *DL-Lite*$_{\mathcal{A}}$ [38, 103], and $\mathcal{ELHI}$ [9]. All the DLs of the *DL-Lite* family and the DL $\mathcal{ELHI}$ [9] can be embedded into guarded existential rules (with NCs and non-conflicting EGDs) [33]. 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 guarded TGDs, note that the DL axiom $\exists \text{EmpProj} \sqcap \text{BaseLevel} \sqsubseteq \exists \mathit{IsManagedBy}$ can be written as the rule (1).

We carried out this work with the intention to lay the theoretical and algorithmic foundations for reasoning with guarded rules over (possibly) infinite models using the SMS, and we hope that our results will pave the way for the design and implementation of new software and will help to extend existing systems for answer set programming such as as DLV [84], which has already been extended by an interesting class of TGDs, the so-called *shy TGDs* [83] (incomparable to guarded TGDs), or Clasp [59, 60]. Very recently, a first successful experimental implementation of query answering with guarded NTGDs under the SMS was described in Reference [115], and promising experiments were reported. This implementation is heavily based on the results of (the conference version of) the present article, with additional nice optimizations that lead to practical improvements. However, their algorithms and implementation do not profit from (a deterministic execution of) the alternation paradigm. In fact, this implementation still computes and materializes the full disjunctive chase tree, up to double-exponential depth, which means that it uses data structures of triple-exponential size, of which quadruple-exponentially many need to be enumerated, which clearly does not match our double-exponential upper bound for the combined complexity. However, the optimization techniques proposed in Reference [115] may be equally useful when designing a new algorithm based on (a deterministic execution) of the alternating algorithms described in Section 5 of our present article. We are looking forward to such implementations.

2 PRELIMINARIES

In this section, we briefly recall some basics on the Datalog$^{\pm}$ family of ontology languages [32, 33], namely, on relational databases, **unions of conjunctive queries (UCQs)**, **tuple-generating dependencies (TGDs)**, **equality-generating dependencies (EGDs)**, and negative constraints. We also describe the decision problems and the complexity measures considered in this article.

2.1 Databases, Interpretations, and Homomorphisms

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, which are placeholders for unknown values, and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, tuple/equality-generating dependencies, and negative constraints). We denote by $\mathbf{x}$ sequences of variables $x_1, \ldots, x_k$ with $k \geqslant 0$. We assume a *relational schema* (or simply *schema*) $\mathcal{R}$, which is a finite set of *relation names* (or *predicate symbols*, or simply *predicates*). We denote by $\text{ar}(\mathcal{R})$ the maximum arity of a predicate in $\mathcal{R}$. A *term* t is

a constant, a null, or a variable. A list of terms is also referred to as a *tuple*. An *atomic formula* over $\mathcal{R}$ (or $\mathcal{R}$-*atom*, or simply *atom*) has the form $P(t_1, \ldots, t_n)$, where P is an n-ary predicate from $\mathcal{R}$, and $t_1, \ldots, t_n$ are terms. For such an atom $\alpha = P(t_1, \ldots, t_n)$, we denote by $dom(\alpha)$ the set $\{t_1, \ldots, t_n\}$. For a set A of atoms, we denote by $dom(A)$ the union $\bigcup_{\alpha \in A} dom(\alpha)$. A conjunction of atoms is often identified with the set of all its atoms. A *homomorphism* is a mapping $h\colon \Delta \cup \Delta_N \cup \mathcal{V} \to \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $h(c) = c$, (ii) $c \in \Delta_N$ implies $h(c) \in \Delta \cup \Delta_N$, and (iii) h is naturally extended to atoms by mapping an atom $P(t_1, \ldots, t_n)$ to $P(h(t_1), \ldots, h(t_n))$, to sets of atoms by mapping each such set I to $\{h(\alpha) \mid \alpha \in I\}$, and, similarly, to conjunctions of atoms by applying h to each of the conjuncts.

Let A and B be two sets of atoms, and let h^+ be a homomorphism such that $\forall x \in dom(A), h^+(x) \in dom(B)$. The restriction $h = h^+|_{dom(A)}$ of h^+ to the domain $dom(A)$ of A is called a *homomorphism* from A to B, and we write $h : A \to B$. A homomorphism $f : dom(A) \to dom(B)$ is an *isomorphism* between A and B if f is a bijection and $f(A) = B$. Note that in case of such an isomorphism, we also have $f^{-1}(B) = A$, and, moreover A and B are equal up to the bijective renaming f of non-constant values. A *database* D for a schema $\mathcal{R}$ is a finite set of $\mathcal{R}$-atoms with arguments from Δ. A *relational structure* (or simply *structure*) for a schema $\mathcal{R}$ is a (possibly infinite) set of $\mathcal{R}$-atoms with arguments from $\Delta \cup \Delta_N \cup \mathcal{V}$. An *interpretation* for a schema $\mathcal{R}$ is a structure whose atoms have arguments from $\Delta \cup \Delta_N$ only. So, every interpretation is a structure but not vice versa.

2.2 Acylic Structures and Treewidth

A *join forest* $JF(I)$ for a (possibly infinite) structure I is a forest whose vertices are the atoms of I, such that the following *connectedness condition* applies: Whenever an element $e \in dom(I)$ occurs in two atoms α and β of I, then α and β are in the same tree of the forest $JF(I)$, and e occurs in each atom of the unique path linking α and β in that tree. That is, the connectedness condition for e means that the set of nodes in which e occurs induces a connected subtree of $JF(I)$. We say that I is *acyclic* if there exists a join forest $JF(I)$ for I.

This definition of acyclicity is based on References [20, 21]. It is the most general type of acyclicity that has been proposed in the literature and coincides with α-*acyclicity* as defined by Fagin [52], which, in turn, is based on the acyclicity concept now known as *GYO acyclicity* originally proposed by Reference [73] and independently by Reference [117].

If $A \in dom(I)$ is a set of elements from the domain of I, then an $[A]$-*join forest* is defined as join forest, except that instead of requiring the connectedness condition for each $e \in dom(I)$, we only require it for each $e \in dom(I) \setminus A$. We say that I is $[A]$-*acyclic* if it has an $[A]$-*join forest*.

The *Gaifman graph* $G(I) = (V(I), E(I))$ of a (possibly infinite) structure I is the (possibly infinite) undirected graph whose set of nodes $V(I)$ is equal to $dom(I)$, and whose edges are all pairs $\{u, v\} \subseteq dom(I)$ such that $u \neq v$ and u and v jointly occur in some atom of I. Given a graph $\mathcal{G} = (V, E)$, a *tree decomposition* of $\mathcal{G}$ is a pair (T, λ), where $T = (N, A)$ is a tree having a set of nodes N and a set of arcs A, and λ is a labeling function $\lambda : N \to 2^V$ such that:

(i) for every $v \in V$, there is some $n \in N$ such that $v \in \lambda(n)$; that is, $\lambda(N) = \bigcup_{n \in N} \lambda(n) = V$;
(ii) for every edge $e = \{v_1, v_2\} \in E$, there is some $n \in N$ such that $\{v_1, v_2\} \subseteq \lambda(n)$; and
(iii) for every $v \in V$, the set $\{n \in N \mid v \in \lambda(n)\}$ induces a connected subtree in T.

The *width* of a tree decomposition $\mathcal{D} = (T, \lambda)$ is defined by $tw(\mathcal{D}) = \max\{|\lambda(n)| - 1 \mid n \in \mathbb{N}\}$. The *treewidth* of a graph $\mathcal{G} = (V, E)$, denoted $tw(G)$, is the minimum width of all tree decompositions of G. Given an interpretation (or equivalently, a structure) I, its treewidth $tw(I)$ is defined as the treewidth of its Gaifman graph: $tw(I) = tw(G(I))$. The treewidth of an infinite interpretation I may be infinite, in which case we write $tw(I) = \infty$.

It is well-known and not difficult to see that the treewidth of an acyclic structure I is at most $a-1$, where a is the maximum arity of any atom in I, or equivalently, $\mathrm{ar}(\mathcal{R})-1$, where $\mathcal{R}$ is the schema of I. Moreover, the treewidth of an $[A]$-acyclic structure I is at most $tw(A)+a-1$, where a is defined as above.

2.3 Unions of Conjunctive Queries (UCQs)

A ***conjunctive query (CQ)*** over a schema $\mathcal{R}$ is a formula $Q(\mathbf{x}) = \exists \mathbf{y}\, \Phi(\mathbf{x},\mathbf{y})$, where $\Phi(\mathbf{x},\mathbf{y})$ is a conjunction of $\mathcal{R}$-atoms, which have as arguments variables in $\mathbf{x}\cup\mathbf{y}$ and constants (but no nulls), and each variable in $\mathbf{x}$ occurs in some atom in $\Phi(\mathbf{x},\mathbf{y})$. The variables in $\mathbf{x}$ are the *answer variables of* Q. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ over $\mathcal{R}$ without answer variables (i.e., all variables are existentially quantified). We often write a BCQ as the set of all its atoms, omitting the quantifiers. A *union of CQs* (*UCQ*) over $\mathcal{R}$ has the form $Q(\mathbf{x}) = \bigvee_{i=1}^{n} Q_i(\mathbf{x})$, where $n \geqslant 1$, and every $Q_i(\mathbf{x})$ with $1 \leqslant i \leqslant n$ is a CQ over $\mathcal{R}$. The CQs $Q_i(\mathbf{x})$ are called the *disjuncts* of Q. Note that each disjunct of Q has the same answer variables, which we also define to be the answer variables of Q. A ***union of BCQs (UBCQ)*** over $\mathcal{R}$ is a UCQ over $\mathcal{R}$ without answer variables. The *width* $\mathrm{wd}(Q)$ of a UCQ Q is the maximal number of atoms in a disjunct of Q.

Answers to UCQs relative to an interpretation I are defined via homomorphisms as follows: The set of all *answers* to a CQ $Q(\mathbf{x}) = \exists \mathbf{y}\, \Phi(\mathbf{x},\mathbf{y})$ over an interpretation I, denoted $Q(I)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism h such that $h(\Phi(\mathbf{x},\mathbf{y})) \subseteq I$ and $h(\mathbf{x}) = \mathbf{t}$. Such answer tuples, which cannot contain null values from Δ_N, are usually called the *certain answers to* Q in the literature. Returning only such certain answers to the user is the most common approach taken when it comes to query answering in presence of uncertain values such as null values [54, 111]. One important advantage of this convention is that in situations, as in the present article, where a finite original database D may be extended by existential rules to an infinite interpretation I containing infinitely many atoms with null values, then the answer $Q(I)$ to any query Q on I is still finite and consists of facts built from known entities only. As will be discussed in later sections, in case of multiple models, the certain answers to a query are the certain answers that are true in all models. The *answer* to a BCQ $Q() = \exists \mathbf{y}\, \Phi(\mathbf{y})$ over an interpretation I is *Yes*, denoted $I \models Q$, if $Q(I) \neq \emptyset$, i.e., if a homomorphism $h\colon \Phi(\mathbf{y}) \to \Delta \cup \Delta_N$ exists such that $h(\Phi(\mathbf{y})) \subseteq I$. The set of all *answers* to a UCQ $Q(\mathbf{x}) = \bigvee_{i=1}^{n} Q_i(\mathbf{x})$ over an interpretation I, denoted $Q(I)$, is the union of all $Q_i(I)$ with $1 \leqslant i \leqslant n$. The *answer* to a UBCQ $Q() = \bigvee_{i=1}^{n} Q_i()$ over an interpretation I is *Yes*, denoted $I \models Q$, if $Q(I) \neq \emptyset$. Note that the definitions of query answers also define query answers over databases D, as every database is an interpretation.

A BCQ of the form $\exists \mathbf{y}(R_1(\mathbf{y_1}) \wedge R_2(\mathbf{y_2}) \wedge \cdots \wedge R_k(\mathbf{y_k}))$ is *acyclic* if the structure $\{R_1(\mathbf{y_1}), R_2(\mathbf{y_2}), \ldots, R_k(\mathbf{y_k})\}$ is acyclic. A UBCQ is acyclic if each of its disjuncts is. While general BCQs are NP-hard to evaluate, acyclic BCQs and UBCQs are polynomial-time decidable, and their evaluation is highly parallelizable [67]. Moreover, whether or not a query is acyclic can be recognized in linear time [109].

2.4 Tuple-generating Dependencies (TGDs) and the Chase

Tuple-generating dependencies (TGDs) describe constraints on interpretations in the form of generalized Datalog rules that may contain existentially quantified conjunctions of atoms in rule heads. Their syntax and semantics are as follows. Given a schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ over $\mathcal{R}$ has the form $\forall \mathbf{x} \forall \mathbf{y}\, (\Phi(\mathbf{x},\mathbf{y}) \to \exists \mathbf{z}\, \Psi(\mathbf{x},\mathbf{z}))$, where $\Phi(\mathbf{x},\mathbf{y})$ and $\Psi(\mathbf{x},\mathbf{z})$ are conjunctions of $\mathcal{R}$-atoms with arguments from $\mathcal{V}$ (i.e., we assume that TGDs do not contain constants). We call $\Phi(\mathbf{x},\mathbf{y})$ and $\Psi(\mathbf{x},\mathbf{z})$ the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. As usual (see, e.g., References [32, 70]), we assume without loss of generality that the head of a TGD is a single atom of the form $P(\mathbf{x},\mathbf{z})$. An interpretation I *satisfies* σ, denoted $I \models \sigma$, if

whenever a homomorphism $h : body(\sigma) \to I$ exists, then h can be extended to a homomorphism $h'(\supseteq h) : atoms(\sigma) \to I$. In this case, note that, in particular, h' fulfills $h'(head(\sigma)) \in I$.

Definition 2.1 (Application of a TGD). Let $\sigma = \Phi(\mathbf{x}, \mathbf{y}) \to \exists \mathbf{z}\, P(\mathbf{x}, \mathbf{z})$ be a TGD of the form $\forall \mathbf{x} \forall \mathbf{y} \left(\Phi(\mathbf{x}, \mathbf{y}) \to \exists \mathbf{z}\, P(\mathbf{x}, \mathbf{z})\right)$, and let I be an interpretation.

- We say that σ *is applicable to* I if there exists a homomorphism $h : body(\sigma) \to I$, that maps all atoms in $body(\sigma)$ to atoms in I. We then also say that σ *is applicable to* I *with* h.
- Let σ be applicable to I with homomorphism h. Applying σ to I with h yields a new interpretation I' that is obtained from I by adding $h'(P(\mathbf{x}, \mathbf{z}))$, where h' is an extension of h that assigns to each variable from $\mathbf{z}$ a distinct fresh null. We call I' the *result* of applying σ to I with h. ◁

Definition 2.2 (Chase Sequence). Let I be an interpretation and Σ a set of TGDs. A *chase sequence* of I with Σ is a potentially infinite sequence $I_0, I_1, I_2, \ldots$ of interpretations such that $I_0 = I$, each I_{i+1} is a result of applying a TGD from Σ to I_i, and for each TGD $\sigma \in \Sigma$ that is applicable to some I_i with a homomorphism h, there exists exactly one index $j \geqslant 0$ such that I_{j+1} is the result of applying σ to I_j with h. We call $I_0 \cup I_1 \cup I_2 \cup \cdots$ the *result* of the chase sequence. For an interpretation I, a TGD set Σ, and a chase sequence ξ based on Σ and starting with I, the result of ξ is denoted by $chase(I, \Sigma, \xi)$. ◁

Remark 2.3. In the above definition, the requirement that for every i or each TGD $\sigma \in \Sigma$ applicable to I_i with a homomorphism h, there exists one index $j \geqslant 0$ such that I_{j+1} is the result of applying σ to I_j with h is usually called a *fairness* condition. This condition guarantees that at the step i of the chase, when I_i is considered, whenever a TGD σ is applicable with a homomorphism h, then this TGD must have been applied either already at some earlier step $j < i$, or is applied precisely at step i, in which case $j = i$, or is guaranteed to be applied at some future step $j > i$. In addition, there needs to be *exactly one* such j only. This means that for each applicable TGD σ and homomorphism h, as before, σ is applied *only once* with h; that is, it is not allowed that there are any $j' \neq j$ such that the same σ is applied with the same homomorphism h to $I_{j'}$.

If I is an interpretation and Σ a set of TGDs, then we denote by (I, Σ) the logical theory $\Gamma = I \cup \Sigma$, where we treat null values from Δ_N as constants.[1]

Even though a set of TGDs Σ and an intial interpretation I may give rise to different chase sequences, the following proposition, which follows easily from Definition 2.2, is well-known [34, 74, 76, 97].

Proposition 2.4. *Let I be an interpretation, Σ be a set of TGDs, and ξ_1 and ξ_2 be chase sequences of I with Σ. Then, the results $chase(I, \Sigma, \xi_1)$ and $chase(I, \Sigma, \xi_2)$ are isomorphic models of (I, Σ) and differ only via a bijective renaming of null values.*

By the above proposition, for a given interpretation I and a given set Σ of TGDs, the result of each chase sequence of I with Σ is essentially the same. Therefore, we usually abstract from a specific chase sequence and simply denote this result by $chase(I, \Sigma)$, and sometimes refer to it as "the chase of I with Σ." Moreover, by "the chase," we refer to the procedure of constructing a chase sequence.

[1] Alternatively, we could define the logical theory via a formula in infinitary logic

$$\exists \mathbf{z}\Big(\bigwedge_{\alpha \in I} \alpha\Big) \wedge \bigwedge_{\sigma \in \Sigma} \sigma,$$

where $\mathbf{z}$ is the (possibly infinite) list of all null values occurring in I. (In case I is a database, $\mathbf{z}$ is obviously empty.)

The chase has been used for practical database query answering and as a theoretical tool for deriving decidability and complexity results. Its crucial property related to query answering is that each chase sequence result for an interpretation I with a set Σ of TGDs constitutes a *universal model* for the logical theory (I, Σ). That is, there is a homomorphism from $chase(I, \Sigma)$ to each model of (I, Σ). This was shown for a different version of the chase (the restricted chase) in Reference [54], and for the version used here (the oblivious chase) in Reference [32]. As a consequence, the following is well-known:

Proposition 2.5. *Let D be a database, Σ a set of TGDs, and Q a UBCQ. Then, $(I, \Sigma) \models Q$ iff $chase(I, \Sigma) \models Q$.*

We say that a TGD σ is *guarded* if its body contains an atom, called *guard atom* (or *guard*) of σ, that contains all variables that occur in the body of σ. If σ is guarded, then we denote by $guard(\sigma)$ any fixed guard of σ (e.g., the leftmost one). It is easy to see that when guarded rules are chased over an instance I, then the overall treewidth of the chase result $chase(I, \Sigma)$ cannot be significantly larger than the treewidth of I, and thus, chasing guarded TGDs preserves bounded treewidth. This is stated formally in the following proposition, which is based on Lemma 3.11 of Reference [32] and a slight refinement of Lemma 3.13 of Reference [32].

Proposition 2.6 (Bounded Treewidth Preservation Property; based on Reference [32]). *Let I be a finite interpretation and Σ a set of guarded TGDs, then $chase(I, \Sigma)$ is $[dom(I)]$-acyclic and has a finite treewidth. In particular, $tw(chase(I, \Sigma)) \leqslant tw(dom(I)) + w - 1 < |dom(I)| + w$, where w is the maximum predicate arity[2] in Σ.*

2.5 Equality-generating Dependencies (EGDs) and Negative Constraints (NCs)

An *equality-generating dependency (EGD)* σ is a first-order formula of the form $\forall \mathbf{x} \left(\Phi(\mathbf{x}) \rightarrow x_i = x_j\right)$, where $\Phi(\mathbf{x})$ is a conjunction of atoms (without nulls), called the *body* of σ, denoted $body(\sigma)$, and x_i and x_j are variables from $\mathbf{x}$. We call $x_i = x_j$ the *head* of σ, denoted $head(\sigma)$. Such σ is *satisfied* in an interpretation I for $\mathcal{R}$, denoted $I \models \sigma$, if, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{x})) \subseteq I$, it holds that $h(x_i) = h(x_j)$.

Another crucial ingredient of Datalog$^{\pm}$ for ontological modeling consists of *negative constraints* (*NCs*, or simply *constraints*) ν, which are first-order formulas $\forall \mathbf{x} \left(\Phi(\mathbf{x}) \rightarrow \bot\right)$, where $\Phi(\mathbf{x})$ is a conjunction of atoms (without nulls), called *body* of ν, denoted $body(\nu)$. An interpretation I *satisfies* ν, denoted $I \models \nu$, if no homomorphism h satisfies $h(\Phi(\mathbf{x})) \subseteq I$.

For a set Σ of TGDs, EGDs, and NCs, we write $I \models \Sigma$ if $I \models \sigma$ for all $\sigma \in \Sigma$. We usually omit the universal quantifiers in TGDs, EGDs, and NCs, and all sets of TGDs, EGDs, and NCs are implicitly assumed to be finite here. *Query answering* under TGDs, EGDs, and NCs, i.e., the evaluation of UCQs and UBCQs on databases D for $\mathcal{R}$ under a set Σ of TGDs, EGDs, and NCs on $\mathcal{R}$ is defined as follows: The set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all interpretations I such that (i) $D \subseteq I$ and (ii) $I \models \Sigma$. The set of *answers* for a UCQ Q over D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{t}$ such that $\mathbf{t} \in Q(I)$ for all $I \in mods(D, \Sigma)$. The *answer* for a UBCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, if $ans(Q, D, \Sigma) \neq \emptyset$.

2.6 Decision Problems Considered and Complexity Measures

Computing the answer to a non-Boolean query is a computation (or search) problem, rather than a decision problem. Moreover, the answer to even very simple queries can be of exponential size. For

[2]These bounds apply for single-head TGDs as assumed in this article; for multi-head TGDs, the proposition is still correct when w denotes the maximum over all rules $\sigma \in \Sigma$ of the sum of all arities of the head atoms of σ.

example, if $R = \{a, b\}$ is a unary database relation, then the CQ $Q(x_1, \ldots, x_n\} = R(x_1) \wedge \cdots \wedge R(x_n)$ has an answer comprising 2^n tuples. For these reasons, in the context of data and knowledge bases, one usually considers the following ***query-of-tuple (QOT)*** decision problem: For a database D, possibly with a set Σ of rules, and a fixed inference relation $\models^*$ that gives a concrete meaning to "$(D, \Sigma) \models^* Q(a_1, \ldots, a_r)$," where $a_1, \ldots, a_r \in \Delta$, decide whether actually $(D, \Sigma) \models^* Q(a_1, \ldots, a_r)$. Here, $Q(a_1, \ldots, a_r)$ denotes that the tuple $\langle a_1, \ldots, a_r \rangle$ is an answer to Q. The QOT problem is logspace-equivalent to the problem of Boolean query answering. In fact, for all entailment relations "$\models^*$" used in the present article, the QOT problem "$(D, \Sigma) \models^* Q(a_1, \ldots, a_r)$" is equivalent to the problem "$(D', \Sigma) \models^* \exists x_1 \ldots \exists x_r (Adom(x_1) \wedge \cdots \wedge Adom(x_r) \wedge Q(x_1, \ldots, x_r))$," where D' is the database resulting from D by adding a unary relation $Adom$, which contains all data constants (i.e., elements from Δ) present in D, and which thus comprises the so-called *active domain* of D. For this reason, we only derive complexity results for Boolean query evaluation, keeping in mind that the same results also hold for the QOT problem.

In analogy to concepts introduced in Reference [112], when analyzing the complexity of deciding $(D, \Sigma) \models^* Q$, we distinguish between the *combined complexity* where D, Σ, and Q are all three the input parameters, and the *data complexity*, where the database D is the only input parameter, and $\mathcal{R}$, Σ, and Q are fixed. When relevant, we will also study the setting when D and Q are both the input parameters, and Σ is fixed.

3 STABLE MODEL SEMANTICS FOR TGDS WITH NEGATION

This section describes the stable model semantics for TGDs with negation. We first present an extension of TGDs, called *normal TGDs*, that allows for negated atoms to occur in their bodies. Negation in front of atoms will then be interpreted in a non-monotonic fashion by restricting attention to the *stable models* of a set of normal TGDs. The notion of a stable model originated in the logic programming community [62], where it was used to give semantics to logic programs with negated atoms in rule bodies, which are evaluated using negation as failure. Here, we lift it to sets of normal TGDs by first Skolemizing existential quantifiers in the heads of such TGDs and then taking the stable models of the resulting logic program as the semantics of the set of normal TGDs. Alongside, we also fix some terminology and notation that will be used in the rest of this article.

3.1 Normal TGDs

Normal TGDs extend TGDs to allow for negated atoms (also called *negative literals*) to occur in their bodies. Given a schema $\mathcal{R}$, a *literal over* $\mathcal{R}$ (or $\mathcal{R}$*-literal*) is either an $\mathcal{R}$-atom α, called *positive* literal, or the negation $\neg\alpha$ of an $\mathcal{R}$-atom α, called *negative* literal. If $\mathcal{R}$ is irrelevant or clear from the context, then we drop $\mathcal{R}$ and speak of a *literal* only. A *normal TGD* (*NTGD*, for short) over $\mathcal{R}$ has the form

$$\sigma = \forall \mathbf{x} \forall \mathbf{y} \big(\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \Psi(\mathbf{x}, \mathbf{z}) \big), \tag{1}$$

where $\Phi(\mathbf{x}, \mathbf{y})$ is a conjunction of $\mathcal{R}$-literals with variables as their arguments, and $\Psi(\mathbf{x}, \mathbf{z})$ is a conjunction of $\mathcal{R}$-atoms with variables as their arguments. Such an NTGD is also abbreviated as $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \Psi(\mathbf{x}, \mathbf{z})$. As with TGDs, we call $\Phi(\mathbf{x}, \mathbf{y})$ and $\Psi(\mathbf{x}, \mathbf{z})$ the *body* and the *head* of σ, respectively, and, as for TGDs (see References [32, 70]), we assume without loss of generality that the head is a single atom. We denote by $body^+(\sigma)$ the set of all atoms that occur positively in the body of σ, by $body^-(\sigma)$ the set of all atoms that occur negated in the body of σ, and by $head(\sigma)$ the atom in the head of σ.

In this article, we are mostly interested in *guarded* NTGDs: An NTGD σ is *guarded* if its body has a positive literal, called a *guard* of σ, that contains all universally quantified variables of σ (i.e.,

all body variables of σ). If σ is guarded, then $guard(\sigma)$ denotes a fixed guard of σ (e.g., the leftmost one).

As for the semantics, an NTGD σ of the form (1) is *satisfied* in an interpretation I for $\mathcal{R}$ if, whenever there exists a homomorphism h with $h(body^+(\sigma)) \subseteq I$ and $h(body^-(\sigma)) \cap I = \emptyset$, then there exists a homomorphism $h' \supseteq h_{|\Phi(\mathbf{x},\mathbf{y})}$ with $h'(head(\sigma)) \in I$. The semantics of sets of NTGDs will be defined in terms of stable models. To this end, we lift the notion of a stable model from logic programming to sets of NTGDs. Before we do this, we review stable models in the context of logic programming.

3.2 Stable Model Semantics for Logic Programming

Here, we briefly recall the notion of a stable model from logic programming. For an in-depth exposition of the stable model semantics for logic programming, we refer the reader to Reference [62].

Stable models give semantics to *normal (logic) programs*, which are logic programs with negated atoms in rule bodies. We start by giving a more precise definition of normal programs. Let $\mathcal{F}$ be a set of function symbols. An $\mathcal{F}$*-term* t (or simply a *term*, when $\mathcal{F}$ is clear from the context) is either a variable, or a constant from Δ, or of the form $f(t_1, \ldots, t_m)$, where f is an m-ary function symbol in $\mathcal{F}$, and each t_i with $1 \leqslant i \leqslant m$ is an $\mathcal{F}$-term. In normal programs, the terms in atoms are allowed to be $\mathcal{F}$-terms. Formally, in the context of normal programs, an $\mathcal{R}$-atom has the form $R(t_1, \ldots, t_m)$, where R is an m-ary predicate in $\mathcal{R}$, and the t_i's are $\mathcal{F}$-terms. A *normal rule* (or simply *rule*) is of the form

$$\beta_1, \ldots, \beta_n, \text{not } \beta_{n+1}, \ldots, \text{not } \beta_{n+m} \rightarrow \alpha\,, \tag{2}$$

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 we refer to the conjunction $\beta_1, \ldots, \beta_n, \text{not } \beta_{n+1}, \ldots, \text{not } \beta_{n+m}$ as 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* if $m = 0$. We say r is a *fact* if $m = n = 0$. A *normal* (respectively, *positive*) *program* P is a finite set of normal (respectively, positive) rules. We denote by P^+ the positive program obtained from P by dropping all negated atoms (i.e., those prefixed with "not") from rule bodies.

A normal program is interpreted with respect to a subset of its Herbrand base. Let P be a normal program, and let $\mathcal{F}_P$ be the set of function symbols occurring in P. The *Herbrand universe* HU_P for P is the set of all ground $\mathcal{F}_P$-terms, where an $\mathcal{F}_P$-term is *ground* if it does not contain any variables. The ground terms of HU_P are subject to the ***unique name assumption (UNA)***, which requires that different ground terms are (or refer to) different entities and cannot be unified. Thus, if a and b are data constants from Δ, then $a \neq b$ is valid, and if f and g are function symbols, then $f(a) \neq f(b)$, $f(a) \neq g(a)$, and so on. The *Herbrand base* HB_P for P is the set of all atoms with arguments in HU_P. A *Herbrand interpretation* I for P is a subset of HB_P.

To define what it means for such a Herbrand interpretation to be a model of P, we consider the ground instances of all the rules in P. 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 use $ground(r)$ (respectively, $ground(P)$) to denote the set of all ground instances of r (respectively, rules in P). A Herbrand interpretation I is a *Herbrand model* of a ground rule r if $body^+(r) \subseteq I$ and $body^-(r) \cap I = \emptyset$ implies $\alpha \in I$. It is a *Herbrand model* of P if it is a Herbrand model of every $r \in ground(P)$.

We are now ready to define stable models of normal programs as follows: The ***Gelfond-Lifschitz reduct (GL reduct)*** of a normal program P relative to *a Herbrand interpretation* I for P, denoted P^I, is the (possibly infinite) ground positive program that is obtained from $ground(P)$ by

(1) deleting every rule r such that $B^-(r) \cap I \neq \emptyset$, and
(2) deleting all negated atoms from each remaining rule.

Since P^I is positive, it has a unique minimal Herbrand model. A *stable model* of a normal program P is *a Herbrand interpretation* I for P such that I is the minimal Herbrand model of P^I. Note that every stable model of P is also a minimal Herbrand model of P.

Example 3.1. Consider the normal program consisting of the following rules:

$$\begin{aligned} &\rightarrow P(0),\\ &\rightarrow P(s(0)),\\ P(x), \text{not } Q(x) &\rightarrow Q(s(x)),\\ Q(x), \text{not } P(x) &\rightarrow P(s(x)). \end{aligned}$$

It is not difficult to verify that this program has two stable models, namely, the finite model $I_1 = \{P(0), P(s(0)), Q(s(0))\}$ and the infinite model $I_2 = \{P(0), Q(0), P(s(0)), Q(s(s(0))), P(s(s(s(0)))), \ldots\}$.

3.3 Stable Model Semantics for Sets of Normal TGDs

We now lift the definition of stable models for normal programs to sets of NTGDs. To this end, we first use Skolemization to translate each NTGD into a normal rule and then view the stable models of the resulting normal program as the semantics of the set of NTGDs.

Consider an NTGD $\sigma = \Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \Psi(\mathbf{x}, \mathbf{z})$, where $\Phi(\mathbf{x}, \mathbf{y}) = \beta_1 \wedge \cdots \wedge \beta_n \wedge \neg\beta_{n+1} \wedge \cdots \wedge \neg\beta_{n+m}$ and $\mathbf{z} = (z_1, \ldots, z_k)$. The *functional transformation* of σ is the normal rule σ^f defined as

$$\beta_1, \ldots, \beta_n, \text{not } \beta_{n+1}, \ldots, \text{not } \beta_{n+m} \rightarrow \Psi(\mathbf{x}, \mathbf{f}_\sigma(\mathbf{x}, \mathbf{y})),$$

where $\mathbf{f}_\sigma(\mathbf{x}, \mathbf{y}) = (f_{\sigma,z_1}(\mathbf{x}, \mathbf{y}), \ldots, f_{\sigma,z_k}(\mathbf{x}, \mathbf{y}))$, and each f_{σ,z_i}, $1 \leqslant i \leqslant k$, is a function symbol for σ and z_i. We also need to cover the special case where both $\mathbf{x}$ and $\mathbf{y}$ are empty, and where thus σ is of the form "$body \rightarrow \exists z_1, \ldots, z_k\, R(z_1, \ldots, z_k)$," such that $body$ is either empty or contains propositional symbols (i.e., zero-ary predicates) only. In this case, σ^f is defined to be the logic programming rule "$body \rightarrow R(f_{\sigma,z_1}(0), \ldots, f_{\sigma,z_k}(0))$," where 0 is assumed to belong to the domain of constants Δ.

Given a set Σ of NTGDs, the *functional transformation* of Σ, denoted Σ^f, is obtained from Σ by replacing each $\sigma \in \Sigma$ by σ^f.

Example 3.2. Consider the following set Σ of (guarded) NTGDs:

$$Node(x) \wedge \neg Parent(x) \rightarrow End(x), \tag{1}$$
$$Node(x) \wedge \neg End(x) \rightarrow Parent(x), \tag{2}$$
$$Children(x, y, z) \rightarrow Node(u), \qquad \text{for each } u \in \{y, z\}, \tag{3}$$
$$\sigma_4 \colon Node(x) \wedge Parent(x) \rightarrow \exists y \exists z\, Children(x, y, z). \tag{4}$$

The functional transformation Σ^f of Σ is the following normal program:

$$Node(x), \text{ not } Parent(x) \rightarrow End(x), \tag{5}$$
$$Node(x), \text{ not } End(x) \rightarrow Parent(x), \tag{6}$$
$$Children(x, y, z) \rightarrow Node(u), \qquad \text{for each } u \in \{y, z\}, \tag{7}$$
$$\sigma_4^f \colon Node(x), Parent(x) \rightarrow Children(x, f_{\sigma_4,y}(x), f_{\sigma_4,z}(x)). \tag{8}$$

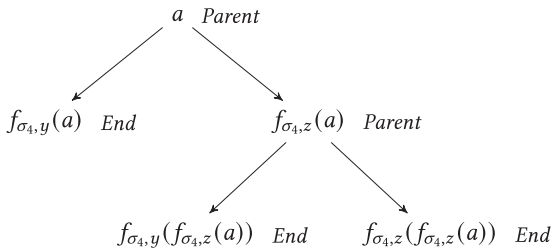
Note that rules (5)–(8) are essentially the NTGDs (1)–(4), with the only difference being the different representations of conjunction and negation, and with the existentials replaced by appropriate Skolem terms.

The stable model semantics of a set of NTGDs is now defined in terms of the stable models of its functional transformation as follows:

Definition 3.3. Let Σ be a finite set of NTGDs over a schema $\mathcal{R}$, and let D be a database for Σ. A *stable model* for D and Σ is a stable model of the normal program $P = D^f \cup \Sigma^f$, having $D^f = \{\rightarrow \alpha \mid \alpha \in D\}$ as a collection of ground facts. The set of all *stable models for D and Σ* is denoted by $SMod(D, \Sigma)$. Instead of "stable model for D and Σ," we may also say "stable model for (D, Σ)." ◁

In the sequel, we do not explicitly distinguish between the database D and the corresponding collection D^f of ground facts (i.e., we simply write D instead of D^f).

Example 3.4. Consider the set Σ of NTGDs from Example 3.2 and the database $D = \{Node(a)\}$. Each stable model I for D and Σ (equivalently, the stable models of $P = D \cup \Sigma^f$) can be thought of as a binary branching tree rooted at a, where branches can either be infinite or finite. For example, the following represents a stable model I for D and Σ, where each node represents an element b with $Node(b) \in I$, and each pair of edges from a node b to two distinct children c and d represents the atom $Children(b, c, d) \in I$:



Note that rules (5) and (6) simulate an *exclusive or*. The two rules jointly make a decision on whether a node b should be a leaf ($End(b) \in I$) or whether b should be an inner node ($Parent(b) \in I$). Moreover, the UNA ensures that nodes marked as inner nodes have exactly two children, because the Skolem terms $f_{\sigma_4,y}(f_{\sigma_4,z}(a))$ and $f_{\sigma_4,z}(f_{\sigma_4,z}(a))$ are different elements of the Herbrand universe. Moreover, only *Parent* nodes are expanded by rule σ_4^f, as there is no rule that would assign children to an *End* node. Hence, the latter are the leaves of the tree, and we obtain a finite model. Rule (7) marks all children of a node as nodes, so the process continues recursively, and every time a node is constructed, this node can in one model be an *End* node and will not be further expanded, and in another one a parent node with two children. Σ^f on $D = \{Node(a)\}$ has therefore an infinite set of finite stable models and also an infinite set of infinite stable models.

Let Σ be a set (i.e., conjunction) of TGDs and Σ^f the corresponding negation-free logic program. Then, by results of classical logic programming (see, e.g., Reference [86]), for each database D, $D \cup \Sigma^f$ has only a single model, which is the unique least Herbrand model of (D, Σ^f). It is well-known that this model can be computed by a least fixpoint iteration of a monotonic transformation T_Σ that essentially fires rules as long as they are applicable with some new homomorphism.

Proposition 3.5. *If D is a database, Σ a set of TGDs, then up to isomorphism, chase(D, Σ) coincides with the unique stable model of D with Σ^f, i.e., with the unique model in $SMod(D, \Sigma^f)$. For the specific chase sequence whose generated null values are taken to be the Skolem terms generated by the respective application of the rules of Σ^f on D, this isomorphism becomes the equality relation.*

Our main interest in this article is the problem of answering queries with respect to NTGDs under the stable model semantics. Since the stable model semantics allows us to reason about negative information in a meaningful way, it makes sense to extend the standard query language of UBCQs so we can also ask for the absence of atoms. Here, we first present such an extension and then show that answering such queries relative to sets of *guarded* NTGDs under the stable model semantics is decidable.

We start by extending conjunctive queries to allow for negative literals in their bodies. A ***normal conjunctive query (NCQ)*** over a schema $\mathcal{R}$ is a formula of the form

$$Q(\mathbf{x}) = \exists \mathbf{y}\, \Phi(\mathbf{x}, \mathbf{y}),$$

where $\Phi(\mathbf{x}, \mathbf{y})$ is a conjunction of $\mathcal{R}$-literals, whose arguments are variables and constants, and each variable from $\mathbf{x}$ occurs in some positive literal in $\mathbf{x}$. In technical arguments, we often tacitly assume without loss of generality that Q does not contain any constants. The variables in $\mathbf{x}$ are called the *answer variables of* Q. We call Q a ***normal Boolean conjunctive query (NBCQ)*** if Q has no answer variables (i.e., if all variables are existentially quantified). We denote by Q^+ (respectively, Q^-) the set of all atoms α such that α occurs positively in Q (respectively, $\neg\alpha$ occurs in Q). Of particular importance are *covered* NCQs Q, which have the property that for every negative literal $\neg\alpha$ in Q there exists a positive literal (i.e., an atom) β in Q such that every variable and every constant in α occurs in β. For example, the NCQ $Q(x) = \exists y\,(R(x, y) \wedge \neg S(y))$ is covered, while $Q'(x, z) = \exists y\,(R(x, y) \wedge R(y, z) \wedge \neg S(x, z))$ is not covered. Observe that the coveredness of Q implies also the safeness of Q, but not vice versa. Covered NBCQs are, in a precise sense, a perfect match to guarded NTGDs, as their negative atoms can actually be simulated via guarded NTGDs, and therefore, covered (U)NBCQs can be transformed into plain (U)BCQs via a minor modification of a given guarded NTGD set Σ. Details will be given in Section 5.2.4 (in a broader context).

We are mostly interested in answering ***unions of NCQs (UNCQs)*** and ***unions of NBCQs (UNBCQs)***. A *UNCQ* is a disjunction $Q(\mathbf{x}) = Q_1(\mathbf{x}) \vee \cdots \vee Q_n(\mathbf{x})$ of NCQs $Q_i(\mathbf{x})$, $1 \leqslant i \leqslant n$, all with the same answer variables, where $n \geqslant 1$. We call $\mathbf{x}$ the *answer variables* of Q, and the NCQs $Q_i(\mathbf{x})$ are the *disjuncts* of Q. A *UNBCQ* is a Boolean UNCQ or, equivalently, a UNCQ without answer variables. A UNCQ is *covered* if each of its disjuncts is covered. The *width* of a UNCQ Q, denoted $\mathrm{wd}(Q)$, is the maximal number of atoms in a disjunct of Q. In Section 5, we also refer to *acyclic UNBCQ*, which are UNBCQs that can be transformed into acyclic UBCQs by replacing each negative literal $\neg\alpha$ by the positive literal α.

The set of all *answers* to an NCQ $Q(\mathbf{x}) = \exists \mathbf{y}\, \Phi(\mathbf{x}, \mathbf{y})$ over an interpretation I, denoted $Q(I)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism h with $h(Q^+) \subseteq I$, $h(Q^-) \cap I = \emptyset$, and $h(\mathbf{x}) = \mathbf{t}$. This definition extends to NBCQs, UNCQs, and UNBCQs in exactly the same way as in the positive case. The answers of UNCQs relative to sets of NTGDs under the stable model semantics is defined in terms of certain answers over stable models:

Definition 3.6. Let Σ be a set of NTGDs over $\mathcal{R}$, and let D be a database for $\mathcal{R}$. The set of all *answers* for a UNCQ Q over D and Σ under the stable model semantics is the set $ans_{stable}(Q, D, \Sigma)$ of all tuples $\mathbf{t}$ such that $\mathbf{t} \in Q(I)$ for all $I \in SMod(D, \Sigma)$. If Q is a UNBCQ, then we write $(D, \Sigma) \models_{stable} Q$ to denote the fact that $ans_{stable}(Q, D, \Sigma) \neq \emptyset$, that is, Q is satisfied in all stable models for D and Σ. ◁

Under the stable model semantics for a database D and a set Σ of NTGDs, we obtain different answers to queries than under the classical first-order semantics of D and Σ. For instance, let $\mathcal{R} = \{p, q\}$ be a schema where p and q are two different propositional letters (i.e., zero-ary predicate symbols), and let D be empty and Σ contain only the single NTGD $\neg p \rightarrow p$. It is well-known and easy to see that (D, Σ) has no stable model, thus p and q are vacuously true in all stable models,

and $(D, \Sigma) \models_{stable} q$. However, while $(D, \Sigma) \models p$, it clearly holds that $(D, \Sigma) \not\models q$. Thus, the stable model semantics differs from the classical one even in case of positive queries, when negations are allowed to occur in rule bodies.

However, when negation occurs neither in rule bodies nor in rule heads, the stable model semantics coincides with the first-order semantics. The following proposition is an immediate consequence of Proposition 3.5:

Proposition 3.7. *Let Σ be a set of TGDs, denote by Σ^f the corresponding logic program, and let D be a database. Then, for any UBCQ Q, we have that $(D, \Sigma^f) \models_{stable} Q$ iff chase$(D, \Sigma) \models Q$ iff $(D, \Sigma) \models Q$.*

As the following example shows, a similar result does not hold for UNBCQs, and actually not even for NBCQs.

Example 3.8. Consider an empty database $D = \emptyset$ and the set (conjunction) Σ consisting of the following two TGDs:

$$true \rightarrow \exists y \exists z\, R(y, z),$$
$$\forall x\, (R(x, x) \rightarrow hello).$$

Its stable models are, by definition, those of the logic program Σ^f:

$$\sigma_1 : \quad \rightarrow R(f_{\sigma_1, y}(0), f_{\sigma_1, z}(0)),$$
$$\sigma_2 : \quad R(x, x) \rightarrow hello.$$

Observe that (D, Σ) has, among others, the two models $M_1 = \{R(a, b)\}$ and $M_2 = \{R(a, a), hello\}$, where a and b are some domain elements in Δ. M_2 arises in case a is used as witness for both y and z. However, $D \cup \Sigma^f$ is a classical logic program and has as such only one stable model: $M_0 = \{R(f_{\sigma_1, y}(0), f_{\sigma_1, z}(0))\}$. Via the UNA, the two arguments of the R-atom must be different. Thus, by the UNA, the interpretation $\{R(f_{\sigma_1, y}(0), f_{\sigma_1, y}(0)), hello\}$ is not a model. Now let Q be the query $\neg hello$. The latter is a covered UNBCQ, because *hello* is a propositional atom that has no variables that need to occur in any other literal. Q is satisfied in M_0 and M_1, but not in M_2. Therefore, $(D, \Sigma) \not\models Q$, but for the logic program $D \cup \Sigma^f$, we have $D \cup \Sigma^f \models_{stable} Q$ and thus $(D, \Sigma) \models_{stable} Q$.

3.4 Comparison to Other Approaches to the Semantics of Stable Models for TGDs

The **stable models semantics (SMS)** adopted in the present article relies on the classical SMS for **logic programming (LP)** with function symbols over Herbrand interpretations with the UNA. This semantics was proposed in the well-known 1988 paper by Gelfond and Lifschitz [62]. Through grounding, this classical semantics is also applicable to LP with function symbols, giving rise to decision problems over infinite Herbrand universes that were studied, for example, in References [25, 26, 91, 92, 107]. We have explained, in the introduction, why we have chosen to base our SMS for NTGDs on the classical LP SMS. Here, we discuss the relationship of our approach to two major more recent approaches, which are actually shown to be semantically equivalent for NTGDs: (i) The equilibrium logic approach introduced and studied by Pearce in 1996 [98, 99], based on the so-called here-and-there logic, and further studied and extended in References [31, 100–102], and (ii) the second-order logic stable semantics, here abbreviated by *SOS semantics*, by Ferraris, Lee, and Lifschitz [56, 57], which is based on model minimization techniques similar to circumscription. Even though these semantics have been defined for full first-order logic, we here stick to the NTGD fragment only, because the 1988 LP semantics has not been defined for full first-order logic, and NTGDs are the formalism relevant to the present article. The first-order equilibrium logic here-and-there stable semantics approach [102] yields precisely the same stable model as the SOS approach;

see Proposition 4 of Reference [56]. For this reason, we here only compare the LP-based approach to the SOS approach, but retain that this comparison is also valid for quantified equilibrium here-and-there logic for NTGDs. Given the restriction to NTGDs, the definition of the SOS approach given here is somewhat simpler than for general first-order theories [57].

In this section, we make the somewhat subtle difference between a relational schema $\mathcal{R}$, which is a set, and an associated ordered list $\mathbf{R}$ of relation symbols (or equivalently, predicate variables or also predicate symbols). In formulas of second-order logic, such a list $\mathbf{R}$ is to be interpreted by *extensions* of $\mathbf{R}$, which is the content (value) of concrete relations associated with the list of relation symbols in $\mathbf{R}$. Such an extension is a list of relations, i.e., sets of tuples), where the relations are given in the same order as the symbols in $\mathbf{R}$, but where each single relation incorporates no reference to a relation symbol.

For a (sub-)formula Φ of a second-order formula, and a list of predicate variables $\mathbf{X}$, we write $\Phi(\mathbf{X})$ to denote that $\mathbf{X}$ are the free local variables of Φ, i.e., those free variables that occur in Φ only.

Consider a database D and a set of NTGDs Σ over a joint schema $\mathcal{R}$ with list of predicate variables $\mathbf{R} = (R_1, \ldots, R_r)$. Let $T(\mathbf{R})$ denote the first-order formula corresponding to the joint logical theory of D and Σ, i.e., $T = (D, \Sigma)$. Let $\mathbf{U} = (U_1, \ldots, U_r)$ be a list of distinct fresh predicate variables, where $\mathbf{U}$ is arity-compatible with $\mathbf{R}$, i.e., $\forall 1 \leqslant i \leqslant r \; ar(R_i) = ar(U_i)$.

From the theory $T = (D, \Sigma)$, we shall define a theory $T^{\mathbf{R}}(\mathbf{U})$ as follows: Let $D(\mathbf{U})$ denote the database resulting from D by replacing every symbol R_i with the relation symbol U_i. For an NTGD σ of the form

$$\forall \mathbf{x} \; \Big(R_{j_1}(\mathbf{x}_{\mathbf{j}_1}) \wedge \cdots \wedge R_{j_\ell}(\mathbf{x}_{\mathbf{j}_\ell}) \wedge \neg R_{j_{\ell+1}}(\mathbf{x}_{\mathbf{j}_{\ell+1}}) \wedge \cdots \wedge \neg R_{j_m}(\mathbf{x}_{\mathbf{j}_m}) \rightarrow \exists \mathbf{y} \, R_0(\mathbf{x_0}, \mathbf{y}) \Big) ,$$

where each $\mathbf{x}_{\mathbf{j}_i}$ is a list of variables also occurring in $\mathbf{x}$ and where all predicate symbols are from $\mathbf{R}$, let $\sigma^{\mathbf{R}}(\mathbf{U})$ denote the NTGD

$$\forall \mathbf{x} \; \Big(U_{j_1}(\mathbf{x}_{\mathbf{j}_1}) \wedge \cdots \wedge U_{j_\ell}(\mathbf{x}_{\mathbf{j}_\ell}) \wedge \neg U_{j_{\ell+1}}(\mathbf{x}_{\mathbf{j}_{\ell+1}}) \wedge \neg R_{j_{\ell+1}}(\mathbf{x}_{\mathbf{j}_{\ell+1}}) \wedge \cdots \\ \wedge \neg U_{j_m}(\mathbf{x}_{\mathbf{j}_m}) \wedge \neg R_{j_m}(\mathbf{x}_{\mathbf{j}_m}) \rightarrow \exists \mathbf{y} \, U_0(\mathbf{x_0}, \mathbf{y}) \Big) .$$

Then, define $\Sigma^{\mathbf{R}}(\mathbf{U}) = \{\sigma^{\mathbf{R}}(\mathbf{U}) \mid \sigma \in \Sigma\}$, and finally $T^{\mathbf{R}}(\mathbf{U}) = (D(\mathbf{U}), \Sigma^{\mathbf{R}}(\mathbf{U}))$. Intuitively, for each extension of $\mathbf{R}$, $T^{\mathbf{R}}(\mathbf{U})$ simulates the GL-reduct of $T(\mathbf{U})$ with $\mathbf{R}$, where $T(\mathbf{U})$ is obtained from $T(\mathbf{R})$ by replacing each symbol R of $\mathbf{R_i}$ by the corresponding symbol U_i from $\mathbf{U}$.

For predicate symbols X and Y of the same arity, $X \leqslant Y$ is an abbreviation for $\forall \mathbf{x} \, (X(\mathbf{x}) \rightarrow Y(\mathbf{x}))$, and $X < Y$ for $X \leqslant Y \; \wedge \; \exists \mathbf{x} \, (Y(\mathbf{x}) \wedge \neg X(\mathbf{x}))$. For lists $\mathbf{X} = (X_1, \ldots, X_k)$ and $\mathbf{Y} = (Y_1, \ldots, Y_k)$ of compatible arities, let $\mathbf{X} < \mathbf{Y}$ abbreviate the formula $(X_1 \leqslant Y_1 \wedge \cdots \wedge X_k \leqslant Y_k) \wedge (X_1 < Y_1 \vee \cdots \vee X_k < Y_k)$.

Definition 3.9 (SOS Semantics [57] for NTGDs). Let D be a database and $\Sigma(\mathbf{R})$ a set of NTGDs over joint relation symbols $\mathcal{R}$ with list of relation symbols $\mathbf{R} = (R_1, \ldots, R_r)$. Furthermore, let $T(\mathbf{R}) = (D, \Sigma)$ be the logical theory formed by D and Σ. The SOS semantics for D with Σ is defined by a second-order formula $SOS(D, \Sigma)$ as follows:

$$SOS(D, \Sigma) \equiv T(\mathbf{R}) \wedge \neg \exists \mathbf{U} \; \Big(\mathbf{U} < \mathbf{R} \; \wedge \; T^{\mathbf{R}}(\mathbf{U}) \Big) .$$

The set $SMod_{\text{sos}}(D, \Sigma)$ of stable models according to the SOS semantics consists of the models of $SOS(D, \Sigma)$. These models are over arbitrary universes and are not restricted to be Herbrand models. If Q is a UNBCQ, then instead of writing $SOS(D, \Sigma) \models Q$, we may write $(D, \Sigma) \models_{\text{sos}} Q$. ◁

Before discussing the above definition and comparing it to the LP-approach, we adapt some concepts from the latter to the settings of NTGDs. If A is an arbitrary set, and Σ a set of NTGDs, then $ground(\Sigma, A)$ is the conjunction of all possible ground NTGD instances arising from each NTGDs σ of Σ by eliminating the universal quantifiers from σ and uniformly replacing the variables of

σ in all possible ways with domain elements from A. For an interpretation I with domain $dom(I)$, the *GL-reduct* Σ^I results from $ground(\Sigma, dom(I))$ by eliminating each ground instance containing a negative literal $\neg\alpha$ such that $I \models \alpha$ and by eliminating each remaining negative literal from each ground NTGD (which is tantamount to replacing it with *true*). Finally, for a database D and NTGD set Σ, the GL-reduct T^I of the logical theory $T = (D, \Sigma)$ relative to an interpretation I is the ground logical theory $T^I = (D, \Sigma^I)$.

Let us now come back to Definition 3.9. First note that an interpretation I of $T = (\Sigma, D)$ determines an extension $[\mathbf{R}]_I$ of the free predicate variables $\mathbf{R}$ of (Σ, D). As customary in logic, this extension consists of the tuples of values in the relations of $\mathbf{R}$, but has no reference to predicate symbols. By construction, $T^{\mathbf{R}}(\mathbf{U})$ simulates the GL-reduct in the following sense: For each fixed interpretation I, which thus also fixes an extension $[\mathbf{R}]_I$, the set of extensions $\{[\mathbf{U}]_J \mid J \models T^{\mathbf{R}}(\mathbf{U})\}$ is equal to the extensions of the GL-reduct T^I. In other words, for fixed I, the possible extensions of $T^{\mathbf{R}}(\mathbf{U})$ are exactly the extensions of models of T^I. However, the subformula $\neg\exists\mathbf{U}\left(\mathbf{U} < \mathbf{R} \wedge T^{\mathbf{R}}(\mathbf{U})\right)$ cuts off all interpretations I of $T(\mathbf{R})$, whose extensions are not the extensions of $\mathbf{U}$ in a *minimal* model of $T^{\mathbf{R}}(\mathbf{U})$, and thus of a minimal model of T^I. Only those interpretations I of T remain, which are also minimal models of T^I. We keep track of this in the following proposition, which in essence goes back to Reference [57] (Theorem 1 and Corollary 1).

Proposition 3.10. *For a database D and a set of NTGDs Σ, an interpretation I is a model of $SOS(D, \Sigma)$ iff I is a model of T and a minimal model of T^I, where $T = (D, \Sigma)$.*

The above proposition also makes evident that, in the context of NTGDs, the only difference between the SOS approach and the LP approach adopted here is that the former allows for arbitrary interpretations and models, while the latter uses Herbrand interpretations and models only. Regarding entailment, this would not really be a problem if Herbrand interpretations were not subject to the UNA. The following example shows that the adoption of the UNA may make a significant difference:

Example 3.11. Consider a database $D = \{Person(joe)\}$ and a set Σ consisting of the following NTGDs:

$$\sigma_1 : \quad \rightarrow \exists x(Person(x) \wedge Guilty(x)),$$
$$\sigma_2 : Person(x) \wedge \text{not } Guilty(x) \rightarrow Innocent(x).$$

According to the LP semantics adopted in the present article, there is only one stable model in $SMod(D, \Sigma)$, which is $M_1 = \{Person(joe), Person(f_{\sigma_1,x}(0)), Guilty(f_{\sigma_1,x}(0)), Innocent(joe)\}$. Due to the UNA, $f_{\sigma_1,x}(0)$ is assumed to be different from joe. Thus, $(D, \Sigma) \models_{stable} Innocent(joe)$. However, $SOS(D, \Sigma)$ has an infinity of models, but up to isomorphism only two, namely, the above model M_1 and the model $M_2 = \{Person(joe), Guilty(joe)\}$. M_2 arises from the unification of the witness x for σ_1 with the constant joe. We thus have: $(D, \Sigma) \not\models_{\text{sos}} Innocent(joe)$. The two semantics thus significantly differ.

Other examples, where the SOS produces additional models, not isomorphic to those produced by the LP-semantics were given in Reference [57] and elsewhere. One may be tempted to think that, due to the additional possibility of unifying witness values, the SOS semantics always generates at least all the stable models that are produced according to the LP-semantics. However, this is not so. There are cases in which an LP stable model M is actually not an SOS model, because there exists a smaller SOS stable non-Herbrand model $M' < M$, and M is thus not minimal according to the SOS semantics. The following example illustrates such a case:

Example 3.12. Let D be the empty database, and let Σ contain the following NTGDs.

$$true \rightarrow \exists x, \exists y H(x, y), \tag{1}$$

$$true \rightarrow \exists x H(x, x), \tag{2}$$

$$\forall x \forall y \, (H(x, y) \rightarrow E(x, x)), \tag{3}$$

$$\forall x \forall y \, (H(x, y) \rightarrow E(y, y)), \tag{4}$$

$$\forall x \forall y \, (H(x, y) \wedge \neg E(x, y) \rightarrow hello). \tag{5}$$

According to the LP approach, (D, Σ) has exactly one stable model, namely,

$$M = \{H(f_{1,x}(0), f_{1,y}(0)), H(f_{2,x}(0), f_{2,x}(0)), E(f_{1,x}(0), f_{1,x}(0)), E(f_{1,y}(0), f_{1,y}(0)), \\ E(f_{2,x}(0), f_{2,x}(0)), hello\}.$$

This, however, is not a stable model according to the SOS semantics, because the non-Herbrand model M_0 that unifies the witnesses of the TGDs (3) and (4) is smaller. This model is: $M_0 = \{H(f_{2,x}(0), f_{2,x}(0)), E(f_{2,x}(0), f_{2,x}(0))\}$, which is a proper subset of M. It is not hard to see that (up to isomorphism) this is the only SOS model of (D, Σ). We thus have $(D, \Sigma) \models_{stable} hello$, but $(D, \Sigma) \not\models_{\text{sos}} hello$.

In the following, we show that the SOS semantics can express the LP semantics. Towards this aim, we first show how the UNA can be enforced for an NTGD theory $T = (D, \Sigma)$ by transforming T into another NTGD theory T_{UNA}.

Let D be a database and Σ be a set of NTGDs defined over a joint schema $\mathcal{R}$, and let $\mathbf{R} = (R_1, \ldots, R_r)$ be an ordered list of all the relation (= predicate) symbols in $\mathcal{R}$. First, towards a simpler presentation, and without loss of generality, we assume that every existential NTGD in Σ has only one existentially quantified variable. It is well-known and not hard to see that every set of NTGDs can be rewritten into an equivalent set (relative to query-answering over $\mathcal{R}$) in this form by use of auxiliary predicate symbols; see Reference [32].

Let us now split every existential NTGD $\sigma \in \Sigma$ of the form $\forall \mathbf{x} \forall \mathbf{y} \; (body(\mathbf{x}, \mathbf{y}) \rightarrow \exists z \, R(\mathbf{x}, z))$, into the following two NTGDs σ_a and σ_b:

$$\sigma_a : \; \forall \mathbf{x} \forall \mathbf{y} \; (body(\mathbf{x}, \mathbf{y}) \rightarrow \exists z \; Witness_\sigma(\mathbf{x}, \mathbf{y}, z)), \text{ and}$$

$$\sigma_b : \; \forall \mathbf{x} \forall \mathbf{y} \forall z \; (Witness_\sigma(\mathbf{x}, \mathbf{y}, z) \rightarrow R(\mathbf{x}, z)),$$

where $Witness_\sigma$ is a fresh auxiliary predicate symbol whose purpose is to keep track of the existential witness assignments together with the values (instances of $\mathbf{x}$ and $\mathbf{y}$) on which such an assignment depends. Obviously, these splits yield a set of NTGDs equivalent to Σ with respect to answering queries over the schema $\mathcal{R}$.

The UNA states that all database constants and witness values are mutually different. This is easily expressed in first-order logic by use of the $\neq$ predicate, however, this is outside the realm of NTGDs. We will therefore simulate the $=$ and $\neq$ predicates with NTGDs. We introduce a new auxiliary predicate symbol E for equality and add for each relation symbol $R \in \mathcal{R}$ having arity k a rule, for $1 \leqslant i \leqslant k$ a TGD $R(x_1, \ldots, x_i, \ldots x_k) \rightarrow E(x_i, x_i)$. This way, for whatever argument value v that appears in a stable model M of (D, Σ), $E(v, v)$ will be true in M. Moreover, given that there is no further rule with head $E(\ldots)$, due to the minimality of M, E will precisely represent equality in M. Now let bad and bad' be auxiliary propositional atoms and add the NTGD $bad \wedge \neg bad' \rightarrow bad'$. If bad is inferred, then this rule becomes equivalent to $\neg bad' \rightarrow bad'$, which cannot be satisfied by any stable model. The UNA axioms can then be asserted with five groups of NTGDs (of which some are unguarded, but this will not create a problem).

(1) *Different database constants do not unify.* For each pair a and b of distinct values in $dom(D)$, add the TGD $E(a, b) \rightarrow bad$.
(2) *Existential witnesses are different from database constants.* For each existential NTGD $\sigma \in \Sigma$ and each database constant a, add the NTGDs $\forall \mathbf{x} \forall \mathbf{y} \forall z \ (Witness_\sigma(\mathbf{x}, \mathbf{y}, z) \wedge E(z, a) \rightarrow bad)$.
(3) *Witnesses for distinct NTGDs are are mutually distinct.* For each different pair $\sigma, \sigma' \in \Sigma$ of existential NTGDs, add the TGD
$\forall \mathbf{x} \forall \mathbf{y} \forall z \forall z' \ (Witness_\sigma(\mathbf{x}, z) \wedge Witness_{\sigma'}(\mathbf{y}, z') \wedge E(z, z') \rightarrow bad)$.
(4) *Witnesses for different instantiations of (i.e., homomorphisms from) the same NTGD rule body are different.* For each existential NTGD σ and each variable x_i of the body $body(x_1, \ldots x_i, \ldots, x_k)$ of σ, add the NTGD

$$\forall x_1 \ldots x_k \forall y_1 \ldots y_k \forall z \, (Witness_\sigma(x_1, \ldots, x_i, \ldots, x_k, z) \wedge Witness_\sigma(y_1, \ldots, y_i, \ldots, y_k, z) \wedge \neg E(x_i, y_i) \rightarrow bad).$$

(5) *The Witness relation is functional.* Informally, this assures that the witness assignment indeed works akin to Skolem functions that create a *single* witness rather than multiple witnesses for each NTGD application. For each existential NTGD σ, add the following NTGD:

$$\forall \mathbf{x} \forall \mathbf{y} \forall z \forall z' \ (Witness_\sigma(\mathbf{x}, \mathbf{y}, z) \wedge Witness_\sigma(\mathbf{x}, \mathbf{y}, z') \wedge \neg E(z, z') \rightarrow bad).$$

Definition 3.13. For a database D and NTGD set Σ, we denote by Σ^D_{UNA} the set of NTGDs resulting from D and Σ by the above rule splittings and NTGD additions, including the assertion of the above UNA axioms. Moreover, if $T = (D, \Sigma)$, then $T_{UNA} = (D, \Sigma^D_{UNA})$ denotes the logical theory of D and Σ^D_{UNA}, i.e., the set (conjunction) of the logical ground atoms in D and the NTGDs in Σ^D_{UNA}. ◁

Let Γ be a set of interpretations over a schema $\mathcal{S}$, and let $\mathcal{R}$ be a sub-schema of $\mathcal{S}$, then $\Gamma[\mathcal{R}]$ denotes set of projections of interpretations in Γ over the relations of sub-schema $\mathcal{R}$, i.e., the set of extension of the $\mathcal{R}$-predicates of the interpretations in Γ.

For two sets Γ and Γ' of interpretations over a same schema $\mathcal{R}$, we say that Γ *and* Γ' *coincide up to isomorphisms and write* $\Gamma \doteq \Gamma'$ if every interpretation of Γ is isomorphic to at least one interpretation of Γ' and vice versa.

The following theorem finally shows that NTGD sets under the classical LP-semantics can be equivalently reformulated as equivalent NTGD sets under the SOS semantics by adding the UNA axioms (in form of NTGDs). This means that the SOS formalism can simulate Herbrand models and is at least as expressive as the LP-semantics. While we will only need the following result for UNBCQs, we formulate it for arbitrary first-order Boolean queries. That is, we consider the problem $(D, \Sigma) \models_{stable} Q$ for first-order sentences Q, where, by definition, $(D, \Sigma) \models_{stable} Q$ if for each $M \in SMod(D, \Sigma)$, $M \models Q$.

Theorem 3.14. *Let D be a database, Σ a set of NTGDs, and Q a first-order Boolean query over a joint schema $\mathcal{R}$, then*

(1) $SMod(D, \Sigma) \doteq SMod_{\text{sos}}(D, \Sigma^D_{UNA})[\mathcal{R}]$ *and*
(2) $(D, \Sigma) \models_{stable} Q$ *iff* $(D, \Sigma^D_{UNA}) \models_{\text{sos}} Q$ *iff* $SOS(D, \Sigma^D_{UNA}) \models Q$.

The above theorem formally states that the UNA, and only the UNA, makes the difference between the LP-based and the SOS semantics in case of NTGDs, and that the UNA can be added in form of NTGDs to any set Σ of original NTGDs to make both semantics coincide. Informally, we have thus established the following equation: *LP Semantics = SOS Semantics + UNA*, and we may say that the LP-semantics is equivalent to *the SOS semantics under the UNA*. In this sense, the positive decidability and complexity results that we will derive in the sequel are also positive

results for the SOS semantics under the UNA. Recall that the UNA axioms are not all guarded, but this will not be a problem, as we are in the subsequent Section 3.5 mainly interested in the following *semantic* property of the UNA axioms: Whenever added to a set of guarded TGDs, they enforce the so-called tree model property.

3.5 Decidability

It is well known that answering queries relative to sets Σ of TGDs is undecidable, even if both Σ and the query are fixed, the query is atomic, and if only one of the TGDs is existential (see Section 3 of Reference [32] and the discussion and references therein). This raises the question for natural restrictions of NTGDs for which query answering under the stable model semantics is decidable. In this article, we study an important such restriction, namely, the restriction to the above-introduced guarded NTGDs.

Theorem 3.17 below will give a first decidability proof for query answering with guarded NTGDs. This proof rests on a strong logical metatheorem that generalizes Rabin's Theorem on the decidability of monadic second-order logic over trees [104]. We believe that this proof will provide valuable insight into the decidability issue. However, the proof of Theorem 3.17 is not well-suited for directly deriving precise complexity results, let alone practical algorithms. Therefore, in the next two sections, a much more algorithmic proof will be developed, which may guide future implementations and which will allow us to derive precise complexity bounds.

Rabin's Theorem [104] states that S2S, the monadic theory of two successor functions, is decidable. That is, **monadic second-order logic (MSO)** is decidable over the infinite binary tree, and, via interpretability, also over countable trees. Results by Shelah [108] and LeTourneau [85] show that MSO over the class of *all trees* is decidable. This, in turn, was generalized by Courcelle [42, 43], who proved the decidability of MSO over structures of bounded treewidth (as defined in Section 2). A further generalization [72] (see also References [64, 105]) considers the stronger *guarded second-order logic (GSO)*, for which the decidability over structures of bounded treewidth equally holds. GSO, whose expressive power lies properly between the one of MSO and the expressive power of full **second-order logic (SO)**, is the logic which will be used to prove Theorem 3.17. Therefore, we will introduce it here more carefully.[3]

A GSO formula over a schema $\mathcal{R}$ is a formula of classical second-order logic, in which the relational symbols of $\mathcal{R}$ occur as free relational variables, and where each quantified second-order variable is constrained to range over subsets of the extension of some variable R from $\mathcal{R}$. This can be enforced syntactically by restricting second-order existential quantification to the form $\exists X(X \leqslant R \wedge \varphi(X))$, and universal second-order quantification to the form $\forall X((X \leqslant R) \rightarrow \varphi)$, which can be abbreviated by $\forall X \leqslant R\ \varphi$. Observe that first-order quantification does not need to be guarded in GSO. Let us also remark that GSO over graph signatures, i.e., signatures with at most binary relations, is equivalent to the well-known logic MSO_2 by Courcelle [42, 43]. GSO, as presented here, can thus be considered as a straightforward extension of MSO_2 to arbitrary arities.

Definition 3.15. A class C of second-order logic formulas has the *generalized tree model property* if there exists a computable function t, assigning to every formula $\Phi(\mathbf{R}) \in C$ a natural number $t(\Phi)$ such that, whenever Φ is satisfiable, then φ has a model (i.e., an extension of the free predicate variables $\mathbf{R}$ of Φ) of treewidth at most $t(\varphi)$.

[3]Actually, for a simpler presentation, we here introduce a syntactic sub-fragment of full GSO that is sufficient for our purposes. Trivially, all decidability results for full GSO also hold for this sub-fragment.

The following is well-known [42, 72]:

Proposition 3.16 (Decidability Criterion via GSO). *Each decision problem that can be Turing-reduced to the satisfiability problem for GSO formulas of some class C that has the generalized tree model property is decidable.*

The above result was stated in a slightly more general form in Reference [72]. A full proof can be found in a most useful survey paper by Thoralf Räsch [105]. Propositions such as 3.16 are often referred to as *metatheorems*, meaning that they provide tools for generating other theorems, in this case, decidability results. We now use Proposition 3.16 and the second-order formulation of "$(D, \Sigma) \models_{stable} Q$" derived in Section 3.4 to prove our decidability result. As in Theorem 3.14, we consider arbitrary first-order Boolean queries and not merely UNBCQs.

Theorem 3.17. *For databases D, finite sets Σ of guarded NTGDs, and first-order Boolean queries Q, the problem "$(D, \Sigma) \models_{stable} Q$" is decidable.*

Proof. For proving the decidability of $(D, \Sigma) \models_{stable} Q$, we will actually prove the decidability of $(D, \Sigma) \not\models_{\text{stable}} Q$, which is, of course, equivalent. We denote the list of free predicate variables of T by $\mathbf{R}$, and let $\mathbf{R}^+$ denote the list of free predicate variables of $T_{UNA} = (D, \Sigma^D_{UNA})$. Thus, $\mathbf{R}^+$ extends $\mathbf{R}$ by the predicate symbols $Witness_\sigma$ and E.

By Theorem 3.14, $(D, \Sigma) \not\models_{\text{stable}} Q$ is equivalent to $SOS(D, \Sigma^D_{UNA}) \not\models Q$, and thus to the satisfiability of the second-order formula $SOS(D, \Sigma^D_{UNA}) \wedge \neg Q$, which is equivalent to

$$T_{UNA}(\mathbf{R}^+) \wedge \neg \exists \mathbf{U} \left(\mathbf{U} < \mathbf{R}^+ \ \wedge \ T^{\mathbf{R}^+}_{UNA}(\mathbf{U}) \right) \wedge \neg Q.$$

The above is a GSO formula, because by the condition $\mathbf{U} < \mathbf{R}^+$, the extensions of the existentially quantified $\mathbf{U}$ predicate variables are constrained to be subsets of the corresponding extensions of the free $\mathbf{R}^+$ predicate variables.

Let $\mathcal{G}$ be the class of all GSO formulas $T_{UNA}(\mathbf{R}^+) \wedge \neg \exists \mathbf{U} \left(\mathbf{U} < \mathbf{R}^+ \ \wedge \ T^{\mathbf{R}^+}_{UNA}(\mathbf{U}) \right) \wedge \neg Q$ for arbitrary $T = (D, \Sigma)$, where D is a database, and Σ is a guarded set of NTGDs, and arbitrary Boolean first-order query Q, and where, as above, $\mathbf{R}$ and $\mathbf{R}^+$ denote the lists of free predicate symbols occurring in T and T_{UNA}, respectively.

Given that the decision problem $(D, \Sigma) \models_{stable} Q$ for guarded NTGDs is reducible to the satisfiability problem for formulas in class $\mathcal{G}$, by Proposition 3.16, for proving the theorem, it now suffices to establish that $\mathcal{G}$ enjoys the generalized tree model property.

For database D and arbitrary guarded NTGD set Σ, let Σ_* be the set of NTGDs resulting from Σ by the rule splitting with the introduction of the $Witness_\sigma$ atoms and the addition of the rules $R(x_1, \ldots, x_i, \ldots, x_k) \rightarrow E(x_i, x_i)$ as specified in Section 3.4. In other words, Σ is equal to Σ^D_{UNA} minus the NTGDs from the five groups of UNA axioms. Clearly, given that Σ is guarded, so is Σ_*, which results from Σ by the addition of guarded TGDs only. Now observe that via Theorem 3.14, the models of $\Phi = T_{UNA}(\mathbf{R}^+) \wedge \neg \exists \mathbf{U} \, (\mathbf{U} < \mathbf{R}^+ \ \wedge \ T^{\mathbf{R}^+}_{UNA}(\mathbf{U}))$, which are those in $SMod_{\text{sos}}(D, \Sigma^D_{UNA})$, are precisely the stable models of $T_* = (D, \Sigma_*)$. In fact, these theories are over the same schema $\mathbf{R}^+$, and hence no schema projection is necessary. Each such stable model M is a minimal model of T^M_* and can thus be computed as $chase(D, \Sigma^M_*)$. Given that Σ_* is guarded, so is Σ^M_*, and thus, by Proposition 2.6, $tw(M) = tw(chase(D, \Sigma_*)) \leqslant |dom(D)| + w$, where w is the largest arity in Σ_*, and thus, clearly, $tw(M) \leqslant |dom(D)| + size(\Sigma) \leqslant size(\Phi)$. It follows that *all* models in $SMod_{\text{sos}}(D, \Sigma^D_{UNA})$, which are, by definition, the models of $T_{UNA}(\mathbf{R}^+) \wedge \neg \exists \mathbf{U} \, (\mathbf{U} < \mathbf{R}^+ \ \wedge \ T^{\mathbf{R}^+}_{UNA}(\mathbf{U}))$ have bounded treewidth. Conjoining the closed formula $\neg Q$ to the latter formula can only cut models off, and the remaining models (if any) are still models of $SMod_{\text{sos}}(D, \Sigma^D_{UNA})$, and thus all have bounded treewidth. In summary, each satisfiable formula Φ in $\mathcal{G}$, has models whose treewidth are bounded by $size(\Phi)$. (Actually, not just some but *all* models of Φ obey this treewidth bound, which is even more than is required for ensuring the generalized tree model property.) Therefore, $\mathcal{G}$ has the generalized tree model property. □

The decidability result is mainly due to the generalized tree model property of formulas $\Phi = T_{UNA}(\mathbf{R}^+) \wedge \neg\exists \mathbf{U}\,(\mathbf{U} < \mathbf{R}^+ \wedge T^{\mathbf{R}^+}_{UNA}(\mathbf{U}))$, which is entailed by the fact that the models in $SMod(D, \Sigma_*)$ have bounded treewidth. This, in turn, is due to two simultaneous features, which jointly enforce tree-shaped models: (i) the use of the UNA in the SMS semantics and (ii) the guardedness of Σ (and thus also of Σ_*). Each feature alone would not suffice. We briefly discuss the role of these two features.

(i) Role of the UNA. The UNA (implicit in the classical definition of logic programming with Herbrand constants and terms) ensures that the witnesses corresponding to existentially quantified variables are Herbrand terms that are fresh new values different from any database domain value and from any other witness generated. The addition of the UNA thus enforces that the SOS semantics behaves like the classical SMS semantics (Theorem 3.14), which, by Proposition 3.5, corresponds (up to isomorphism of models) to the chase semantics. The chase semantics, in turn, guarantees $[dom(D)]$-acyclicity and the bounded treewidth model property, provided the rules of Σ are guarded. Here, we have thus an interaction between the UNA and guardedness. The following example illustrates this and highlights, in particular, the role of the UNA:

Example 3.18. Consider the database $D = \{R(a, b)\}$, which is obviously acyclic, and let Σ contain the unique guarded rule $R(x, y) \rightarrow \exists z\, R(y, z)$. The UNA of the SMS then enforces a unique SMS-model of (D, Σ), which is the infinite chain $\{R(a, b), R(b, z_1), R(z_1, z_2), R(z_2, z_3), \ldots\}$, where the z_i are the created null values that serve as witnesses. Additional cyclic models such as $\{R(a, b), R(b, a)\}$ or $\{R(a, b), R(b, z_1), R(z_1, a)\}$, and so on, that are all SOS models of (D, Σ), are ruled out by the SMS.

However, when assuming single-headed NTGDs (as done here) and the UNA, *witness creation* by itself cannot introduce new uncovered cycles, and thus cannot spoil model acyclicity, let alone bounded treewidth. Similarly, multi-headed guarded NTGDs would not spoil finite treewidth. However, it is well possible that *unguarded* rules operate on the null values created under the UNA and take them as prime material for weaving interconnections that result in highly cyclic infinite structure of infinite treewith such as infinite grids or even cliques, as will be illustrated under Point (ii) below by Example 3.19. However, even when Σ is restricted to guarded TGDs, dropping the UNA may spoil bounded treewidth. This was shown in Reference [4] and has grim consequences for decidability. In fact, in Reference [4], it is shown that under the SOS semantics, which does not adopt the UNA, sets of NTGDs with appropriate queries are able to enforce models with infinite grids that can be used to express the halting problem for Turing machines. By such a reduction, the query-answering problem for NTGDs under the SOS was shown to be undecidable in Reference [4].

(ii) Role of guardedness. As already explained in (i), the UNA plus guardedness jointly ensure $[D]$-acyclicity and thus also the bounded treewidth property. The following example shows that not only the UNA, but also guardedness is essential here. In fact, the bounded treewidth property is not guaranteed by the UNA alone, as it can be spoiled by even a single unguarded rule. In fact, as the following example shows, unguarded sets of TGDs, even under the UNA, can be used to generate a structure of infinite treewidth that consists of an infinite clique.

Example 3.19. Consider a database $D = \{Vertex(0)\}$ and a set Σ with the three following rules:

$$Vertex(x) \rightarrow \exists y\, Succ(x, y),$$
$$Succ(x, y) \rightarrow Vertex(y),$$
$$Vertex(x) \wedge Vertex(y) \rightarrow Edge(x, y).$$

Clearly, this defines an infinite clique (with self-loops). Such a clique has infinite treewidth.

By a mildly more complicated unguarded set Σ of TGDs, one can define an infinite grid that also has infinite treewidth, and one can then use the ith horizontal lines of the grid to represent the configuration of a Turing machine at time i, starting with the input configuration at line 1. The configuration transitions over time, and thus the computation of the Turing machine, can be simulated by further TGDs. This way, it is possible to encode the halting problem for Turing machines by using TGDs. A detailed account of this (using several guarded and a single unguarded TGD only) is given in Reference [32].

3.6 Complexity Considerations

Unfortunately, Theorem 3.17 does not provide good complexity bounds, let alone acceptable algorithms for deciding $(D, \Sigma) \models_{stable} Q$. Consider, for example, the case of data complexity, when both a guarded set of NTGDs Σ and a conjunctive query Q are fixed and the input consists of a database D only. Even then, the best upper bound that we have on the treewidth of the models in $SMod_{\text{sos}}(D,\Sigma^{D}_{UNA})$ is $dom(D) + \text{ar}(\mathcal{R}) - 1$, which grows linearly in the size of the input D. With such a (non-constant) treewidth, techniques for directly reading off complexity bounds from the quantifier-alternation of SO formulas having the generalized tree model properties yields EXPTIME bounds at best. This motivates the study of refined decision procedures and of their associated complexities, which we carry out in the next sections, where we will show, among other results, that deciding $(D, \Sigma) \models_{stable} Q$ is co-NP-complete in data complexity.

4 STABLE MODEL SEMANTICS FOR GUARDED NTGDS VIA GUARDED DISJUNCTIVE NTGDS WITH STRATIFIED NEGATION

The proof of Theorem 3.17 does neither provide tight bounds for deciding $(D, \Sigma) \models_{stable} Q$ for guarded NTGDs Σ, nor provides appropriate algorithms for this problem. To obtain such bounds and algorithms, we pursue the following approach: In the present section, we polynomially transform the query answering problem for NTGDs under the SMS to a different problem in another formalism that is easier to analyze and lends itself better to an algorithmic approach due to its more procedural nature. Then, in Section 5, we provide complexity bounds and algorithms. More specifically, in the present section, we show how to polynomially transform the decision problem $(D, \Sigma) \models_{stable} Q$, where Σ is guarded, into an equivalent decision problem $(D, \Sigma') \models_{\text{strat}} Q'$, where Σ' is a set of *disjunctive NTGDs (DNTGDs)*, which, moreover, are guarded and stratified, i.e., allow for stratified negation in rule bodies, and where Q' is obtained from Q by a simple modification. The semantics of the entailment relation "$\models_{\text{strat}}$" is based on the *disjunctive chase* procedure, which is a generalization of the chase as defined for plain TGDs in Section 2. The disjunctive chase (just as the chase) can still generate infinite interpretations. However, it will be shown in Section 5 that, thanks to the *guardedness* of the stratified DNTGDs in Σ', only finite parts of these interpretations are relevant for query answering.

We would like to stress that, in the present article, stratified DNTGDs are merely used as an appropriate tool and auxiliary formalism for the derivation of our decision algorithms and complexity results. Given that we focus on guarded NTGDs, we are mainly interested in translating *guarded* NTGD sets into sets of *guarded* stratified DNTGDs, which is done in the proof of Theorem 4.17. However, the translation given in Theorem 4.17 also works for arbitrary (possibly unguarded) sets Σ of DNTGDs, which would then be translated into sets Σ' of unguarded stratified DNTGDs. This could be useful in other contexts that may be of interest elsewhere, for example, for finding further decidable fragments of DNTGDs under the SMS. Moreover, stratified DNTGDs, which are a generalization of DTGDs as studied in References [28, 45, 46], may be considered as a knowledge representation formalism of interest in its own right.

In the following Section 4.1, we define DNTGDs and the disjunctive chase. In Section 4.2, we show our transformation that simulates the stable model semantics for guarded NTGDs by guarded DNTGDs and stratified negation.

4.1 Disjunctive NTGDs with Stratified Negation

We begin by introducing the notion of disjunctive NTGDs, which extend NTGDs by the possibility to use disjunction in their heads. More precisely, a *disjunctive NTGD (DNTGD)* σ *over a schema* $\mathcal{R}$ has the form

$$\forall \mathbf{x} \forall \mathbf{y} \left(\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \bigvee_{i=1}^{n} \exists \mathbf{z}_i \, \Psi_i(\mathbf{x}, \mathbf{z}_i) \right), \tag{6}$$

where $\Phi(\mathbf{x}, \mathbf{y})$ is defined as for NTGDs (i.e., it is a conjunction of literals over $\mathcal{R}$ whose arguments are variables), each $\Psi_i(\mathbf{x}, \mathbf{z}_i)$ is a conjunction of atoms over $\mathcal{R}$ with variables as their arguments, and $n \geqslant 1$. We call $\Phi(\mathbf{x}, \mathbf{y})$ the *body* of σ and $\bigvee_{i=1}^{n} \exists \mathbf{z}_i \, \Psi_i(\mathbf{x}, \mathbf{z}_i)$ the *head* of σ. As usual, we drop the universal quantifiers and abbreviate such a DNTGD as $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \bigvee_{i=1}^{n} \exists \mathbf{z}_i \, \Psi_i(\mathbf{x}, \mathbf{z}_i)$. Throughout this article, we assume without loss of generality that each $\Psi_i(\mathbf{x}, \mathbf{z}_i)$ is a single atom, and that for each pair i and j of distinct indices from $\{1, \ldots, n\}$, we have that $\mathbf{z}_i$ and $\mathbf{z}_j$ are disjoint and do not contain any variables from the body of σ. Notations like $body^+(\sigma)$ and $body^-(\sigma)$ are defined analogously as for NTGDs. By $head(\sigma) := \{\Psi_i(\mathbf{x}, \mathbf{z}_i) \mid 1 \leqslant i \leqslant n\}$, we denote the set of all atoms that occur in the head of σ (recall that we assume that each $\Psi_i(\mathbf{x}, \mathbf{z}_i)$ is a single atom).

Guarded DNTGDs play an important role in the sequel. Here, we say that a DNTGD σ is *guarded* if there is an atom in $body^+(\sigma)$, called a *guard* of σ, that contains all variables that occur in the body of σ. If σ is guarded, then $guard(\sigma)$ denotes an arbitrary guard of σ. Sometimes, we also consider *disjunctive TGDs (DTGDs)*, which are DNTGDs without negative literals.

A DNTGD σ over $\mathcal{R}$ is *satisfied* in an interpretation I for $\mathcal{R}$ if, whenever there is a homomorphism h such that $h(body^+(\sigma)) \subseteq I$ and $h(body^-(\sigma)) \cap I = \emptyset$, then there is an extension $h' \supseteq h_{|\Phi(\mathbf{x},\mathbf{y})}$ and an atom $\alpha \in head(\sigma)$ with $h'(\alpha) \in I$.

Example 4.1. Let σ be $R(x, y, z) \wedge S(x, y) \wedge \neg T(y, z) \rightarrow S(y, z) \vee \exists u \exists v \, U(y, z, u, v)$. Then, σ is a guarded DNTGD, with $R(x, y, z)$ as a guard. In fact, $R(x, y, z)$ is the only guard of σ, and thus $guard(\sigma) = R(x, y, z)$. If I is an interpretation containing the atoms $R(a, b, c)$ and $S(a, b)$, then at least one of the following atoms must be present in I to satisfy σ: $S(b, c)$, $T(b, c)$, or $U(b, c, d, e)$ for some $d, e \in \Delta \cup \Delta_N$.

In this article, we focus on sets of DNTGDs with *stratified negation*. Stratified negation is a well-known form of non-monotonic negation and has recently been studied for sets of NTGDs [33, 88]. It is weaker than negation under the stable model semantics in that it does not permit "negative cycles." To this end, predicates are partitioned into strata in such a way that each atom $R(\mathbf{a})$ only depends on atoms $S(\mathbf{b})$ for which S belongs to a lower stratum than R, or to the same stratum if the dependence on $S(\mathbf{b})$ is positive. Such a partition into strata is called a *stratification*, and a set of DNTGDs that admits a stratification is called *stratified*:

Definition 4.2 (Stratification). Let Σ be a set of DNTGDs over a schema $\mathcal{R}$. A *stratification* of Σ (of length $l \geqslant 1$) is a mapping $s \colon \mathcal{R} \rightarrow \{1, \ldots, l\}$ such that for all $\sigma \in \Sigma$ and all $R(\mathbf{x}) \in head(\sigma)$, we have that

- $s(S) \leqslant s(R)$ for all $S(\mathbf{y}) \in body^+(\sigma)$, and
- $s(S) < s(R)$ for all $S(\mathbf{y}) \in body^-(\sigma)$.

We say that Σ is *stratified* if there is a stratification of Σ.

Given a set Σ of DNTGDs over a schema $\mathcal{R}$, a stratification s of Σ, and an interpretation I for $\mathcal{R}$, we define the *ith stratum* of I as the set of all atoms $R(\mathbf{a})$ in I with $s(R) = i$. ◁

Example 4.3. Let Σ be the set consisting of the following guarded DNTGDs:

$$R(x, y) \wedge \neg A(x) \rightarrow B(y), \tag{7}$$

$$R(x, y) \wedge B(y) \rightarrow \exists z\, S(x, y, z), \tag{8}$$

$$S(x, y, z) \rightarrow C(z) \vee R(y, z), \tag{9}$$

$$S(x, y, z) \wedge \neg C(z) \rightarrow D(x), \tag{10}$$

$$S(x, y, z) \wedge \neg R(y, z) \rightarrow E(x). \tag{11}$$

This set is stratified. A possible stratification of Σ of length 3 is given by the mapping s with $s(A) = 1$, $s(B) = s(C) = s(R) = s(S) = 2$, and $s(D) = s(E) = 3$. It is easy to see that every stratification of Σ has a length of at least 3, since the DNTGDs in Σ enforce that for any such stratification s', we have that $s'(A) < s'(B) \leqslant s'(S) \leqslant s'(C) < s'(D)$.

The semantics of stratified sets of DNTGDs is based on the notion of a *canonical model*. A canonical model is constructed one stratum at a time, starting with the lowest stratum and turning to a higher stratum only after all strictly lower strata have been constructed. This process guarantees that at the time when a DNTGD σ is applied, we have full information about the presence or absence of all negative atoms that are relevant for the application of σ. To give a more precise definition of canonical models, we first generalize the well-known chase procedure [15] (see our Definition 2.2 and Remark 2.3) to sets of DNTGDs and then define canonical models in terms of the generalized chase procedure.

Definition 4.4 (Application of a DNTGD). Let σ be a DNTGD and I be an interpretation.

- We say that σ *is applicable to* I if there exists a homomorphism $h\colon body^+(\sigma) \rightarrow I$ mapping all atoms in $body^+(\sigma)$ to atoms in I, such that h maps no atom in $body^-(\sigma)$ to any atom in I. In this case, we also say that σ *is applicable to* I *with* h.
- Let σ be applicable to I with homomorphism h. Applying σ to I with h yields a new interpretation I' that is obtained from I by adding $h'(\alpha)$, where $\alpha \in head(\sigma)$, and h' is an extension of h that assigns to each variable in α that does not occur in the domain of h a fresh null. We call I' the *result* of applying σ to I (with h and α). ◁

We will apply the chase procedure only to semi-positive sets of DNTGDs. Here, we call a set Σ of DNTGDs *semi-positive* if each predicate R that occurs in a negative literal in the body of some DNTGD in Σ is extensional (i.e., R does not occur in the head of some DNTGD in Σ).

Definition 4.5 (Chase). Let I be an interpretation and Σ be a semi-positive set of DNTGDs.

- A *chase sequence* of I with Σ is a potentially infinite sequence $I_0, I_1, I_2, \ldots$ of interpretations such that $I_0 = I$, each I_{i+1} is a result of applying a DNTGD in Σ to I_i, and for each DNTGD $\sigma \in \Sigma$ that is applicable to some I_i with a homomorphism h,
  —there exists an index $j \geqslant 0$ such that I_{j+1} is the result of applying σ to I_j with h; and
  —for each atom $\alpha \in head(\sigma)$, there exists at most one index $j \geqslant 0$ such that I_{j+1} is the result of applying σ to I_j with h and α.
  We call $I_0 \cup I_1 \cup I_2 \cup \cdots$ the *result* of the chase sequence.
- We denote by $chase(I, \Sigma)$ the set of all interpretations that are the result of some chase sequence of I with Σ. ◁

Remark 4.6. Note that the above definition is a true generalization of the TGD chase of Definition 2.2. In particular, if I is an instance and Σ a set of TGDs, then $chase(I, \Sigma)$ has

precisely the same meaning according to both definitions. Furthermore, recall the requirement from Definition 2.2 (see also Remark 2.3) that each TGD applicable with some homomorphism must be applied exactly one time with this homomorphism. According to the above Definition 4.5, the conditions from Definition 2.2 are now generalized to the following conditions for stratified DNTGDs: For each stratified DNTGD σ that is applicable with a homomorphism h, σ must be applied at least once in the chase sequence (with at least one head atom of σ); moreover, σ with h can be applied at most once per head atom of σ. However, σ with h does not need to be applied with *each* head atom. A disjunctive rule head is thus interpreted as an inclusive disjunction.

Remark 4.7. As for the TGD chase (in analogy to Proposition 2.4), $chase(I, \Sigma)$ may contain many isomorphic models due to different possible choices of names of nulls. If we adopt a fixed deterministic policy of naming nulls, such as, for example, the use of unambiguously identified Herbrand terms, as in Section 3.3, then, for each $I \in chase(I, \Sigma)$, models isomorphic to I due to different name choices for fresh nulls would be cut off. It is not difficult to see that cutting off isomorphic models has no effect whatsoever on query answering. Therefore, whenever convenient, we may assume without loss of generality a fixed deterministic naming policy for nulls.

Example 4.8. The set Σ' consisting of the first three DNTGDs from Example 4.3 (DNTGDs (7)–(9)) is semi-positive. Let $I = \{R(a, b), R(a', b'), A(a')\}$. Then, a possible chase sequence of I with Σ' is $I_0, I_1, \ldots, I_7$, where

$$
\begin{aligned}
I_0 &:= I, & I_4 &:= I_3 \cup \{R(b, u_1)\},\\
I_1 &:= I_0 \cup \{B(b)\}, & I_5 &:= I_4 \cup \{B(u_1)\},\\
I_2 &:= I_1 \cup \{S(a, b, u_1)\}, & I_6 &:= I_5 \cup \{S(b, u_1, u_2)\},\\
I_3 &:= I_2 \cup \{C(u_1)\}, & I_7 &:= I_6 \cup \{C(u_2)\}.
\end{aligned}
$$

Here, u_1 and u_2 are distinct nulls.

Note that by Definition 4.5 (see also Remark 4.6), we are not allowed to apply any DNTGD twice with the same homomorphism and rule head. Consider, for example, the DNTGD (8), which is here applied with the homomorphism $x, y \mapsto a, b$ to interpretation I_1 to yield I_2 with the new atom $S(a, b, u_1)$. Of course, according to Definition 4.4, the DNTGD (8) also remains applicable in all further steps of the chase, i.e., for interpretations $I_2, I_3, \ldots, I_7$. However, by Definition 4.5, given that it was already applied to I_2, we cannot apply it with the same homomorphism again, say, to I_7, where such an application would generate an additional atom like $S(a, b, u_3)$, where u_3 is a fresh null distinct from u_1. However, the repeated application of the DNTGD (9) with the homomorphism $h\colon x, y, z \mapsto a, b, u_1$ does not violate the conditions of the chase, since each application of the DNTGD (9) with h uses a different atom from the head of the DNTGD. A chase sequence can be finite or infinite. The chase sequence in the present example is finite, because all combinations (σ, h) of a DNTGD and a homomorphism that need to be applied have already been applied (and in this particular case, this chase sequence even yields a maximal interpretation as result, because no further option for a rule application yielding a new atom is possible). However, if we were allowed to fire DNTGDs with the same homomorphism multiple times, then, even for this example, we could end up with additional infinite chase sequences, where the DNTGD (9) with the homomorphism $h\colon x, y, z \mapsto a, b, u_1$ is applied over and over, generating an infinite number of atoms of the form $S(a, b, u_i)$ for infinitely many distinct null values u_i.

For a given interpretation I and set of DNTGDs Σ, $chase(I, \Sigma)$ may be finite or infinite and may even consist of uncountably many finite and infinite interpretations. Some may contain others, thus the elements in $chase(I, \Sigma)$ are neither necessarily maximal nor necessarily minimal

interpretations. This is illustrated by Example 4.9, which can be equivalently formulated by NT-GDS under the SMS.

Example 4.9. Let $D = \{R(a,b)\}$ and $\Sigma = \{\sigma : R(x,y) \rightarrow \exists z\,(R(y,z) \vee A(y)\}$. One chase sequence just applies σ a single time with the second head atom, which yields the interpretation $I_1 = \{R(a,b), A(b)\} \in \mathit{chase}(D,\Sigma)$. This is a minimal, but not a maximal one. Another chase sequence applies σ infinitely often, always with the first rule head, which yields $I_2 = \{R(a,b), R(b,u_1), R(u_1,u_2), \ldots\} \in \mathit{chase}(D,\Sigma)$, where the u_i are the generated null values, which is equally a minimal one (we cannot eliminate any atom from it), but which is, again, not maximal. Then, there exist infinitely many chase sequences that generate a certain number of $R(u_i,u_j)$ atoms, and maybe some $A(u_\ell)$ atoms and then stop at some point at a final $A(u_k)$. These are neither minimal nor maximal. In this example, there is also a unique maximal interpretation $I_{\max} \in \mathit{chase}(D,\Sigma)$, which looks as follows: $I_{\max} = \{R(a,b), A(b), R(b,u_1), A(u_1), R(u_1,u_2), A(u_2), \ldots\}$. Here, σ fires infinitely often, and always with each rule head. Moreover, there is a set S of interpretations in $\mathit{chase}(D,\Sigma)$ that differ from $I_{\max}$ only by the fact that some (finite or infinite) subset $V \subseteq W = \{A(a), A(u_1), A(u_2), \ldots\}$ is missing. Given that W is already of cardinality $\aleph_0$, there are $2^{\aleph_0}$ subsets V, and thus S is an uncountable set and so is its superset $\mathit{chase}(D,\Sigma)$.

The variant of the chase introduced in Definition 4.5 is similar to the variant of the chase that was introduced in Reference [46] to deal with an extension of disjunctive TGDs. The main differences are that here we have to deal with negative literals in the bodies of DNTGDs, and that our variant of the chase is *oblivious* in the sense that a DNTGD may be applied (with either a different homomorphism or with another head atom) even if its head is already satisfied. Moreover, we adopt the *inclusive or* as the semantics of disjunction in the heads of DNTGDs. This is because we are allowed to apply a DNTGD several times with the same homomorphism (but with different atoms from the head of that DNTGD). However, it is important to remember that we are not allowed to apply a DNTGD twice with the same homomorphism and head atom. This property is crucial for our constructions in Section 4.2 to work.

We are now ready to define the notion of a canonical model. Given a set Σ of DNTGDs and a stratification s of Σ, we let $\Sigma_s^{(i)}$ be the set of all DNTGDs σ in Σ such that i is the minimum of $s(R)$, where R ranges over all the predicates in the head of σ. For example, if Σ and s are as in Example 4.3, then $\Sigma_s^{(2)}$ is the set Σ' from Example 4.8. It is not difficult to see that if $\sigma \in \Sigma_s^{(i)}$ and $R(\mathbf{x}) \in \mathit{body}^-(\sigma)$, then any DNTGD that contains R in its head must belong to $\Sigma_s^{(j)}$ for some $j < i$. Thus, each $\Sigma_s^{(i)}$ is semi-positive.

Definition 4.10 (Canonical Model). Let I be an interpretation, let Σ be a set of DNTGDs, and let s be a stratification of Σ of length l. We define

- $\mathit{CMod}_s^{(0)}(I,\Sigma) := \{I\}$;
- $\mathit{CMod}_s^{(i)}(I,\Sigma) := \bigcup_{J \in \mathit{CMod}_s^{(i-1)}(I,\Sigma)} \mathit{chase}(J, \Sigma_s^{(i)})$ for $i \in \{1,\ldots,l\}$;
- $\mathit{CMod}_s(I,\Sigma) := \mathit{CMod}_s^{(l)}(I,\Sigma)$.

A *canonical model* for I and Σ relative to s is any interpretation in $\mathit{CMod}_s(I,\Sigma)$. ◁

Example 4.11. Consider the set Σ of guarded DNTGDs and the stratification s of Σ from Example 4.3, as well as the two interpretations I and $J := I_7$ from Example 4.8. Then, $K := J \cup \{E(b)\}$ is a canonical model for I and Σ relative to s, since $I \in \mathit{CMod}_s^{(0)}(I,\Sigma)$, $I \in \mathit{chase}(I, \Sigma_s^{(1)}) \subseteq \mathit{CMod}_s^{(1)}(I,\Sigma)$ (as $\Sigma_s^{(1)} = \emptyset$), $J \in \mathit{chase}(I, \Sigma_s^{(2)}) \subseteq \mathit{CMod}_s^{(2)}(I,\Sigma)$ (as shown in Example 4.8), and $K \in \mathit{chase}(J, \Sigma_s^{(3)}) \subseteq \mathit{CMod}_s^{(3)}(I,\Sigma) = \mathit{CMod}_s(I,\Sigma)$.

It is worth mentioning that the chase at a given stratum S_1 may be infinite, and at the next stratum, the chase would already run on an infinite interpretation, and this could continue for further strata. As we will explain in Section 5, guardedness will ensure that, for answering queries of a given size, it will be sufficient to use only a finite initial fragment of each stratum of the chase.

It is not difficult to verify that a canonical model for I and Σ relative to s is a *model* of I and Σ, that is, it contains I and satisfies all DNTGDs in Σ. Moreover, the following lemma implies that a canonical model can be obtained independently from a concrete stratification:

Lemma 4.12. *Let I be an interpretation for $\mathcal{R}$, let Σ be a set of DNTGDs over $\mathcal{R}$, and let s and t be stratifications of Σ. Then, $CMod_s(I,\Sigma) = CMod_t(I,\Sigma)$.*

Proof. For all stratifications u_1, u_2 of Σ, we write $u_1 \rightsquigarrow u_2$ if there exists an $R^* \in \mathcal{R}$ with $u_2(R^*) = u_1(R^*) - 1$, and $u_2(R) = u_1(R)$ for all $R \in \mathcal{R} \setminus \{R^*\}$. Let $\rightsquigarrow^*$ be the reflexive and transitive closure of $\rightsquigarrow$. It is not difficult to verify that there exists a stratification u of Σ such that for the two stratifications s, t mentioned in the lemma, we have $s \rightsquigarrow^* u$ and $t \rightsquigarrow^* u$. It is therefore sufficient to prove the lemma for the case that $s \rightsquigarrow t$.

In what follows, we assume $s \rightsquigarrow t$ and prove $CMod_s(I,\Sigma) = CMod_t(I,\Sigma)$. To simplify the presentation, we assume that s and t have the same length l; the case that t has one stratum less than s can be dealt with in a similar way. We also focus on the inclusion $CMod_s(I,\Sigma) \subseteq CMod_t(I,\Sigma)$; the converse inclusion is similar.

To prove $CMod_s(I,\Sigma) \subseteq CMod_t(I,\Sigma)$, consider any interpretation $J \in CMod_s(I,\Sigma)$. Let R^* be the predicate in $\mathcal{R}$ with $t(R^*) = s(R^*) - 1$, and set $n := t(R^*)$. By definition, there exist interpretations $J_0, J_1, \ldots, J_l$ such that $J_0 = I$, $J_i \in chase(J_{i-1}, \Sigma_s^{(i)})$ for all $i \in \{1, \ldots, l\}$, and $J_l = J$. Note that for all $i \in \{1, \ldots, n-1, n+2, \ldots, l\}$, we have $\Sigma_s^{(i)} = \Sigma_t^{(i)}$, and consequently $J_i \in chase(J_{i-1}, \Sigma_t^{(i)})$. Hence, if we can show that there is an interpretation $J_n' \in chase(J_{n-1}, \Sigma_t^{(n)})$ with $J_{n+1} \in chase(J_n', \Sigma_t^{(n+1)})$, then we obtain $J = J_l \in CMod_t^{(l)}(I,\Sigma) = CMod_t(I,\Sigma)$.

We now show that such an interpretation J_n' exists. Since $J_n \in chase(J_{n-1}, \Sigma_s^{(n)})$, we know that J_n is the result of a chase sequence $K_0, K_1, K_2, \ldots$ of J_{n-1} with $\Sigma_s^{(n)}$. Similarly, since $J_{n+1} \in chase(J_n, \Sigma_s^{(n+1)})$, we know that J_{n+1} is the result of a chase sequence $L_0, L_1, L_2, \ldots$ of J_n with $\Sigma_s^{(n+1)}$. Let $\mathcal{A}$ be the set of all tuples (i, σ, h, α) consisting of an index $i \geqslant 0$, a DNTGD $\sigma \in \Sigma_s^{(n+1)} \cap \Sigma_t^{(n)}$, a homomorphism h, and an atom $\alpha \in head(\sigma)$ such that the interpretation L_{i+1} is obtained by applying σ to L_i with h and α. If $(i, \sigma, h, \alpha) \in \mathcal{A}$, then for all predicates R that occur in the body of σ we have $t(R) \leqslant n$, with $t(R) < n$ if R occurs in $body^-(\sigma)$. Moreover, no DNTGD in $\Sigma_s^{(n)}$ depends on the atom generated by the application of σ with h and α. Thus, it is possible to extend the chase sequence $K_0, K_1, K_2, \ldots$ in such a way that in addition to the applications of DNTGDs from the original chase sequence, we have chase steps, one for each $(i, \sigma, h, \alpha) \in \mathcal{A}$, where σ is applied with h and α, using exactly the same nulls for existentially quantified variables as in the chase step from L_i to L_{i+1}. Let J_n' be the result of the so-extended chase sequence. Then, we have that $J_n' \in chase(J_n, \Sigma_t^{(n)})$. By dropping from the chase sequence $L_0, L_1, L_2, \ldots$ all chase steps that involve a DNTGD in $\Sigma_s^{(n+1)} \cap \Sigma_t^{(n)}$, we obtain a new chase sequence of J_n' with $\Sigma_t^{(n+1)}$ that results in J_{n+1}. Consequently, $J_{n+1} \in chase(J_n', \Sigma_t^{(n+1)})$, which completes the proof of the lemma. □

In the rest of this article, we often do not explicitly mention the stratification s under consideration, and we speak of a canonical model for I and Σ without referring to s. Similarly, we write $CMod(I,\Sigma)$ instead of $CMod_s(I,\Sigma)$. In particular, $CMod(I,\Sigma)$ is the set of all canonical models for I and Σ.

The answers to any earlier introduced Boolean query relative to an interpretation I and a stratified set Σ of DNTGDs are now defined in terms of the canonical models for I and Σ.

Definition 4.13 (Certain Answers to Boolean Queries). Given an interpretation I and a stratified set Σ of DNTGDs, and a Boolean query Q, we write $(I, \Sigma) \models_{\text{strat}} Q$ to denote the fact that for all $J \in \mathit{CMod}(I, \Sigma)$, we have that $J \models Q$. ◁

4.2 Simulating the Stable Model Semantics for Guarded NTGDs by Guarded DNTGDs and Stratified Negation

We now turn to the main result of this section, which states that the problem of answering a UNBCQ Q with respect to guarded NTGDs under the stable model semantics can be translated in polynomial time into the problem of answering a UNBCQ Q' with respect to stratified sets of guarded DNTGDs. Our translation will work for arbitrary UNBCQs, but we also show that it preserves coveredness, i.e., if Q is covered, then so is the translated query Q'. This will be of relevance to Section 5, where we will mainly study covered queries. The core of this translation is Theorem 4.14 below, which states that for every finite set of guarded NTGDs, one can compute in polynomial time a stratified set of guarded DNTGDs such that the stable models of the former correspond—in a way to be made precise below—to the canonical models of the latter relative to the same database.

Theorem 4.14. *There is a polynomial time algorithm that, given a finite set Σ of guarded NTGDs over a schema $\mathcal{R}$, outputs a stratified set Σ' of guarded DNTGDs over a schema $\mathcal{R}'$ that extends $\mathcal{R}$ for each $R \in R$ by auxiliary predicates comprising, among others a predicate R^+ for each $R \in \mathcal{R}$ and a predicate Fail, with $|\mathcal{R}'| \leqslant 5|\mathcal{R}| + |\Sigma| + 1$ and $\text{ar}(\mathcal{R}') \leqslant 2 \cdot \text{ar}(\mathcal{R})$ such that the following holds for each database D for $\mathcal{R}$:*

(1) For each $I \in \mathit{SMod}(D, \Sigma)$, there exists a $J \in \mathit{CMod}(D, \Sigma')$ such that $I = \{R(\mathbf{a}) \mid R \in \mathcal{R},\ R^+(\mathbf{a}) \in J\}$ and $\mathit{Fail} \notin J$.
(2) For each $J \in \mathit{CMod}(D, \Sigma')$ with $\mathit{Fail} \notin J$, the interpretation $I := \{R(\mathbf{a}) \mid R \in \mathcal{R},\ R^+(\mathbf{a}) \in J\}$ is in $\mathit{SMod}(D, \Sigma)$.

Proof. Let D be a database and Σ a finite set of guarded NTGDs over a schema $\mathcal{R}$. The construction of the desired stratified set Σ' of guarded DNTGDs is based on the following ideas:

Let $\hat{\Sigma}$ be obtained from Σ by eliminating all negative atoms in rule bodies of Σ. Clearly, $\hat{\Sigma}$ is a set of plain TGDs (that is, TGDs that contain neither negative atoms nor disjunctions).

We assume without loss of generality that Herbrand terms as in Section 3.3 are used for nulls, and thus that the names of fresh nulls are unambiguously determined (see Remark 4.7). Then, $\mathit{chase}(D, \hat{\Sigma})$ contains a unique interpretation, which is also the unique minimal Herbrand model of $(D, \hat{\Sigma})$. We call this interpretation $\hat{I}$.

Clearly, every stable model of D with Σ is contained in $\hat{I}$. Thus, $\hat{I}$ is an *overapproximation* for all such stable models.

We include in Σ' guarded TGDs that generate a representation of $\hat{I}$ (see the rule sets (1) and (2) below).

Once we have access to $\hat{I}$, we use disjunctive TGDs to "guess" a subset $I^+ \subseteq \hat{I}$ and its complement $I^- := \hat{I} \setminus I^+$. This step is performed by the rule sets (3) and (4) below. This is actually the only time disjunction is used in Σ'.

The final step is to make sure that I^+ is a stable model of D with Σ, which requires us to verify that I^+ corresponds to the unique minimal model of the Gelfond-Lifschitz reduct of Σ relative to I^+. To this end, Σ' first derives a new set I^* from D by applying the guarded NTGDs in Σ in such a way that negative literals in the bodies of those NTGDs are evaluated in I^+ (see the rule sets (5) and (6) below, where the rules (6) use stratified negation).

Next, Σ' verifies that I^* indeed coincides with I^+. This also requires stratified negation.

Let us now give a more detailed description of the construction. We will use five different sets of predicates to represent the original database D, $\hat{I}$, I^+, I^-, and I^*. More precisely, for each predicate $R \in \mathcal{R}$, we introduce four new predicates $\hat{R}$, R^+, R^-, and R^* of the same arity as R, intended to represent the predicate R in $\hat{I}$, I^+, I^-, and I^*, respectively. The predicate R itself is reserved for the corresponding predicate in D. We will also use special predicates, *Fail* and $Witness_\sigma$ ($\sigma \in \Sigma$), where *Fail* will indicate some inconsistency, and $Witness_\sigma$ will be used to remember the nulls assigned to existentially quantified variables in the head of σ during the construction of $\hat{I}$. Whenever we write an atom $R(\mathbf{x})$ without specifying $\mathbf{x}$ further, we assume that $\mathbf{x}$ is a tuple of $\mathrm{ar}(R)$ distinct variables. Now, to generate the interpretation $\hat{I}$, the set Σ' contains the following guarded TGDs:

(1) $R(\mathbf{x}) \rightarrow \hat{R}(\mathbf{x})$ for each predicate $R \in \mathcal{R}$;
(2) for each $\sigma \in \Sigma$ of the form $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, S(\mathbf{v})$,

$$\bigwedge_{R(\mathbf{u}) \,\in\, body^+(\sigma)} \hat{R}(\mathbf{u}) \;\rightarrow\; \exists \mathbf{z}\, Witness_\sigma(\mathbf{x}, \mathbf{y}, \mathbf{z}) \qquad \text{and} \qquad Witness_\sigma(\mathbf{x}, \mathbf{y}, \mathbf{z}) \;\rightarrow\; \hat{S}(\mathbf{v}).$$

Moreover, to "guess" $I^+ \subseteq \hat{I}$ and its complement $I^- := \hat{I} \setminus I^+$, it contains the following guarded disjunctive TGDs for each predicate $R \in \mathcal{R}$:

(3) $\hat{R}(\mathbf{x}) \rightarrow R^+(\mathbf{x}) \vee R^-(\mathbf{x})$;
(4) $R^+(\mathbf{x}) \wedge R^-(\mathbf{x}) \rightarrow \mathit{Fail}$.

To compute the set I^*, we include the following guarded NTGDs in Σ':

(5) $R(\mathbf{x}) \rightarrow R^*(\mathbf{x})$ for each predicate $R \in \mathcal{R}$;
(6) for each $\sigma \in \Sigma$ of the form $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, S(\mathbf{v})$,

$$\bigwedge_{R(\mathbf{u}) \in body^+(\sigma)} R^*(\mathbf{u}) \;\wedge\; \bigwedge_{R(\mathbf{u}) \in body^-(\sigma)} \neg R^+(\mathbf{u}) \;\wedge\; Witness_\sigma(\mathbf{x}, \mathbf{y}, \mathbf{z}) \;\rightarrow\; S^*(\mathbf{v}).$$

The final ingredient is a set of guarded NTGDs that verifies $I^+ = I^*$. To this end, Σ' contains the following guarded NTGDs for each predicate $R \in \mathcal{R}$:

(7) $R^*(\mathbf{x}) \wedge \neg R^+(\mathbf{x}) \rightarrow \mathit{Fail}$;
(8) $R^+(\mathbf{x}) \wedge \neg R^*(\mathbf{x}) \rightarrow \mathit{Fail}$.

This finishes the description of the construction of Σ'.

It is straightforward to verify that Σ' is a stratified set of guarded DNTGDs. Indeed, one possible stratification of Σ' is the mapping that assigns the integer 1 to all predicates of the form R, $\hat{R}$, R^+, R^-, and $Witness_\sigma$, the integer 2 to all predicates of the form R^*, and the integer 3 to *Fail*. It is also straightforward to verify that the schema $\mathcal{R}'$ of Σ' consists of $5|\mathcal{R}| + |\Sigma| + 1$ predicates of maximum arity $\mathrm{ar}(\mathcal{R}') \leqslant 2 \cdot \mathrm{ar}(\mathcal{R})$ (note that the positions in $Witness_\sigma$ correspond to the positions for variables that occur in the guard of σ and the positions for variables that occur in the head of σ). Clearly, Σ' can be constructed in time polynomial in the size of Σ.

It remains to prove that Σ' satisfies properties 1 and 2 of the theorem. For property 1, let $I \in SMod(D, \Sigma)$, and let $\hat{I}$ be the unique minimal model of D with $\hat{\Sigma}$. Then, $I \subseteq \hat{I}$. Let W be the set of all tuples $(\sigma, \mathbf{a}, \mathbf{b}, \mathbf{c})$ with the following properties: (i) σ is a guarded NTGD in Σ of the form $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \Psi(\mathbf{x}, \mathbf{z})$; and (ii) there exists a homomorphism h from $body^+(\sigma)$ to $\hat{I}$ such that $\mathbf{a} = h(\mathbf{x})$, $\mathbf{b} = h(\mathbf{y})$, and $\mathbf{c} = \mathbf{f}_\sigma(\mathbf{a}, \mathbf{b})$. Then,

$$\begin{aligned} J := D \cup \{ Witness_\sigma(\mathbf{a}, \mathbf{b}, \mathbf{c}) \mid (\sigma, \mathbf{a}, \mathbf{b}, \mathbf{c}) \in W \} \cup \{ \hat{R}(\mathbf{a}) \mid R(\mathbf{a}) \in \hat{I} \} \\ \cup \{ R^+(\mathbf{a}) \mid R(\mathbf{a}) \in I \} \cup \{ R^-(\mathbf{a}) \mid R(\mathbf{a}) \in \hat{I} \setminus I \} \cup \{ R^*(\mathbf{a}) \mid R(\mathbf{a}) \in I \} \end{aligned}$$

is a canonical model of D and Σ' (assuming that all terms involving function symbols are interpreted as distinguished nulls), and it furthermore satisfies $I = \{R(\mathbf{a}) \mid R \in \mathcal{R}, R^+(\mathbf{a}) \in J\}$ and $\mathit{Fail} \notin J$.

To prove property 2, consider any model $J \in \mathit{CMod}(D, \Sigma')$ with $\mathit{Fail} \notin J$. Define $\hat{I} := \{R(\mathbf{a}) \mid R \in \mathcal{R}, \hat{R}(\mathbf{a}) \in J\}$, $I := \{R(\mathbf{a}) \mid R \in \mathcal{R}, R^+(\mathbf{a}) \in J\}$, $I^- := \{R(\mathbf{a}) \mid R \in \mathcal{R}, R^-(\mathbf{a}) \in J\}$, and $I^* := \{R(\mathbf{a}) \mid R \in \mathcal{R}, R^*(\mathbf{a}) \in J\}$. Given that $\hat{I}$ is the unique minimal model of D with $\hat{\Sigma}$, by the construction of Σ', both I and I^* are subsets of $\hat{I}$. Moreover, since $\mathit{Fail} \notin J$, and J satisfies the DNTGDs in points 7 and 8, we have $I = I^*$. Thus, the NTGDs in points 5 and 6 ensure that I is the unique minimal model of the Gelfond-Lifschitz reduct of (D, Σ) relative to I. It follows that I is a stable model of D and Σ. □

Example 4.15. We illustrate the construction described in the proof of Theorem 4.14 by applying it to the database $D = \{R(a_0, a_1), P(a_0)\}$ and the following set Σ of guarded NTGDs:

$$\begin{aligned}
\sigma_1 &: \; R(x,y) \rightarrow \exists z\, R(y,z),\\
\sigma_2 &: \; R(x,y) \wedge \neg P(x) \rightarrow P(y),\\
\sigma_3 &: \; R(x,y) \wedge \neg Q(x) \rightarrow Q(y).
\end{aligned}$$

The following set I is the unique stable model of D and Σ, where $a_{i+2} := f(a_i, a_{i-1})$:

$$I = \{R(a_i, a_{i+1}),\, P(a_{2i}),\, Q(a_{2i+1}) \mid i \geqslant 0\}.$$

The stratified set Σ' of guarded DNTGDs as constructed in the proof of Theorem 4.14 contains the following DNTGDS. To derive atoms over the schema $\{\hat{R}, \hat{P}, \hat{Q}\}$, it contains the TGDs $R(x,y) \rightarrow \hat{R}(x,y)$, $P(x) \rightarrow \hat{P}(x)$, $Q(x) \rightarrow \hat{Q}(x)$, and the following additional TGDs:

$$\begin{aligned}
\hat{R}(x,y) &\rightarrow \exists z\, \mathit{Witness}_{\sigma_1}(x,y,z), & \mathit{Witness}_{\sigma_1}(x,y,z) &\rightarrow \hat{R}(y,z),\\
\hat{R}(x,y) &\rightarrow \mathit{Witness}_{\sigma_2}(x,y), & \mathit{Witness}_{\sigma_2}(x,y) &\rightarrow \hat{P}(y),\\
\hat{R}(x,y) &\rightarrow \mathit{Witness}_{\sigma_3}(x,y), & \mathit{Witness}_{\sigma_3}(x,y) &\rightarrow \hat{Q}(y).
\end{aligned}$$

To derive the atoms over the schema $\{R^+, R^-, P^+, P^-, Q^+, Q^-\}$, it contains the guarded DTGDs:

$$\begin{aligned}
\hat{R}(x,y) &\rightarrow R^+(x,y) \vee R^-(x,y), & R^+(x,y) \wedge R^-(x,y) &\rightarrow \mathit{Fail},\\
\hat{P}(x) &\rightarrow P^+(x) \vee P^-(x), & P^+(x) \wedge P^-(x) &\rightarrow \mathit{Fail},\\
\hat{Q}(x) &\rightarrow Q^+(x) \vee Q^-(x), & Q^+(x) \wedge Q^-(x) &\rightarrow \mathit{Fail}.
\end{aligned}$$

And to derive the atoms over the schema $\{R^*, P^*, Q^*\}$, it contains the guarded TGDs $R(x,y) \rightarrow R^*(x,y)$, $P(x) \rightarrow P^*(x)$, and $Q(x) \rightarrow Q^*(x)$, together with:

$$\begin{aligned}
R^*(x,y) \wedge \mathit{Witness}_{\sigma_1}(x,y,z) &\rightarrow R^*(y,z),\\
R^*(x,y) \wedge \neg P^+(x) \wedge \mathit{Witness}_{\sigma_2}(x,y) &\rightarrow P^*(y),\\
R^*(x,y) \wedge \neg Q^+(x) \wedge \mathit{Witness}_{\sigma_3}(x,y) &\rightarrow Q^*(y).
\end{aligned}$$

Finally, it contains the following guarded DNTGDs:

$$\begin{aligned}
R^*(x,y) \wedge \neg R^+(x,y) &\rightarrow \mathit{Fail}, & R^+(x,y) \wedge \neg R^*(x,y) &\rightarrow \mathit{Fail},\\
P^*(x) \wedge \neg P^+(x) &\rightarrow \mathit{Fail}, & P^+(x) \wedge \neg P^*(x) &\rightarrow \mathit{Fail},\\
Q^*(x) \wedge \neg Q^+(x) &\rightarrow \mathit{Fail}, & Q^+(x) \wedge \neg Q^*(x) &\rightarrow \mathit{Fail}.
\end{aligned}$$

It is possible to verify that up to renaming of nulls, the following is the unique canonical model of D and Σ', where we interpret $a_2, a_3, \ldots$ as nulls:

$$\begin{aligned} J = D \cup \{ &Witness_{\sigma_1}(a_i, a_{i+1}, a_{i+2}), Witness_{\sigma_2}(a_i, a_{i+1}), Witness_{\sigma_3}(a_i, a_{i+1}) \mid i \geqslant 0\} \\ \cup \{ &\hat{R}(a_i, a_{i+1}), \hat{P}(a_i), \hat{Q}(a_{i+1}) \mid i \geqslant 0\} \\ \cup \{ &R^+(a_i, a_{i+1}), P^+(a_{2i}), Q^+(a_{2i+1}) \mid i \geqslant 0\} \cup \{P^-(a_{2i+1}), Q^-(a_{2i}) \mid i \geqslant 0\} \\ \cup \{ &R^*(a_i, a_{i+1}), P^*(a_{2i}), Q^*(a_{2i+1}) \mid i \geqslant 0\}. \end{aligned}$$

Clearly, the extensions of P^+, Q^+, and R^+ in this model coincides with the extensions of P, Q, and R in I, respectively.

Example 4.16. Recall that the definition of the chase (Definition 4.4) does not allow to fire a DNTGD twice with the same homomorphism and head atom. This requirement is essential. The following example shows that the proof of Theorem 4.14 would fail, if DNTGDs could be applied twice or more for a given homomorphism and head atom. Let Σ contain the following guarded NTGDs:

$$\begin{aligned} &A(x) \rightarrow \exists y\, R(x,y), \qquad R(x,y) \wedge \neg B(y) \rightarrow C(y), \qquad R(x,y) \wedge C(y) \rightarrow C(x), \\ &\qquad\qquad\qquad\quad R(x,y) \wedge \neg C(y) \rightarrow B(y), \qquad R(x,y) \wedge B(y) \rightarrow B(x). \end{aligned}$$

There are exactly two stable models of $D = \{A(a)\}$ and Σ:

$$\begin{aligned} I_1 &= \{R(a, f(a)), A(a), B(a), B(f(a))\}, \\ I_2 &= \{R(a, f(a)), A(a), C(a), C(f(a))\}. \end{aligned}$$

Let Σ' be the stratified set of DNTGDs obtained from Σ using the translation in the proof of Theorem 4.14. If DNTGDs could be applied more than once for each homomorphism and head atom, then we could construct a canonical model J of D and Σ' that represents both I_1 and I_2. Specifically, J would contain $\{R^+(a, X), A^+(a), B^+(a), B^+(X)\}$ and $\{R^+(a, Y), A^+(a), C^+(a), C^+(Y)\}$. This not only shows that we would not have a one-to-one correspondence between the stable models of D and Σ and the canonical models of D and Σ', but it also implies negative consequences for query answering. Indeed, the UNBCQ $\exists x\,(A(x) \wedge \neg B(x)) \vee \exists x\,(A(x) \wedge \neg C(x))$ is true in every stable model of D and Σ, but if we replace the predicates A, B, and C in Q by the corresponding predicates A^+, B^+, C^+ in the schema of Σ', then the resulting query is false in J.

We are now ready to show that the problem of answering UNBCQs with respect to guarded NTGDs under the stable model semantics can be translated in polynomial time into the problem of answering (covered) UNBCQs with respect to stratified sets of guarded DNTGDs.

Theorem 4.17. *There is a polynomial time algorithm that, given a finite set Σ of guarded NTGDs and a UNBCQ Q, outputs a finite stratified set Σ' of guarded DNTGDs and a UNBCQ Q' such that $(D, \Sigma) \models_{stable} Q$ iff $(D, \Sigma') \models_{\text{strat}} Q'$.*

If $\mathcal{R}$ and $\mathcal{R}'$ are the schemas of Σ and Σ', respectively, then $|\mathcal{R}'| \leqslant 5|\mathcal{R}| + |\Sigma| + 1$ and $\text{ar}(\mathcal{R}') \leqslant 2 \cdot \text{ar}(\mathcal{R})$. Moreover, we have that $\text{wd}(Q) = \text{wd}(Q')$, that Q' is covered if Q is covered, and that Q' is a UBCQ if Q is a UBCQ.

Proof. Let Σ and Q be given, and let $\mathcal{R}$ be the schema of Σ and Q. By Theorem 4.14, we can compute in polynomial time a stratified set Σ' of guarded DNTGDs over a schema $\mathcal{R}'$ with $|\mathcal{R}'| \leqslant 5|\mathcal{R}| + |\Sigma| + 1$ and $\text{ar}(\mathcal{R}') \leqslant 2 \cdot \text{ar}(\mathcal{R})$ such that the following holds for each database D for $\mathcal{R}$:

(P1) For each $I \in SMod(D, \Sigma)$, there exists a $J \in CMod(D, \Sigma')$ such that $I = \{R(\mathbf{a}) \mid R \in \mathcal{R},\ R^+(\mathbf{a}) \in J\}$ and $Fail \notin J$.

(P2) For each $J \in \mathit{CMod}(D, \Sigma')$ with $\mathit{Fail} \notin J$, the interpretation $I := \{R(\mathbf{a}) \mid R \in \mathcal{R},\ R^+(\mathbf{a}) \in J\}$ is in $\mathit{SMod}(D, \Sigma)$.

Set $Q' := \mathit{Fail} \vee Q^+$, where Q^+ is obtained from Q by replacing each predicate $R \in \mathcal{R}$ with R^+. Then, Q' is a UNBCQ with $\mathrm{wd}(Q') = \mathrm{wd}(Q')$. Moreover, Q' is covered if Q is covered, and Q' is a UBCQ if Q is a UBCQ. It remains to show that $(D, \Sigma) \models_{\mathit{stable}} Q$ iff $(D, \Sigma') \models_{\mathsf{strat}} Q'$.

For the "only if" direction, assume $(D, \Sigma') \not\models_{\mathsf{strat}} Q'$. Then, there exists a $J \in \mathit{CMod}(D, \Sigma')$ with $J \not\models Q'$, where $J \not\models Q'$ implies that $\mathit{Fail} \notin J$ and $J \not\models Q^+$. Now, by property (P2), $I := \{R(\mathbf{a}) \mid R \in \mathcal{R},\ R^+(\mathbf{a}) \in J\}$ is a stable model of D and Σ. Since $J \not\models Q^+$, we have $I \not\models Q$, so I is also a witness of $(D, \Sigma) \not\models_{\mathsf{stable}} Q$.

For the converse, let $(D, \Sigma) \not\models_{\mathsf{stable}} Q$. Then, there exists a stable model I of D and Σ with $I \not\models Q$. Property (P1) implies that there exists a $J \in \mathit{CMod}(D, \Sigma')$ such that $I = \{R(\mathbf{a}) \mid R \in \mathcal{R},\ R^+(\mathbf{a}) \in J\}$ and $\mathit{Fail} \notin J$. Thus, $J \not\models Q'$, which proves $(D, \Sigma') \not\models_{\mathsf{strat}} Q'$. □

5 COMPLEXITY OF QUERY ANSWERING

In this section, we derive complexity bounds and algorithms for answering UNBCQs for NTGDs under the SMS, that is, deciding $(D, \Sigma) \models_{\mathit{stable}} Q$; and query-answering for stratified sets of DNTGDs, that is, deciding $(D, \Sigma) \models_{\mathsf{strat}} Q$.

Our main result for answering covered UNBCQs with respect to guarded NTGDs under the stable model semantics is stated as follows:

Theorem 5.1. *Given as input a database D for a schema $\mathcal{R}$, a finite set Σ of guarded NTGDs over $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma) \models_{\mathit{stable}} Q$ has the following complexity:*

(1) 2-EXPTIME*-complete in combined complexity, where* 2-EXPTIME*-hardness holds even in case Σ is fixed,* $\mathrm{ar}(\mathcal{R})$ *is bounded by a constant, and Q is a BCQ.*
(2) EXPTIME*-complete, if* $\mathrm{ar}(\mathcal{R})$ *is bounded by a constant, and Q is acyclic, where* EXPTIME*-hardness holds even in case Q is atomic.*
(3) co-NP*-complete in data complexity (i.e., for fixed $\mathcal{R}$, Σ, and Q), or if $|\mathcal{R}|$,* $\mathrm{ar}(\mathcal{R})$*, and* $\mathrm{wd}(Q)$ *are bounded by a constant.*

The lower bounds for this theorem are proven in Section 5.1, where we actually derive slightly stronger results by exhibiting further restrictions on $\mathcal{R}$, Σ, and Q, for which hardness still holds (Theorem 5.5). Similar lower bounds hold for deciding $(D, \Sigma) \models_{\mathsf{strat}} Q$ when Σ is a set of stratified DNTGDs. These lower bounds will be stated in Theorem 5.4.

Section 5.2 deals with the upper bounds. These are obtained by first deriving upper bounds for the decision problem $(D, \Sigma) \models_{\mathsf{strat}} Q$ with DNTGDs, which are summarized in Theorem 5.6, where actually completeness results are stated. From this, by using our polynomial-time transformation of stable-model query-answering into query-answering based on stratified NTGDs proven in Theorem 4.17, we easily derive the upper bounds for stable-model reasoning stated in the above Theorem 5.1.

Regarding the upper bounds for the problem $(D, \Sigma) \models_{\mathsf{strat}} Q$, in Section 5.2, we will first sketch an algorithm that is based on the chase for stratified DNTGDs (Definition 4.5) and computes the disjunctive chase models partially, up to a sufficiently deep level. This algorithm yields a matching upper bound of co-NP for data complexity, but is not optimal for combined complexity and for the case of bounded arities. To obtain matching upper bounds, and thus completeness results, we show how this algorithm can be transformed into a suitable alternating algorithm for deciding $(D, \Sigma) \models_{\mathsf{strat}} Q$. This is first done with a proof sketch for UBCQs; a full proof is deferred to Appendix B. The result for UBCQs is then generalized to covered UNBCQs.

5.1 Lower Bounds

Towards establishing lower bounds (hardness results) for the decision problem $(D, \Sigma) \models_{stable} Q$ for NTGD sets Σ , let us first state a definition and recall a useful result from Reference [28].

Definition 5.2 (Based on [28]). A set Σ of DTGDs is a set of *disjunctive inclusion dependencies in normal form (NF-DIDs)* if each $\sigma \in \Sigma$ is of one of the following three types:

(1) $R(\mathbf{x}) \rightarrow S(\mathbf{x}')$, where $\mathbf{x}' \subseteq \mathbf{x}$,
(2) $R(\mathbf{x}) \rightarrow \exists y\, S(\mathbf{x}', y)$, where $\mathbf{x}' \subseteq \mathbf{x}$,
(3) $R(\mathbf{x}) \rightarrow R_1(\mathbf{x}') \vee R_2(\mathbf{x}'')$, where $\mathbf{x}' \subseteq \mathbf{x}$ and $\mathbf{x}'' \subseteq \mathbf{x}$,

where, moreover, for all three types of rules, in each body or head atom α, each variable from $var(\alpha)$ occurs only once in α.

The following was shown in Reference [28]:

Proposition 5.3 ([28]). *Query answering under the classical logical entailment relation "$\models$" and with fixed sets of NF-DIDs is* 2-EXPTIME*-hard, in particular:*

(1) There is a fixed set Σ_1 of NF-DIDs over a schema $\mathcal{R}$ with $\mathrm{ar}(\mathcal{R}) = 2$ such that, when given a database D and a UBCQ Q as input, deciding $(D, \Sigma_1) \models Q$ is 2-EXPTIME*-hard. (See Reference [28], proof of Theorem 4.5 and its subsequent discussion, and proof of Theorem 4.6.)*
(2) There is a fixed set Σ_2 of NF-DIDs over a schema $\mathcal{R}$ with $\mathrm{ar}(\mathcal{R}) = 3$ such that, when given a database D and a CQ Q as input, deciding $(D, \Sigma_2) \models Q$ is 2-EXPTIME*-hard. (See Reference [28], Theorem 4.8.)*

For rules with purely positive rule bodies, such as NF-DIDs, there is only one stratum, and thus $(D, \Sigma) \models_{\mathrm{strat}} Q$ is equivalent to $chase(D, \Sigma) \models Q$, where $chase(D, \Sigma)$ is defined as in Definition 4.5. For DTGDs and UBCQs, as in our case, the chase is *universal*, which means that $chase(D, \Sigma) \models Q$ is equivalent to $(D, \Sigma) \models Q$ [28, 45].[4] Hence, the "$\models_{strat}$" and the "$\models$" relation are interchangeable in the context of Proposition 5.3. Therefore, by putting together known hardness results for deciding $chase(D, \Sigma) \models Q$ stated in Theorem 6.1 of Reference [32] (where Σ contains no negation and is thus trivially stratified) with the above special cases of fixed NF-DID sets, we get the following lower bounds (i.e., hardness results) for the problem of deciding $(D, \Sigma) \models_{\mathrm{strat}} Q$:

Theorem 5.4. *Given as input a database D for a schema $\mathcal{R}$, a finite stratified set Σ of guarded DNTGDs over $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma) \models_{\mathrm{strat}} Q$ is:*

(1) 2-EXPTIME*-hard, where hardness holds already (a) for fixed ground atomic queries, or when Σ is a fixed set of DTGDs whose bodies all consist of single positive atoms and either (b) $\mathrm{ar}(\mathcal{R}) = 2$, and Q is a UBCQ, or (c) $\mathrm{ar}(\mathcal{R}) = 3$, and Q is a BCQ.*
(2) EXPTIME*-hard, if $\mathrm{ar}(\mathcal{R})$ is bounded by a constant, and Q is acyclic. Hardness in this case holds even for ground atomic queries.*
(3) co-NP*-hard in data complexity, or, if $|\mathcal{R}|$, $\mathrm{ar}(\mathcal{R})$, and $\mathrm{wd}(Q)$ are bounded by a constant. Hardness in this case holds even for ground atomic queries.*

We now provide similar lower bounds for the problem of deciding $(D, \Sigma) \models_{stable} Q$, thus establishing the lower bounds for Theorem 5.1. Actually, regarding the arity $\mathrm{ar}(\mathcal{R})$, we here provide slightly more detailed bounds than those stated in Point 1 of Theorem 5.1.

[4] In References [45] and [28], the chase is defined in a slightly different way than in Definition 4.5, but with respect to UBCQ answering, ours and their definitions are equivalent.

Theorem 5.5. *Given as input a database D for a schema $\mathcal{R}$, a finite set Σ of guarded NTGDs over $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma) \models_{stable} Q$ is:*

(1) 2-EXPTIME*-hard, even if (a) Q is a single atom, and $\mathrm{ar}(\mathcal{R})$ is unbounded, or (b) Σ is fixed, $\mathrm{ar}(\mathcal{R}) = 2$, and Q is an UBCQ, or (c) Σ is fixed, $\mathrm{ar}(\mathcal{R}) = 3$, and Q is a BCQ.*
(2) EXPTIME*-hard, even if $\mathrm{ar}(\mathcal{R})$ is bounded by a constant, and Q is atomic.*
(3) co-NP*-hard in data complexity, or if $|\mathcal{R}|$, $\mathrm{ar}(\mathcal{R})$, and $\mathrm{wd}(Q)$ are bounded by a constant.*

Proof. Point 1 (a) follows from Theorem 6.1 (2) of Reference [32], which proves 2-EXPTIME-hardness of checking $chase(D, \Sigma) \models Q$ for sets Σ of TGDs (without negation) having unbounded arities and for atomic queries Q. By Proposition 3.7 in the present article, we know that this problem is equivalent to $(D, \Sigma) \models_{stable} Q$. Let us now show that Points 1(b) and 1(c) follow from Points 1 and 2 of Proposition 5.3, respectively. For this, it is sufficient to exhibit an arity-preserving polynomial-time transformation τ, which transforms each NF-DID σ into a set $\tau(\sigma)$ of guarded NTGDs, such that $(D, \Sigma) \models_{\mathrm{strat}} Q$ iff $(D, \tau(\Sigma)) \models_{stable} Q$, where $\tau(\Sigma)$ is defined to be $\bigcup_{\sigma \in \Sigma} \tau(\sigma)$. Our transformation τ is defined as follows: For each NF-DID σ that conforms to type (1) or type (2) in Definition 5.2, let $\tau(\sigma) = \{\sigma\}$. Moreover, for each σ conforming to type (3), i.e., when σ is of the form $R(\mathbf{x}) \rightarrow R_1(\mathbf{x}') \vee R_2(\mathbf{x}'')$, where $\mathbf{x}' \subseteq \mathbf{x}$ and $\mathbf{x}'' \subseteq \mathbf{x}$, let $\tau(\sigma)$ be the set consisting of the following five NTGDs, where $\bar{R}_1$ and $\bar{R}_2$ are fresh predicate symbols of the same arity as R_1 and R_2, respectively:

$$\sigma_1\colon\ R(\mathbf{x}) \wedge \neg R_1(\mathbf{x}') \rightarrow \bar{R}_1(\mathbf{x}'), \qquad \sigma_2\colon\ R(\mathbf{x}) \wedge \neg \bar{R}_1(\mathbf{x}') \rightarrow R_1(\mathbf{x}'),$$
$$\sigma_3\colon\ R(\mathbf{x}) \wedge \neg R_2(\mathbf{x}'') \rightarrow \bar{R}_2(\mathbf{x}''), \qquad \sigma_4\colon\ R(\mathbf{x}) \wedge \neg \bar{R}_2(\mathbf{x}'') \rightarrow R_2(\mathbf{x}''),$$

$$\sigma_5\colon\ R(\mathbf{x}) \wedge \bar{R}_1(\mathbf{x}') \wedge \bar{R}_2(\mathbf{x}'') \rightarrow \mathit{Fail}.$$

Here, as customary, *Fail* is a special atom that, by definition, cannot be true in any stable model.[5] For each Herbrand instantiation $h(\sigma)$ of σ, these five rules enforce a choice of a (due to σ_5) nonempty subset of $\{h(R_1(\mathbf{x}')), h(R_2(\mathbf{x}''))\}$ to be made for each stable model, where $h(R(\mathbf{x}))$ is true, mimicking exactly an application of $h(\sigma)$, when the body $h(R(\mathbf{x}))$ is true, and thus σ fires. From this, it can be seen that—up to isomorphic renaming of Skolem terms—the stable Herbrand models of $(D, \tau(\Sigma))$ projected to the schema of Σ coincide with the disjunctive chase models of (D, Σ) (we omit a more formal proof). Thus, τ is an arity-preserving transformation that transforms Σ into a set of guarded NTGDs such that $(D, \Sigma) \models_{\mathrm{strat}} Q$ iff $(D, \tau(\Sigma)) \models_{stable} Q$. This concludes the proof of Points 1(b) and 1(c).

Point 2 follows from the fact that answering atomic queries relative to guarded TGDs is EXPTIME-hard in the bounded arity case [32].

Point 3 follows from John Schlipf's result that 3SAT can be transformed to the problem of whether a guarded normal Datalog program[6] $\mathbf{P}$ applied to a database D admits a stable model (see Example 4.1 in Reference [107]). Add to the schema of $\mathbf{P}$ a new propositional (i.e., zero-ary) predicate q. Then, $(D, \mathbf{P}) \models_{stable} q$ iff $(D, \mathbf{P})$ has no stable model, hence checking $(D, \mathbf{P}) \models_{stable} q$ is co-NP-hard. □

[5] We can assume without loss of generality the existence of such a special atom with this semantics. In fact, to enforce *Fail* to have its intended semantics, one may simply consider *Fail* to be a regular propositional atom and add the rule $\mathit{Fail} \wedge \neg q \rightarrow q$, where q is a new propositional letter.

[6] The program $\mathbf{P} = \mathbf{P_1} \cup \mathbf{P_2}$ of Schlipf's example is actually not safe, because the rules of $\mathbf{P_1}$ are not safe. However, $\mathbf{P}$ can be made safe without essentially changing its semantics as follows: Assert into D a fact $\mathit{Propletter}(p)$ for each propositional letter p, and add $\mathit{Propletter}(X)$ to the bodies of the two rules of $\mathbf{P_1}$.

5.2 Upper Bounds

First, in Section 5.2.1, we state in Theorem 5.6 our main complexity results for query answering with stratified DNTGDs and explain that they are sufficient for achieving the upper bounds in Theorem 5.1, which is our main complexity result. In Section 5.2.2, we then discuss the role of guardedness towards an algorithmic approach and describe a rudimentary decision algorithm and initial upper bounds for combined complexity and a matching upper complexity bound for data complexity (the latter even holding for arbitrary, not necessarily covered UNBCQs). More precise upper bounds for the case of combined complexity for UBCQs will be stated in Section 5.2.3, where we sketch how our rudimentary algorithm can be translated into a more sophisticated alternating algorithm. This is then extended to covered UNBCQs in Section 5.2.4.

5.2.1 Main Technical Complexity Theorem. Our rationale is to prove upper bounds for query answering with stratified DNTGDs, that is, prove upper bounds to Theorem 5.4 and use these upper bounds to easily derive the upper bounds for our main complexity result, Theorem 5.1. The main technical result in the rest of Section 5 are the upper bounds of the following theorem:

Theorem 5.6 (Theorem 5.4 Completed with Upper Bounds). *Given as input a database D for a schema $\mathcal{R}$, a finite stratified set Σ of guarded DNTGDs over $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma) \models_{\text{strat}} Q$ has the following complexity:*

(1) 2-EXPTIME*-complete, where hardness holds already (a) for fixed ground atomic queries, or when Σ is a fixed set of DTGDs whose bodies all consist of single positive atoms and either (b) $\text{ar}(\mathcal{R}) = 2$, and Q is a UBCQ, or (c) $\text{ar}(\mathcal{R}) = 3$, and Q is a BCQ.*

(2) EXPTIME*-complete, if $\text{ar}(\mathcal{R})$ is bounded by a constant, and Q is acyclic. Hardness in this case holds even for ground atomic queries.*

(3) co-NP*-complete in data complexity, or, if $|\mathcal{R}|$, $\text{ar}(\mathcal{R})$, and $\text{wd}(Q)$ are bounded by a constant. Hardness in this case holds even for ground atomic queries.*

The lower bounds (hardness results) of the above theorem were already given in Theorem 5.4 of Section 5.1. The above Theorem 5.6 thus simply turns these lower bounds into completeness results. What remains to do is proving the upper bounds (membership results), which will be done in the sequel of the present Section 5. Before proceeding with this, let us state more formally that, in fact, this immediately also provides upper bounds for query answering with NTGDs under the SMS.

Lemma 5.7. *The upper bounds in Theorem 5.6 imply the upper bounds in Theorem 5.1.*

Proof. Assume the upper bounds in Theorem 5.6 are proven. For the upper bounds of Theorem 5.1, we first use Theorem 4.17 to translate the given set Σ of guarded NTGDs and the covered UNBCQ Q in polynomial time into a stratified set Σ' of guarded DNTGDs and a covered UNBCQ Q' such that $(D, \Sigma) \models_{\textit{stable}} Q$ iff $(D, \Sigma') \models_{\text{strat}} Q'$. This reduction preserves the width of the query and uses $O(|\mathcal{R}| + |\Sigma|)$ predicates of arity $O(\text{ar}(\mathcal{R}))$. Note that $|\Sigma|$ is bounded by a constant if both $|\mathcal{R}|$ and $\text{ar}(\mathcal{R})$ are bounded by a constant. The upper bounds of Theorem 5.1 thus follow via this reduction. □

For completeness, we note that, trivially, the upper bounds of Theorem 5.1 carry over to Theorem 5.5, which gives more detailed lower bounds, and therefore all hardness results in Theorem 5.5 turn into completeness results.

5.2.2 Role of Guardedness, Rudimentary Algorithm, and Upper Bounds. In Section 3.5, we already discussed the role of guardedness for decidability: Together with the UNA, the guardedness

of a set of NTGDs enforces the bounded treewidth model property, which, in turn, enabled us to apply a powerful meta-theorem on guarded second-order logic to show the decidability of the problem $(D, \Sigma) \models_{stable} Q$. In the present Section 5.2.2, as well as in the subsequent Section 5.2.3, we shall heavily exploit guardedness for deriving concrete algorithms and complexity results for deciding $(D, \Sigma) \models_{\mathsf{strat}} Q$ for DNTGD sets Σ and covered NBCQs. We next illustrate in a somewhat informal fashion how guardedness concretely matters. We will discuss two important properties of guardedness and then use them for actually obtaining a first concrete decision algorithm for deciding $(D, \Sigma) \models_{\mathsf{strat}} Q$, and thus, by Lemma 5.7, also for $(D, \Sigma) \models_{stable} Q$. We start by introducing a relevant concept:

$\mathcal{R}$-type. If I is an interpretation over a schema $\mathcal{R}$, and if $\alpha \in I$ is an atom in I, then *the $\mathcal{R}$-type for α in I* is the set of all atoms $\alpha' \in I$ such that $dom(\alpha') \subseteq dom(\alpha)$.

There are mainly two relevant properties of guardedness that we will exploit algorithmically:

Property 1: Chase models are $[dom(D)]$-acyclic (Proposition 2.6). More precisely, each model I in $CMod(D, \Sigma)$ has a $[D]$-join forest. Essentially, this means that the canonical models of $CMod(D, \Sigma)$ can be generated in form of (possibly infinite) forests. Based on this idea, we use so-called *dependency forests*, where each atom is expanded via the chase by all possible rules with all possible homomorphisms, giving rise to child atoms. Such trees can be finite or infinite. An example of a finite dependency forest is given in Appendix B.[7] Each time a multiple head rule is applicable, a specific choice of a head atom has to be made, and different choices usually lead to different models. If all canonical models are finite, then we could iterate over all such choices and compute successively all canonical models. That we can also deal with infinite models is due to *Property 2*, which we explain next.

Property 2: Atoms of chase forests that are isomorphic and have isomorphic $\mathcal{R}$ types have isomorphic sets of possible subtrees in the forest. For this reason, a finite initial fragment (of computable depth) of each canonical model will suffice for query answering. Let us limit our attention for the moment to atomic queries Q. Clearly, the problem $(D, \Sigma) \models_{\mathsf{strat}} Q$ is decidable iff its complementary problem $(D, \Sigma) \not\models_{\mathsf{strat}} Q$ is decidable. We concentrate on the latter, because it is conceptually somewhat simpler: Rather than having to construct *all* chase models in $CMod(D, \Sigma)$, it is sufficient to nondeterministically generate *one* model $I \in CMod(D, \Sigma)$, such that $I \not\models Q$. Each such I is $[D]$-acyclic and can be organized in form of a $[D]$-join tree that corresponds to a dependency forest (without duplicates). Let α be any atom occurring in I, and let $T_I(\alpha)$ denote the subtree of I rooted at α. Since guardedness guarantees the connectedness condition (see Section 2.2), any element of $dom(I)$ that occurs both inside and outside of $T_I(\alpha)$ must also occur in α. Thus, the only outside atoms that could possibly be involved in the generation of $T_I(\alpha)$ are those with arguments in $dom(\alpha)$, which are thus those from the $\mathcal{R}$-type for α in I. Therefore, any subtree $T_I(\alpha)$ depends only on α and on the $\mathcal{R}$-type of α, and, of course, on the nondeterministic choices made to satisfy disjunctive rule heads inside the computation of $T_I(\alpha)$. (This also means that the sets of all possible subtrees that may be rooted at two isomorphic atoms having isomorphic $\mathcal{R}$ types are isomorphic.)

If we refer to the $\mathcal{R}$-type of α in I by $type_I(\alpha)$, then $T_I(\alpha)$ depends, as said, only on the pair $(\alpha, type_I(\alpha))$ and on the nondeterministic choices made while computing $T_I(\alpha)$. Now let us assume that in I, on a branch that already contains the atom α, we encounter somewhere below α another atom α' isomorphic to α, where also $type_I(\alpha')$ is isomorphic to $type_I(\alpha)$, which we denote by $(\alpha, type_I(\alpha)) \approx (\alpha', type_I(\alpha'))$. We would then not need to further expand α'. In fact, due to the isomorphism, we could then always expand α' by creating a subtree rooted at α', making exactly

[7] Technically, such dependency forests are not $[dom(D)]$-acyclic because of repeated atoms with nulls. However, by either ignoring or suppressing such duplicate atoms, one obtains a proper $[D]$-join forest.

the same nondeterministic choices as were already made for the subtree rooted at α. By doing so, the subtree rooted at α' would be isomorphic to the subtree below α, and thus no new atoms would be generated that could make our atomic query Q true. It follows that we need to expand a branch only as long as no isomorphic pairs $(\alpha, type_I(\alpha)) \approx (\alpha', type_I(\alpha'))$ occur on it.

By a combinatorial argument, it can be shown that there is a natural number d^*, depending only on $\mathcal{R}$, such that whenever a branch length exceeds d^*, then the branch must contain at least one such isomorphic pair. Therefore, no branch needs to be continued beyond depth d^*, and it is thus always sufficient to generate a finite initial part of depth not exceeding d^* of the forest generated by the chase.

The combinatorial argument (which is essentially the same as for non-disjunctive TGDs given in the proof of Lemma 2 in Reference [33]) works as follows: Let $a = \text{ar}(\mathcal{R})$. Each branch B of I is rooted in some atom α_0. Due to guardedness, the only database constants in the entire branch are the at most a constants from $dom(\alpha_0)$. An atom α on B can contain at most a such constants as arguments and no more than a nulls. There are no more than $(2a)^a$ ways of distributing a selection from a constants and a nulls over the argument positions of α. There are, moreover, at most $|\mathcal{R}|$ predicates. So, the largest set of mutually non-isomorphic atoms can have no more than $|\mathcal{R}| \cdot (2a)^a$ elements. For each such atom α, $type(\alpha)$ is a subset of the set $\Omega(\alpha)$ of all atoms with arguments from $dom(\alpha)$. $\Omega(\alpha)$ is of cardinality at most $|\mathcal{R}| \times a^a$, and thus there are at most $2^{|\mathcal{R}| \cdot a^a}$ different types for α. The largest set of mutually non-isomorphic pairs $(\alpha, type(\alpha))$ is therefore of cardinality at most $d^* := |\mathcal{R}| \cdot (2a)^a \times 2^{|\mathcal{R}| \cdot a^a}$. This means that every time a branch exceeds depth d^*, it must already contain isomorphic pairs, and we can thus cut it off at depth d^*.

For answering a non-atomic BCQ, NBCQ, or UNBCQ Q, the same line of reasoning applies, except that we must add the factor $\text{wd}(Q)$, which is the number of atoms in Q, and then define for the non-atomic case d^* to be $d^* := \text{wd}(Q) \cdot |\mathcal{R}| \cdot (2a)^a \times 2^{|\mathcal{R}| \cdot a^a}$. As an illustration of why the factor $\text{wd}(Q)$ is needed, consider, for example, the chain query $Q = \exists x_1 \cdots \exists x_{r+1}\, R(x_1, x_2) \wedge R(x_2, x_3) \wedge \ldots \wedge R(x_r, x_{r+1})$. For Q, it may be the case that the first atom $R(x_1, x_2)$ only matches some atom at depth $d = |\mathcal{R}| \cdot (2a)^a \times 2^{|\mathcal{R}| \cdot a^a} d^*$ in I; from there, we may thus have to continue the branch for another d elements before matching $R(x_2, x_3)$ and so on for all remaining atoms of Q. In case of multiple strata, we have to develop each stratum up to the above depth d^*. (An analogous result for NTGDs is Theorem 32 of Reference [33].) It is not hard to see that this also works for queries Q with negated atoms.

In summary, it is sufficient to compute a finite $[D]$-acyclic structure I_{fin} of which each branch of each stratum has at most depth d^*, such that $(D, \Sigma) \models_{\text{strat}} Q$ iff $I_{fin} \models Q$. Also observe that d^* only depends on $a = \text{ar}(\mathcal{R})$, $|\mathcal{R}|$, and $\text{wd}(Q)$. Note that I_{fin} is, in general, not a model from $CMod(D, \Sigma)$, but is equivalent to such a model for query answering for queries up to $\text{wd}(Q)$ atoms. This concludes our description of Property 2 and its use.

Obviously, for general unguarded DNTGDs, and even for unguarded TGDs, the above does not hold. Neither do their chase models have an acyclic tree-like structure (see Example 3.19), nor do they necessarily produce repeating isomorphic substructures. In fact, as mentioned in Example 3.19, with unguarded TGDs, we can construct infinite grids on which arbitrary Turing machine computations can be simulated [32], which do not need to contain any periodic repetitions of isomorphic substructures whatsoever. From Properties 1 and 2, in contrast, we derive the following basic algorithm:

Rudimentary nondeterministic algorithm. To decide $(D, \Sigma) \models_{\text{strat}} Q$ for a database D, a set Σ of stratified guarded DNTGDs, and a UNBCQ Q, proceed as follows:

(1) Nondeterministically compute, as above, an initial part I_{fin} of a model in $CMod(D, \Sigma)$ by exhausting rule application, stratum by stratum (according to Definition 4.10), making

nondeterministic choices for multi-atom rule heads and stopping at derivation depth[8] d^* of each branch at each stratum.

(2) If $I_{fin} \models Q$, then return *no*, else return *yes*.

This algorithm, whose correctness follows from the above considerations, gives us first bounds for query answering:

Theorem 5.8. *The following upper bounds apply to both the problem* $(D, \Sigma) \models_{\text{strat}} Q$, *where* Σ *is a set of stratified DNTGDs, and the problem* $(D, \Sigma) \models_{stable} Q$, *where* Σ *is a set of NTGDs, and where in both cases* Q *is a UNBCQ:*

(a) co-3-NEXPTIME *in combined complexity.*
(b) co-2-NEXPTIME *when the arities of the relation symbols in* $\mathcal{R}$ *are bounded.*
(c) co-NP *in data complexity (i.e., when* Σ *and* Q *are fixed), or if* $|\mathcal{R}|$, $\text{ar}(\mathcal{R})$, *and* $\text{wd}(Q)$ *are bounded by a constant.*

Proof. Observe that $d^* = \text{wd}(Q) \cdot |\mathcal{R}| \cdot (2a)^a \times 2^{|\mathcal{R}| \cdot a^a}$ is (a) double-exponential in the arity $\text{ar}(\mathcal{R})$, exponential in *size*$(\mathcal{R})$, and linear in $\text{wd}(Q)$ in general; (b) single-exponential in *size*$(\mathcal{R})$ and linear in $\text{wd}(Q)$ in case $\text{ar}(\mathcal{R})$ is bounded; and (c) constant if $|\mathcal{R}|$, $\text{ar}(\mathcal{R})$, and $\text{wd}(Q)$ are bounded by a constant (and thus also when data complexity is considered). For the size of the structure I_{fin}, this then means the following: The size of I_{fin} is in case (a) triple-exponential in the input (D, Σ, Q) (as I_{fin} now may contain trees of double-exponential depth), in case (b) double-exponential in (D, Σ, Q), and in case (c) polynomial in the input database D.

From the above, and from the observation that checking a polynomially sized query against structures of exponential size such as I_{fin} in cases (a) and (b) above does not create an additional level of exponentiality, by the above algorithm, we immediately get the bounds 3-NEXPTIME, 2-NEXPTIME, and NP for combined complexity, bounded arity, and data complexity, respectively. For the complementary problem $(D, \Sigma) \models_{\text{strat}} Q$, we thus obtain the bounds stated under (a), (b), and (c). By the polynomial problem reduction of Theorem 4.17, the same bounds hold for $(D, \Sigma) \models_{stable} Q$ for sets of NTGDs Σ. □

We have shown in Section 5.1 that the bound (c) for data complexity in Theorem 5.8 is a matching upper bound, but we currently do not know whether the bounds for the combined complexity (a) and the case of bounded arities (b) are optimal for general UNBCQs. We are, however, able to derive precise bounds for all cases for the slightly less expressive class of queries of *covered* UNBCQs. These bounds, which are the main complexity results of the present article, are significantly lower than the above bounds in cases (a) and (b) and will be dealt with in the following Section 5.2.3 for UBCQs, in Section 5.2.4 for covered UNBCQs, and in more detail in Appendix B.

5.2.3 Matching Upper Bounds for UBCQs. We sketch a proof sketch of how, in case of UBCQs, the rudimentary algorithm can be transformed into an alternating algorithm with matching upper bounds. A detailed full proof is given in Appendix A. We will thus show:

Theorem 5.9 (Upper Bounds of Theorem 5.6 Restricted to UBCQs). *Given as input a database* D *for a schema* $\mathcal{R}$, *a finite stratified set* Σ *of guarded DNTGDs over* $\mathcal{R}$, *and a covered UNBCQ* Q *over* $\mathcal{R}$, *deciding* $(D, \Sigma) \models_{\text{strat}} Q$ *is:*

(1) in 2-EXPTIME *in general;*
(2) in EXPTIME, *if* $\text{ar}(\mathcal{R})$ *is bounded by a constant, and* Q *is acyclic;*
(3) in co-NP *in data complexity, or, if* $|\mathcal{R}|$, $\text{ar}(\mathcal{R})$, *and* $\text{wd}(Q)$ *are bounded by a constant.*

[8]The derivation depth of an atom in a stratum is the number of rules that are used to generate it from a database that consists of D plus all atoms in lower strata.

Proof. (Informal sketch) $[D]$-join forests such as those generated by guarded DNTGDs (which correspond to dependency forests after duplicate elimination), have various computational advantages. One is, for example, the following: If α is an atom of a canonical model $I \in CMod(D, \Sigma)$, and $\beta_1, \ldots, \beta_r$ are the children of α in I, which are each generated by a single rule application, then, by the connectedness condition (see Section 2.2), the only values that a subtree rooted at β_i for $1 \leqslant i \leqslant r$ may share with the rest of I (and thus also with the other subtrees) are those from $dom(\alpha)$. Moreover, by the definition of the $\mathcal{R}$-type, the only atoms that such subtrees may share with the rest of I are those from the $\mathcal{R}$-type for α in I. In particular, this means that the side atoms (i.e., the non-guard atoms) that may be used in rule applications in any subtree rooted at α, and that are not generated in the subtree itself, must be elements of the $\mathcal{R}$-type for α in I. This $\mathcal{R}$-type is not only finite, but also of rather limited size. In fact, the values that may appear as arguments in the $\mathcal{R}$-type for α in I are only those few that also appear in the single atom α. This means informally that, after passing a limited finite amount of initial information to the subtrees rooted at $\beta_1, \ldots, \beta_r$, these subtrees can be developed completely independently from each other and from the rest of I. This, in turn, will enable us to use space-bounded alternating Turing machines towards the derivation of optimal complexity bounds.

The above description, however, leaves a puzzling question open: How can we know the $\mathcal{R}$-type for α in I *before* having computed the subtrees rooted at the children $\beta_1, \ldots, \beta_r$ of α? Indeed, some atoms of this $\mathcal{R}$-type may be computed *within* these subtrees, and others even somewhere else in the dependency forest. The answer is that our alternating algorithm will simply *guess* the $\mathcal{R}$-type T for alpha, which, of course, needs to be consistent with the $\mathcal{R}$-type of the parent node of α in case $\alpha \notin D$. Our algorithm will also guess, for each newly guessed atom of T, a subtree where this atom ought to be derived and will then impose to each subtree the additional task to check that those atoms from T assigned to it are effectively derived.

Actually, the information to be passed to each subtree rooted at α will contain not only the $\mathcal{R}$-type of α, but also other data structures that will be used for tackling additional difficulties such as, for example, (i) proving negative literals, i.e., checking that the corresponding (positive) atom cannot be derived; (ii) establishing a (guessed) order of proof of the atoms in each guessed $\mathcal{R}$-type τ to be checked, such that the computation of each subtree has knowledge about which atoms from τ it can use, because they can be deemed to have been already proved; (iii) avoiding that a rule is fired twice with the same homomorphism and head atom (recall Definition 4.5, the discussion in Section 4, and Example 4.16); (iv) managing a set of partial matches of a non-atomic query Q (as described below). All this information is packed in a data structure that we call a Σ-*environment*, where we deal with ground atomic queries only, and (Σ, Q)-*environment*, where we deal with UBCQs. These data structures are described in detail in Appendix A.

By the above-described properties of $[D]$-join forests, and by using the above data structures, we are able to design an alternating algorithm that performs recursive self-reductions for query answering.

Essentially, for each atom α generated, after having guessed (existential step) the $\mathcal{R}$-type of α, the algorithm launches (as a universal step) the independent computations of all the subtrees, where some subtrees must fulfill additional "assignments," such as checking that Q does not become true and proving that certain atoms of the guessed $\mathcal{R}$-type of α are effectively derivable. This alternation between such existential and universal steps is the core of our alternating algorithm for query answering.

In this alternating algorithm, the tree structure I_{fin} from before will be nondeterministically generated and explored by guessing and checking $\mathcal{R}$-types and other auxiliary data structures, and by independently and dynamically developing subtrees without ever materializing I_{fin} in its entirety.

Checking that $I_{fin} \not\models Q$ in the course of such a computation is easy when Q is a ground atomic query: Each subtree computation (including those starting at each database atom) receives the assignment to fail in case atom Q is encountered.

The case of UBCQs Q is somewhat more involved. In essence, the algorithm simultaneously proves for each disjunct Q' of Q that $I_{fin} \not\models Q'$. Roughly, this works by successively matching single query atoms of Q' with atoms of the chase forest, generating (in all possible ways) so-called *partial matches*. The latter are partial mappings from the query variables to database or null values. As the computation processes downwards the forest, refinements of these partial matches are systematically added whenever further atoms can be matched. After each new refinement via an atom matching, a partial match decomposes into (a conjunction of) one or more variable-disjoint blocks of partial matchings for subqueries, where different blocks may be processed and further expanded by different subtrees. To show that $I_{fin} \not\models Q'$, the algorithm, for each such decomposition, nondeterministically chooses one block that shall be doomed to fail and continues to further refine only this block. A block fails if whenever a leaf of I_{fin} is reached, there has been no way to extend the partial matchings in the block to a full matching. All this can be done in form of additional tasks superimposed to the above-described alternating algorithm.

We close our proof sketch with complexity considerations. In the general case (combined complexity), the above refined alternating algorithm uses "only" exponential workspace and thus runs in alternating exponential space AEXPSPACE, which is equal to 2-EXPTIME. This is clearly significantly better than the above co-3-NEXPTIME bound of the rudimentary algorithm. In case of bounded arity and atomic (or even acyclic) queries, the alternating algorithm runs in APSPACE, which is equal to EXPTIME. Finally, in the data complexity case, the alternating algorithm runs in co-NP, which is the same as for the rudimentary algorithm. □

The above alternating algorithm works well for UBCQs, but unfortunately not for general UNBCQs. For example, when processing a query $R(x, y) \wedge R(z, t) \wedge \neg R(x, t)$ (we omit the existential quantifiers), by our algorithm, it could become necessary to decompose the positive part of the query into the blocks $R(x, y)$ and $R(z, t)$ and process each block in a different subtree. However, there would then be no way of enforcing $\neg R(x, t)$, because these subtrees are dealt with independently. However, as shown in the subsequent Section 5.2.4, it is possible to extend our methods to *covered* UNBCQs, because these can be transformed into standard UBCQs with the help of additional guarded NTGDs.

5.2.4 Covered UNBCQs. We finally prove the complexity bounds in Theorem 5.6 for the problem of answering covered UNBCQs with respect to stratified sets of guarded DNTGDs. Note that covered UNBCQs are still much more general than the standard BCQs or UBCQs dealt with in the overwhelming part of the literature on query complexity. The lower bounds were already shown in Theorem 5.4. The upper bounds are obtained from Theorem 5.9 by a polynomial-time reduction to answering UBCQs with respect to stratified sets of guarded DNTGDs:

Lemma 5.10. *There is a polynomial-time algorithm that, given as input a stratified set Σ of guarded DNTGDs over $\mathcal{R}$, a database D for $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, outputs a stratified set Σ' of guarded DNTGDs over $\mathcal{R}' \supseteq \mathcal{R}$ and a UBCQ Q' over $\mathcal{R}'$ such that*

(1) $(D, \Sigma) \models_{\mathsf{strat}} Q$ *iff* $(D, \Sigma') \models_{\mathsf{strat}} Q'$,
(2) $|\mathcal{R}'| \leqslant 2|\mathcal{R}|$,
(3) $\mathrm{ar}(\mathcal{R}') = \mathrm{ar}(\mathcal{R})$, *and*
(4) $\mathrm{wd}(Q') = \mathrm{wd}(Q)$.

Furthermore, Q' is acyclic if Q is acyclic.

Proof. Given Σ, D, and Q, we apply the following transformation to reduce Σ and Q to a stratified set Σ' of guarded DNTGDs and a UBCQ Q'. For each negative literal $\ell = \neg R(\mathbf{x})$ in Q, we create a new guarded NTGD

$$\sigma_\ell : \ \alpha \wedge \ell \rightarrow \bar{R}(\mathbf{x}),$$

where α is a positive literal in Q that covers ℓ, and $\bar{R}$ is a fresh predicate of arity $\mathrm{ar}(R)$. (We introduce at most one new predicate $\bar{R}$ for each $R \in \mathcal{R}$.) Then,

$$\Sigma' := \Sigma \cup \{\sigma_\ell \mid \ell \text{ is a negative literal in } Q\},$$

and Q' is the UBCQ obtained from Q by replacing each negative literal $\neg R(\mathbf{x})$ by the positive literal $\bar{R}(\mathbf{x})$. It is clear that $(D, \Sigma) \models_{\mathsf{strat}} Q$ iff $(D, \Sigma') \models_{\mathsf{strat}} Q'$, and that Σ' and Q' can be computed in polynomial time. Furthermore, letting $\mathcal{R}'$ be the schema of Σ' and Q', we have $|R'| \leqslant 2|R|$ and $\mathrm{ar}(\mathcal{R}') = \mathrm{ar}(\mathcal{R})$. The width of the query is preserved.

Since Q is covered, for each negative literal $\neg\beta$ in Q, there exists a positive literal α in Q with $dom(\beta) \subseteq dom(\alpha)$. This implies that Q' is acyclic if Q is acyclic. □

This completes the Proof of Theorem 5.6 and, by Lemma 5.7, also the Proof of Theorem 5.1.

6 ADDING NORMAL (NEGATIVE) CONSTRAINTS, EQUALITY-GENERATING DEPENDENCIES (EGDS), AND KEYS

In this section, we concentrate on query answering from databases relative to NTGDs and additionally normal (negative) constraints, equality-generating dependencies (EGDs), and keys.

6.1 Normal (Negative) Constraints

Negative constraints (see Section 2.5) are an important ingredient of ontology languages. Here, we consider their generalization to *normal negative constraints* (or simply *normal constraints*), which also allow for negated atoms in their bodies and which are formally first-order formulas σ of the form $\forall \mathbf{x}\, \Phi(\mathbf{x}) \rightarrow \bot$, where $\Phi(\mathbf{x})$ is a (not necessarily guarded) conjunction of literals (without nulls) called its *body*. We denote by $body^+(\sigma)$ (respectively, $body^-(\sigma)$) the set of all atoms that occur positively (respectively, negated) in the body of σ. We say that σ is *covered* if, for every negative literal $\neg\alpha$ in $\Phi(\mathbf{x})$, there exists a positive literal β in $\Phi(\mathbf{x})$ such that every variable and every constant in α occurs in β. We usually omit the universal quantifiers and implicitly assume that all sets of normal constraints are finite here.

Example 6.1. If the two unary predicates c and c' represent two classes (also called concepts in DLs), then we may use the constraint $c(x), c'(x) \rightarrow \bot$ to assert that the two classes have no common instances. Similarly, if additionally the binary predicate r represents a relationship (also called a role in DLs), then we may use the constraint $c(x), r(x, y) \rightarrow \bot$ to enforce that no member of c participates to r, while the two covered normal constraints $\neg c(x), r(x, y) \rightarrow \bot$ and $\neg c'(y), r(x, y) \rightarrow \bot$ enforce that r relates only members from c and c'. Moreover, if the two binary predicates r and r' represent two relationships, then the constraint $r(x, y), r'(x, y) \rightarrow \bot$ expresses that the two relationships are disjoint.

As for the semantics, a normal constraint σ is *satisfied* in an interpretation I for $\mathcal{R}$ if every homomorphism h satisfies either $h(body^+(\sigma)) \not\subseteq I$ or $h(body^-(\sigma)) \cap I \neq \emptyset$. The following definition extends the notions of stable models and of answers to UNCQ and UNBCQs under the stable model semantics to additional normal constraints.

Definition 6.2. Given sets Σ_T and Σ_C of NTGDs and normal constraints, respectively, a *stable model* for D and $\Sigma = \Sigma_T \cup \Sigma_C$ is a stable model for D and Σ_T that satisfies all $\sigma \in \Sigma_C$. The set of all

stable models for D and Σ is denoted by $SMod(D, \Sigma)$. The set of all *answers* for a UNCQ Q over D and Σ under the stable model semantics is the set $ans_{stable}(Q, D, \Sigma)$ of all tuples $\mathbf{t}$ such that $\mathbf{t} \in Q(I)$ for all $I \in SMod(D, \Sigma)$. If Q is a UNBCQ, then we write $(D, \Sigma) \models_{stable} Q$ to denote the fact that $ans_{stable}(Q, D, \Sigma) \neq \emptyset$, that is, Q is satisfied in all stable models for D and Σ. ◁

If a constraint is guarded (and thus *all* the variables from the rule body must occur in the guard, in particular, also those in negative literals), then it is clearly also covered. Guarded constraints are extremely simple to enforce by a well-known old trick: Just replace each guarded constraint $\sigma : body(\mathbf{x}) \rightarrow \bot$ by the rule $\sigma' : body(\mathbf{x}) \wedge \neg p \rightarrow p$, where p is a fresh propositional atom. Clearly, σ' is guarded and destroys any stable model that would satisfy $body(\mathbf{x})$. It has thus the same effect as σ. The trick, however, does not work for the more general class of covered constraints. In fact, then σ' could be an unguarded rule. However, we show below that σ can be transformed to an additional query disjunct.

It is, in fact, not difficult to see that answering UNBCQs Q on a database D relative to a set Σ_T of NTGDs and a set Σ_C of normal constraints under the stable model semantics can be reduced to answering Q on D relative to only Σ_T under the stable model semantics, namely, by evaluating Q only on all stable models that satisfy every normal constraint $\sigma \in \Sigma_C$. This is equivalent to evaluating either Q or $Q_\sigma = \Phi(\mathbf{x})$ for some $\sigma = \Phi(\mathbf{x}) \rightarrow \bot \in \Sigma_C$ to true on D and Σ_T under the stable model semantics. We thus obtain immediately the following result:

Theorem 6.3. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ_T and Σ_C be sets of NTGDs and normal constraints on $\mathcal{R}$, respectively, and Q be a UNBCQ on $\mathcal{R}$. Then, $(D, \Sigma_T \cup \Sigma_C) \models_{stable} Q$ iff $(D, \Sigma_T) \models_{stable} Q \vee \bigvee_{\sigma \in \Sigma_C} Q_\sigma$.*

As an immediate consequence of Theorem 6.3, we obtain that adding a set of normal constraints does not increase the complexity of answering covered UNBCQs on databases relative to guarded NTGDs under the stable model semantics.

Corollary 6.4. *Answering covered UNBCQs on databases relative to guarded NTGDs and covered normal constraints under the stable model semantics has the same (combined and data) complexity as answering covered UNBCQs on databases relative to guarded NTGDs alone under the stable model semantics.*

In particular, by Theorems 6.3 and 5.1, we immediately obtain the following complexity results:

Corollary 6.5. *Given as input a database D for a schema $\mathcal{R}$, finite sets Σ_T and Σ_C of guarded NTGDs and covered normal constraints on $\mathcal{R}$, respectively, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma_T \cup \Sigma_C) \models_{stable} Q$ has the following complexity:*

(1) 2-EXPTIME-*complete in combined complexity.*
(2) EXPTIME-*complete, if* $\mathrm{ar}(\mathcal{R})$ *is bounded by a constant, and* $Q \vee \bigvee_{\sigma \in \Sigma_C} Q_\sigma$ *is acyclic.*
(3) co-NP-*complete in data complexity, or if* $|\mathcal{R}|$, $\mathrm{ar}(\mathcal{R})$, *and* $\mathrm{wd}(Q \vee \bigvee_{\sigma \in \Sigma_C} Q_\sigma)$ *are bounded by a constant.*

By combining Theorems 6.3 and 5.8, we also get co-NP-completeness for (not necessarily covered) normal constraints, and the upper bounds co-3-NEXPTIME for combined complexity, and co-2-NEXPTIME for bounded arity and atomic or acyclic queries.

6.2 Equality-generating Dependencies (EGDs) and Keys

We now add EGDs, which are also important when representing ontologies. Note that EGDs generalize *functional dependencies* (*FDs*) (which informally encode that certain attributes of a relation functionally depend on others) and, in particular, *key dependencies* (or *keys*) (which are informally

tuple-identifying sets of attributes of a relation) [1]. In many DLs, such as *DL-Lite* [38, 103], general EGDs cannot be formulated, but only keys. Therefore, we mainly focus on keys here. We transfer a result about *non-key-conflicting (NKC)* inclusion dependencies [36] and about non-conflicting keys in guarded Datalog$^{\pm}$ [34] to the more general setting of normal (and stratified disjunctive normal) guarded Datalog$^{\pm}$.

However, while adding covered normal constraints is effortless from a computational perspective, adding EGDs is more problematic: The interaction of TGDs and EGDs leads to undecidability of query answering even in simple cases, such that of functional and inclusion dependencies [40, 95], or keys and inclusion dependencies (see, e.g., Reference [36]), where this undecidability result is proven in a similar way as a result attributed to Vardi, which was reported and proven in Corollary 2.3 of Reference [78]). Even a *fixed* set of EGDs and guarded TGDs can simulate a universal Turing machine, and thus query answering and even propositional ground atom inference is undecidable for such dependencies. For this reason, we consider a restricted class of EGDs, namely, *strongly non-conflicting keys (SNC keys)*, which show (in the guarded case) a controlled interaction with NTGDs and stratified DNTGDs (and covered normal constraints), such that they do not increase the complexity of answering covered UNBCQs. Intuitively, the main underlying idea is to restrict the interplay between TGDs and EDGs by requiring a semantic property of separability, which is implied by a syntactic property of strongly non-conflicting TGDs and EGDs, so it is sufficient to check the EGDs on the database only (returning a positive query answer if satisfied and a negative one if not satisfied) and otherwise ignore the EGDs.

We first define the notion of a stable (respectively, canonical) model of a database relative to a set of NTGDs (respectively, a stratified set of DNTGDs) and a set of EGDs as follows:

Definition 6.6. Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ_T and Σ_E be a set of NTGDs (respectively, a stratified set of DNTGDs) and a set of EGDS on $\mathcal{R}$, respectively. A *stable* (respectively, *canonical*) *model* for D and $\Sigma = \Sigma_T \cup \Sigma_E$ is a stable (respectively, canonical) model S for D and Σ_T such that $S \models \sigma$ for all $\sigma \in \Sigma_E$. We denote by $SMod(D, \Sigma)$ (respectively, $CModD\Sigma$) the set of all stable (respectively, canonical) models for D and Σ. ◁

The following theorem is an extension of Theorem 4.14 to the setting of EGDs, which formally states the result that the translation of guarded NTGDs into stratified sets of guarded DNTGDs in Theorem 4.14 also carries over to the more general case where we additionally have EGDs:

Theorem 6.7. *There is a polynomial time algorithm that, given a finite set Σ_T of guarded NTGDs and a set Σ_E of EGDs over a schema $\mathcal{R}$, outputs a stratified set Σ'_T of guarded DNTGDs over a schema $\mathcal{R}'$ with $|\mathcal{R}'| \leqslant 5|\mathcal{R}| + |\Sigma_T| + 1$ and $\mathrm{ar}(\mathcal{R}') \leqslant 2 \cdot \mathrm{ar}(\mathcal{R})$ such that the following holds for each database D for $\mathcal{R}$:*

(1) For each $I \in SMod(D, \Sigma)$, there exists a $J \in CMod(D, \Sigma')$ such that $I = \{R(\mathbf{a}) \mid R \in \mathcal{R}, R^+(\mathbf{a}) \in J\}$ and $Fail \notin J$.
(2) For each $J \in CMod(D, \Sigma')$ with $Fail \notin J$, the interpretation $I := \{R(\mathbf{a}) \mid R \in \mathcal{R}, R^+(\mathbf{a}) \in J\}$ is in $SMod(D, \Sigma)$.

Here, R^+ (for each $R \in \mathcal{R}$) and $Fail$ are distinguished predicates in $\mathcal{R}' \setminus \mathcal{R}$, $\Sigma = \Sigma_T \cup \Sigma_E$, $\Sigma' = \Sigma'_T \cup \Sigma'_E$, and Σ'_E is obtained from Σ_E by replacing every $R \in \mathcal{R}$ by R^+.

Proof. Immediate by the proof of Theorem 4.14, since $I \models \sigma$ for $\sigma \in \Sigma_E$ in (1) and (2) is equivalent to $I \models \sigma'$, where $\sigma' \in \Sigma'_E$ is obtained from σ by replacing every $R \in \mathcal{R}$ by R^+. □

The following definition generalizes the notion of *separability* for TGDs from Reference [33] to DNTGDs. Here, there is a *hard violation* of an EGD σ on $\mathcal{R}$ of the form $\Phi(\mathbf{x}) \rightarrow x_i = x_j$ in an interpretation I (respectively, a database D) for $\mathcal{R}$ iff there exists a homomorphism $\eta\colon \Phi(\mathbf{x}) \rightarrow I$ (respectively, $\eta\colon \Phi(\mathbf{x}) \rightarrow D$) such that $\eta(x_i)$ and $\eta(x_j)$ cannot be unified via homomorphisms (respectively, are different constants in Δ).

Definition 6.8. Let $\mathcal{R}$ be a relational schema, and let Σ_T and Σ_E be a stratified set of DNTGDs and a set of EGDs on $\mathcal{R}$, respectively. Then, Σ_E is *separable* from Σ_T relative to a class of queries C if for every database D for $\mathcal{R}$, the following conditions (i) and (ii) are both satisfied:

(i) If there is a hard violation of an EGD of Σ_E in some $S \in \mathit{CMod}(D, \Sigma_T \cup \Sigma_E)$, then there is also a hard violation of some EGD of Σ_E in D.
(ii) If there is no hard violation of an EGD of Σ_E in D, then for every query Q of class C, it holds that Q is true in all $S \in \mathit{CMod}(D, \Sigma_T \cup \Sigma_E)$ iff Q is true in all $S \in \mathit{CMod}(D, \Sigma_T)$. ◁

The next definition extends the notion of *non-conflicting (NC)* keys for TGDs from Reference [33] to DNTGDs.

Definition 6.9. Let $\mathcal{R}$ be a relational schema, and let Σ_T and Σ_K be a set of DNTGDs and a set of keys on $\mathcal{R}$, respectively.

- Let $\kappa \in \Sigma_K$ be defined on a predicate r, and let $\sigma \in \Sigma_T$ be of the form $\Phi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \Psi(\mathbf{x}, \mathbf{z})$. The key κ is *non-conflicting (NC)* with σ if for every disjunct ψ in $\Psi(\mathbf{x}, \mathbf{z})$, either (i) r is different from the relational predicate in ψ; or (ii) the positions of κ in r are not a proper subset of the $\mathbf{x}$-positions in r in ψ, and every variable in $\mathbf{z}$ appears only once in the head of σ.
- A key $\kappa \in \Sigma_K$ is *non-conflicting (NC)* with Σ_T if κ is NC with every DNTGD in Σ_T.
- We say Σ_K is *non-conflicting (NC)* with Σ_T if every key in Σ_K is NC with Σ_T. ◁

It was shown in Reference [33] that if Σ_K is a set of keys that is NC with a set Σ_T of guarded TGDs, then Σ_K is separable from Σ_T. The following example shows that this is not the case for *stratified* sets Σ_T of guarded DNTGDs. Note, however, that the implication still holds for sets Σ_T of guarded disjunctive TGDs. That is, the implication does not hold for stratified guarded DNTGDs due to negated literals in rule bodies (as the example also shows) but not due to disjunctions in rule heads.

Example 6.10. Consider the following stratified set Σ_T of guarded NTGDs:

$$\{P(x) \rightarrow \exists y\, R(x, y),$$
$$R(x, y) \wedge \neg P(y) \rightarrow S(x)\},$$

the database $D = \{P(a), R(a, a)\}$, and the key $\kappa = \{1\}$ on R. Without the key, there is a model that contains $Q(a)$, thus $Q = \exists x\, (P(x) \wedge \neg S(x))$ evaluates to false. With the key, there is no model that contains $Q(a)$, thus Q evaluates to true.

We now define a stronger NC condition for DNTGDs, which will imply the desired separability. A *position* of a relational schema $\mathcal{R}$ is a pair (R, i), where $R \in \mathcal{R}$ and $i \in \{1, \ldots, \mathrm{ar}(R)\}$. Given a set (or Boolean combination) A of atoms, a constant or variable t *occurs at position* (R, i) *in* A if A has an atom $R'(t_1, \ldots, t_n)$ such that $R' = R$ and $t_i = t$.

Definition 6.11. Let $\mathcal{R}$ be a relational schema, let Σ_T be a stratified set of guarded DNTGDs on $\mathcal{R}$, and let Σ_K be a set of keys on $\mathcal{R}$. We define

$$K_0 := \{(R,i) \mid \text{there is a key } \kappa \in \Sigma_K \text{ defined on } R, \text{ and } i \in \{1,\ldots,\text{ar}(R)\} \setminus \kappa\},$$

$$K_{i+1} := \{(S,j) \mid \text{there is an } (R,i) \in K_i \text{ and a } \sigma \in \Sigma_T \text{ such that } (R,i) \text{ is a position in } body^+(\sigma) \text{ and } (S,j) \text{ is a position in } head(\sigma) \text{ that contains the same variable as position } (R,i) \text{ in } body^+(\sigma)\},$$

$$K_\infty := \bigcup_{i \geqslant 0} K_i.$$

We say Σ_K is ***strongly NC (SNC)*** with Σ_T if Σ_K is NC with Σ_T, and there is no DNTGD $\sigma \in \Sigma_T$ that contains a negative literal with a variable that occurs in $body^+(\sigma)$ at a position in K_∞. A single key κ is *SNC* with Σ_T if $\{\kappa\}$ is SNC with Σ_T. ◁

We illustrate the concept of SNC sets of keys and sets of TGDs by the following example:

Example 6.12. Consider again the stratified set Σ_T of guarded NTGDs, the database $D = \{P(a), R(a,a)\}$, and the key $\kappa = \{1\}$ on R of Example 6.10. Then, κ and thus also $\Sigma_K = \{\kappa\}$ are not SNC with Σ_T, as $K_\infty = \{R(2)\}$, and the second TGD in Σ_T contains the negative literal $\neg P(y)$, the positive literal $R(x,y)$, the variable y occurs in the former, and y also occurs in the latter at a position $R(2) \in K_\infty$. Let $\Sigma'_K = \{\kappa, \kappa'\}$ with $\kappa' = \{2\}$ on R. Then, $K_\infty = \emptyset$ for Σ'_K and Σ_T, and thus Σ'_K is SNC with Σ_T.

The following theorem shows that the stronger NC condition for DNTGDs implies the desired separability, i.e., if Σ_K is a set of keys that is SNC with a stratified set Σ_T of guarded DNTGDs, then Σ_K is separable from Σ_T.

Theorem 6.13. *Let $\mathcal{R}$ be a relational schema, let Σ_T be a stratified set of guarded DNTGDs on $\mathcal{R}$, and let Σ_K be a set of keys on $\mathcal{R}$. Then: If Σ_K is SNC with Σ_T, then Σ_K is separable from Σ_T relative to UBCQs.*

The following result generalizes Theorem 6.13 to strongly covered UNBCQs. Here, a UNBCQ Q is *strongly covered* if for every negative literal $\neg\alpha$ in Q, there exists a cover β of $\neg\alpha$ in Q such that all variables x from α that occur in β occur in β only at positions not in K_∞.

Theorem 6.14. *Let $\mathcal{R}$ be a relational schema, let Σ_T be a stratified set of guarded DNTGDs on $\mathcal{R}$, and let Σ_K be a set of keys on $\mathcal{R}$. Then: If Σ_K is SNC with Σ_T, then Σ_K is separable from Σ_T relative to strongly covered UNBCQs.*

As an immediate consequence from Theorems 6.3 and 6.14, since an SNC set of keys relative to a set of guarded NTGDs remains an SNC set of keys relative to its encoding as stratified set of guarded DNTGDs (see the proof of Theorem 4.14), we obtain that adding SNC keys and a set of strongly covered normal constraints does not increase the complexity of answering strongly covered UNBCQs on databases relative to guarded NTGDs under the stable model semantics. Here, a normal constraint σ is *strongly covered* if for every negative literal $\neg\alpha$ in σ there exists a cover β of $\neg\alpha$ in σ such that all variables x from α that occur in β occur in β only at positions not in K_∞. In particular, the complexity results of Theorem 5.1 immediately carry over to the more general case of answering strongly covered UNBCQs under additionally SNC keys and a set of strongly covered normal constraints.

Corollary 6.15. *Answering strongly covered UNBCQs on databases relative to guarded NTGDs, SNC keys, and strongly covered normal constraints under the stable model semantics has the same*

(combined and data) complexity as answering covered UNBCQs on databases relative to guarded NTGDs alone under the stable model semantics. In particular, given as input a database D for a schema $\mathcal{R}$, finite sets Σ_T and Σ_C of guarded NTGDs and strongly covered normal constraints on $\mathcal{R}$, respectively, a finite set Σ_K of keys on $\mathcal{R}$ that is SNC with Σ_T, and a strongly covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma_T \cup \Sigma_C \cup \Sigma_K) \models_{stable} Q$ has the following complexity:

(1) 2-EXPTIME-*complete.*
(2) EXPTIME-*complete, if* $\mathrm{ar}(\mathcal{R})$ *is bounded by a constant, and* $Q \vee \bigvee_{\sigma \in \Sigma_C} Q_\sigma$ *is acyclic.*
(3) co-NP-*complete in data complexity (i.e., for fixed* Q *and fixed* Σ*), or if* $|\mathcal{R}|$, $\mathrm{ar}(\mathcal{R})$, *and* $\mathrm{wd}(Q \vee \bigvee_{\sigma \in \Sigma_C} Q_\sigma)$ *are bounded by a constant.*

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 nonmonotonic negation. The proposed logics are essentially the same as in Reference [66], but the crucial difference is that nonmonotonic negation is now interpreted via the stable model semantics. This can lead to better query answers, as Example 7.8 below demonstrates. We demonstrate how to extend *DL-Lite*$_\mathcal{R}$ (which is underlying the OWL 2 QL profile), *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{A}$ [38, 103], and $\mathcal{ELHI}$ [9] with nonmonotonic negation under the stable model semantics. That is, we confine our discussion to the most paradigmatic examples of lightweight DLs. It is to be expected, however, that our techniques can be adapted to an even larger variety of DLs (involving features such as n-ary predicates or number restrictions). In the sequel, a *normal Datalog*$^\pm$ *program* $\Sigma = \Sigma_T \cup \Sigma_C \cup \Sigma_K$ consists of a finite set Σ_T of guarded NTGDs, a finite set Σ_C of strongly covered normal constraints, and a finite set Σ_K of keys that are SNC with Σ_T. If $\Sigma_K = \emptyset$, then Σ_C may also be a finite set of only covered normal constraints.

7.1 The *DL-Lite*$_{\mathrm{not}}$-family

We first discuss the extension of members of the *DL-Lite*-family with nonmonotonic negation. Here, we only recall the definition for *DL-Lite*$_\mathcal{A}$ in detail—*DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_\mathcal{F}$ are easily obtained as the usual fragments extending *DL-Lite*$_{\mathrm{core}}$ with role inclusions and functionality constraints, respectively.

Definition 7.1. Let $\mathbf{A}, \mathbf{R}_A, \mathbf{R}_D$, and $\mathbf{I}$ be sets of atomic concepts, atomic roles, atomic attributes, and individuals, respectively, and let D be a set of basic data types, where each $d \in D$ is associated with a set of values V_d. Then, we define *basic* concepts B, general concepts C, *basic* roles R, general roles E, value-domain expressions T and F, and attribute expressions V_C using the following grammar:

- Concept expressions:

$$\begin{array}{rcl} B & \longrightarrow & A \mid \exists R \mid \delta(U_C)\,, \\ C & \longrightarrow & B \mid \neg B \mid \top_C \mid \exists R.C\,. \end{array}$$

- Role expressions:

$$\begin{array}{rcl} R & \longrightarrow & Q \mid Q^-\,, \\ E & \longrightarrow & R \mid \neg R\,. \end{array}$$

- Value-domain expressions:

$$\begin{array}{rcl} T & \longrightarrow & \rho(U_C)\,, \\ F & \longrightarrow & \top_D \mid d\,. \end{array}$$

- Attribute expressions:

$$V_C \longrightarrow U_C \mid \neg U_C\,.$$

Here, $A \in \mathbf{A}$, $Q \in \mathbf{R}_A$, Q^- denotes the inverse of an atomic role $Q \in \mathbf{R}_A$, $U_C \in \mathbf{R}_D$, $\delta(U_C)$ and $\rho(U_C)$ denote the domain and the range of the atomic attribute U_C, respectively, and $d \in D$. As usual, for an arbitrary basic role R, we write R^- to denote its inverse, i.e., $(Q)^- = Q^-$ and $(Q^-)^- = Q$, for each atomic role Q.

A *DL-Lite*$_{\mathcal{A}}$-TBox is a finite set of *concept inclusion axioms* of the form $B \sqsubseteq C$ (with B a basic concept and C an arbitrary concept), *role inclusion axioms* of the form $R \sqsubseteq E$, *role functionality axioms* funct(R) that express that a basic role R is functional, *value-domain inclusion axioms* of the form $T \sqsubseteq F$, *attribute inclusion axioms* $U_C \sqsubseteq V_C$, and *attribute functionality axioms* of the form funct(U_C). If a TBox $\mathcal{T}$ contains functionality axiom for a role/attribute, then we call the role/attribute an *identifying property*. The following condition is required to hold for *DL-Lite*$_{\mathcal{A}}$-TBoxes $\mathcal{T}$:

> For any identifying property S in $\mathcal{T}$, S does not occur positively on the right-hand side of a role/attribute inclusion axiom in $\mathcal{T}$ or in an expression of the form $\exists S.C$. (*)

An Abox $\mathcal{A}$ is a finite set of membership axioms of the form $A(a)$, $Q(a, b)$, and $U_C(a, v)$, where A is an atomic concept, Q an atomic role, a, b are constants in $\mathbf{I}$, U_C is an atomic attribute, and v is a value.

The weaker logic *DL-Lite*$_{\text{core}}$ disallows basic concepts $B = \delta(U_C)$, general concepts $C = \top_C$ and $C = \exists R.C$, and all value-domain and attribute expressions; it then only allows for concept inclusion axioms $B \sqsubseteq C$ in the TBox and membership axioms $A(a)$ and $Q(a, b)$ in the ABox. The more expressive *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$ additionally allow for role inclusion axioms $R \sqsubseteq E$ and role functionality axioms funct(R) in the TBox, respectively. Thus, the condition (*) is vacuous for *DL-Lite*$_{\text{core}}$, *DL-Lite*$_{\mathcal{R}}$, and *DL-Lite*$_{\mathcal{F}}$, as *DL-Lite*$_{\text{core}}$ and *DL-Lite*$_{\mathcal{R}}$ do not allow for role/attribute functionality axioms, and *DL-Lite*$_{\mathcal{F}}$ has no role/attribute inclusion axioms and no general concepts $\exists R.C$.

In the following, let $X \in \{\mathcal{R}, \mathcal{F}, \mathcal{A}, \text{core}\}$. A *DL-Lite*$_X$ knowledge base is a *DL-Lite*$_X$-TBox together with an ABox. The semantics is defined as usual via interpretations $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that interpret the atomic concepts and roles as subsets and binary relations over Δ, respectively, the atomic attributes as relations between elements of Δ and data values, and individuals as elements of Δ. Furthermore, interpretations are inductively extended to general concepts, roles, and attributes. For each DL *DL-Lite*$_X$, the variant *DL-Lite*$_{X,\sqcap}$ is defined by allowing for (finite) conjunctions of basic concepts or basic roles on the left-hand side of concept and role inclusions, respectively (i.e., by allowing for concept inclusions of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq C$ and role inclusions of the form $R_1 \sqcap \cdots \sqcap R_m \sqsubseteq E$). ◁

It has been shown in Reference [33] how to translate the DLs from the *DL-Lite*-family into Datalog$^{\pm}$. Before we recall this translation, we first introduce our own new extensions of these DLs with nonmonotonic negation, which essentially extends *DL-Lite*$_X$ by the possibility of adding an arbitrary number of negative side conditions to the left-hand side of an inclusion.

Definition 7.2. Let $X \in \{\mathcal{R}, \mathcal{F}, \mathcal{A}, \text{core}\}$. A *DL-Lite*$_{X,\text{not}}$-TBox can contain concept or role inclusions of the form $U \sqcap \text{not}U'_1 \sqcap \cdots \sqcap \text{not}U'_n \sqsubseteq V$, where $U \sqsubseteq V$ and $U'_1 \sqcap \cdots \sqcap U'_n \sqsubseteq V$, with $n \geqslant 0$, are concept or role inclusions in *DL-Lite*$_X$ and *DL-Lite*$_{X,\sqcap}$, respectively. In addition, a *DL-Lite*$_{\mathcal{F},\text{not}}$ or *DL-Lite*$_{\mathcal{A},\text{not}}$ TBox $\mathcal{T}$ may contain role functionality axioms funct(Q) and funct(Q^-) as well as attribute functionality axioms funct(U_C) under the condition that (*) holds. ◁

Remark 7.3. The above family of logics *DL-Lite*$_{X,\neg}$ could be easily extended to variants *DL-Lite*$_{X,\sqcap,\neg}$ that allow an arbitrary number of positive concepts or roles on the left-hand side of a concept or role inclusion, respectively. Any complexity result in this section that is proven for

DL-Lite$_{X,\neg}$ carries easily over to the corresponding extension *DL-Lite*$_{X,\sqcap,\neg}$. To simplify the presentation, we opted for confining our discussion to the logics *DL-Lite*$_{X,\neg}$.

To provide a semantics for the logics *DL-Lite*$_{X,\text{not}}$, we use a translation into databases under normal Datalog$^{\pm}$ programs. First, data values v are mapped to constants $c_v \in \Delta$ such that any two data domains are mapped to disjoint sets of constants, and individuals i are mapped to constants $c_i \in \Delta$ different from the constants that the data values are mapped to. Then, we fix for every atomic concept A a unary predicate P_A, for each atomic role Q a binary predicate P_Q and two unary predicates $P_{\exists Q}$ and $P_{\exists Q^-}$, for each attribute U a binary predicate P_U and a unary predicate $P_{\delta(U)}$, and for each data domain $d \in D$ a unary predicate P_d, along with the normal constraints $P_d(x) \wedge P_{d'}(x) \rightarrow \bot$ for any two data domains $d, d' \in D$. Furthermore, we note that any inclusion axiom of the form $U_1 \sqcap \cdots \sqcap U_m \sqsubseteq \exists R.C$ can be replaced by the axioms $U_1 \sqcap \cdots \sqcap U_m \sqsubseteq \exists R.A$ and $A \sqsubseteq C$ for a fresh atomic concept A, and thus we can assume without loss of generality for any concept of the form $\exists R.C$ on the right-hand side of a conclusion that C is atomic. The translation of a *DL-Lite*$_{X,\text{not}}$ knowledge base $(\mathcal{A}, \mathcal{T})$ into a database $D_{\mathcal{A}}$ under a normal Datalog$^{\pm}$ program $\Sigma_{\mathcal{T}}$ is then obtained as follows:

- All membership axioms $A(a)$, $Q(a,b)$, and $U(a,v)$ in $\mathcal{A}$ are translated into atoms $P_A(c_a)$, $P_Q(c_a,c_b)$, and $P_U(c_a,c_v)$ in the database $D_{\mathcal{A}}$, respectively.
- For each atomic role Q and each attribute U, the normal Datalog$^{\pm}$ program $\Sigma_{\mathcal{T}}$ contains the rules

$$\begin{array}{lcl} P_Q(y,x) & \rightarrow & P_{\exists Q^-}(x), \\ P_Q(x,y) & \rightarrow & P_{\exists Q}(x). \end{array} \qquad\qquad P_U(x,y) \rightarrow P_{\delta(U)}(x),$$

- For each concept inclusion axiom $B \sqcap \text{not}\, B'_1 \sqcap \cdots \sqcap \text{not}\, B'_m \sqsubseteq C$, we add the rule $\sigma(B) \wedge \neg\, \sigma(B'_1) \wedge \cdots \wedge \neg\, \sigma(B'_m) \rightarrow \tau(C)$, where

$$\sigma(B) = \begin{cases} P_A(x), & \text{if } B = A \text{ for an atomic concept } A, \\ P_{\exists R}(x), & \text{if } B = \exists R \text{ for a basic role } R, \\ P_{\delta(U)}(x), & \text{if } B = \delta(U) \text{ for an attribute } U \end{cases}$$

and

$$\tau(C) = \begin{cases} P_A(x), & \text{if } C = A \text{ for an atomic concept } A, \\ \exists y.P_Q(x,y), & \text{if } C = \exists Q \text{ for an atomic role } Q, \\ \exists y.P_Q(y,x) & \text{if } C = \exists Q^- \text{ for an atomic role } Q, \\ P_{\delta(U)}(x), & \text{if } C = \delta(U) \text{ for an attribute } U, \\ \exists y.P_Q(x,y) \wedge P_A(y), & \text{if } C = \exists Q.A \text{ for atomic } A \text{ and atomic } Q, \\ \exists y.P_Q(y,x) \wedge P_A(y) & \text{if } C = \exists Q^-.A \text{ for atomic } A \text{ and atomic } Q. \end{cases}$$

- For each concept inclusion axiom $B_0 \sqcap \text{not}\, B'_1 \sqcap \cdots \sqcap \text{not}\, B'_m \sqsubseteq \neg B$, we add the normal constraint $\sigma(B_0) \wedge \neg\ \sigma(B'_1) \wedge \cdots \wedge \neg\ \sigma(B'_m) \wedge \sigma(B) \rightarrow \bot$, where σ is defined as in the previous case.
- For each role inclusion axiom $R_0 \sqcap \text{not}\, R'_1 \sqcap \cdots \sqcap \text{not}\, R'_m \sqsubseteq R$, we add the rule $\sigma(R_0) \wedge \neg\, \sigma(R'_1) \wedge \cdots \wedge \neg\, \sigma(R'_m) \rightarrow \sigma(R)$, where $\sigma(R) = P_Q(x,y)$, if $R = Q$, and $\sigma(R) = P_Q(y,x)$, if $R = Q^-$.
- For each role inclusion axiom $R_0 \sqcap \text{not}\, R'_1 \sqcap \cdots \sqcap \text{not}\, R'_m \sqsubseteq \neg R$, we add the normal constraint $\sigma(R_0) \wedge \neg\, \sigma(R'_1) \wedge \cdots \wedge \neg\, \sigma(R'_m) \wedge \sigma(R) \rightarrow \bot$, where σ is defined as in the previous case.
- For all role functionality axioms $\text{funct}(Q)$ and $\text{funct}(Q^-)$, we add the EGDs

$$\begin{array}{lcl} Q(x,y) \wedge Q(x,y') & \rightarrow & y = y', \\ Q(x,y) \wedge Q(x',y) & \rightarrow & x = x'. \end{array}$$

- For each attribute functionality axiom $\mathrm{funct}(U)$, we add the EGD

$$U(x,y) \wedge U(x,y') \quad \rightarrow \quad y = y'.$$

- For all attribute inclusion axioms $U \sqsubseteq U'$ and $U \sqsubseteq \neg U'$, we add the rule $P_U(x,y) \rightarrow P_{U'}(x,y)$ and the normal constraint $P_U(x,y) \wedge P_{U'}(x,y) \rightarrow \bot$, respectively.
- For each value-domain inclusion axiom $\rho(U) \sqsubseteq d$, we add the rule $P_U(x,y) \rightarrow P_d(y)$.

It is not difficult to see that all produced rules are guarded and that all produced normal constraints are covered. In addition, in the presence of functionality axioms, we assume that *DL-Lite*$_{X,\mathrm{not}}$ knowledge bases are admissible: A *DL-Lite*$_{X,\mathrm{not}}$ knowledge base $(\mathcal{A},\mathcal{T})$ is *admissible*, if Σ_K is SNC with Σ_T, and all normal constraints in Σ_C are strongly covered relative to Σ_T, where Σ_K, Σ_T, and Σ_C are the sets of all keys, NTGDs, and normal constraints in $\Sigma_{\mathcal{T}}$. Furthermore, in the presence of functionality axioms, any UNBCQ to such knowledge bases has to be strongly covered. We define the stable model semantics for *DL-Lite*$_{X,\mathrm{not}}$ knowledge bases as follows:

Definition 7.4. Let $(\mathcal{A},\mathcal{T})$ be a *DL-Lite*$_{X,\mathrm{not}}$ knowledge base. Then, a *stable model* for $(\mathcal{A},\mathcal{T})$ is a stable model for $(D_{\mathcal{A}},\Sigma_{\mathcal{T}})$. The *answer* to a UNBCQ Q on $(\mathcal{A},\mathcal{T})$ *under the stable model semantics* is *Yes*, denoted $(\mathcal{A},\mathcal{T}) \models_{stable} Q$, if the answer to Q on $(D_{\mathcal{A}},\Sigma_{\mathcal{T}})$ is Yes. ◁

7.2 The Logic $\mathcal{ELHI}_{\mathrm{not}}$

Similarly to what we have done with *DL-Lite*, we now extend the DL $\mathcal{ELHI}$ with nonmonotonic negation and show how to translate knowledge bases for the resulting logic $\mathcal{ELHI}_{\mathrm{not}}$ into normal guarded Datalog$^{\pm}$ programs to obtain a stable model semantics for $\mathcal{ELHI}_{\mathrm{not}}$.

Definition 7.5. We assume a set $\mathbf{A}$ of atomic concepts, a set $\mathbf{R}_A$ of atomic roles, and a set $\mathbf{I}$ of individuals, and we recall the notions of concepts and roles in $\mathcal{ELHI}$:

$$\begin{aligned} C &\longrightarrow A \mid C \sqcap C \mid \exists R.C \mid \top, \\ R &\longrightarrow Q \mid Q^{-}, \end{aligned}$$

where $A \in \mathbf{A}$ and $Q \in \mathbf{R}_A$ are arbitrary atomic concepts and roles, respectively. An $\mathcal{ELHI}$ TBox is a finite set of concept inclusion axioms $C \sqsubseteq D$ for arbitrary concepts C, D and of role inclusion axioms $R \sqsubseteq S$ for arbitrary roles R, S.

We define an $\mathcal{ELHI}_{\mathrm{not}}$ TBox to be a finite set of

- $\mathcal{ELHI}$ axioms,
- concept inclusion axioms of the form $C_1 \sqcap \cdots \sqcap C_m \sqcap \mathrm{not}\, C'_1 \sqcap \cdots \sqcap \mathrm{not}\, C'_n \sqsubseteq D$, where $C_1 \sqcap \cdots \sqcap C_m \sqsubseteq D$ and $C'_1 \sqcap \cdots \sqcap C'_n \sqsubseteq D$, with $m, n > 0$, are $\mathcal{ELHI}$ axioms, and at least one $1 \leqslant j \leqslant m$ exists with $C_j \neq \top$, and
- role inclusion axioms of the form $R \sqcap \mathrm{not}\, R' \sqsubseteq S$, where R, R', and S are arbitrary roles.

It is not difficult to see that concept inclusion axioms can be assumed to be of a certain normal form

$$C_1 \sqcap \cdots \sqcap C_n \sqcap \mathrm{not}\, C'_1 \sqcap \cdots \sqcap \mathrm{not}\, C'_m \sqsubseteq D,$$

where $n > 0$, $m \geqslant 0$, at most one of the C_i's has the form $\exists R.A$, all other concepts (including all negated ones) on the left-hand side are atomic or $\top$, and D is either an atomic concept A or of the form $\exists R.A$ for an atomic concept A. ◁

For the translation of a $\mathcal{ELHI}_{\mathrm{not}}$-TBox $\mathcal{T}$ into a normal Datalog$^{\pm}$ program $\Sigma_{\mathcal{T}}$, we assume for every concept in $\mathbf{A}$ a unary predicate P_A, for every role $Q \in \mathbf{R}_A$ a binary predicate P_Q, and we

define a mapping τ on (certain) concepts and roles as follows:

$$\begin{array}{ll} \tau(\top) = U(x), & \tau(A) = P_A(x), \\ \tau(Q) = P_Q(x,y), & \tau(Q^-) = P_Q(y,x), \\ \tau(\exists R.A) = \tau(R) \wedge P_A(y). & \end{array}$$

The definition of τ gives rise to the following translation of $\mathcal{ELHI}_{\text{not}}$ axioms into normal guarded Datalog$^{\pm}$:

- every concept inclusion axiom $C_1 \sqcap \cdots \sqcap C_n \sqcap \text{not } C'_1 \sqcap \cdots \sqcap \text{not } C'_m \sqsubseteq D$ in normal form is mapped to the rule

$$\tau(C_1) \wedge \cdots \wedge \tau(C_n) \wedge \neg\, \tau(C'_1) \wedge \cdots \wedge \neg\, \tau(C'_m) \rightarrow \exists y.\tau(D),$$

- every role inclusion $R_1 \sqcap \text{not} R_2 \sqsubseteq S$ is mapped to $\tau(R_1) \wedge \neg\, \tau(R_2) \rightarrow \tau(S)$, and
- for every predicate Q(x,y) or P(x) in the translation, we add the rules

$$Q(x,y) \rightarrow U(x), \qquad Q(x,y) \rightarrow U(y), \qquad P(x) \rightarrow U(x).$$

  Hence, U holds for all constants and nulls that occur in a stable model. In this way, $U(x)$ encodes the universal concept $\top$, which holds everywhere in a given model.

Observe that the above translation produces no keys, all produced rules are guarded, and all produced normal constraints are covered. We define the stable model semantics for $\mathcal{ELHI}_{\text{not}}$ knowledge bases as follows:

Definition 7.6. Let $(\mathcal{A}, \mathcal{T})$ be an $\mathcal{ELHI}_{\text{not}}$ knowledge base. Then, a *stable model* for $(\mathcal{A}, \mathcal{T})$ is a stable model for $(D_{\mathcal{A}}, \Sigma_{\mathcal{T}})$. The *answer* to a UNBCQ Q on $(\mathcal{A}, \mathcal{T})$ *under the stable model semantics* is *Yes*, denoted $(\mathcal{A}, \mathcal{T}) \models_{\textit{stable}} Q$, if the answer to Q on $(D_{\mathcal{A}}, \Sigma_{\mathcal{T}})$ is Yes. ◁

The following example demonstrates how Example 1.1 from the introduction can be formalized in $\mathcal{ELHI}_{\text{not}}$:

Example 7.7. Consider the ABox $\mathcal{A} = \{\text{Person(mary)}\}$ and the TBox $\mathcal{T}$ consisting of the following axioms:

$$\begin{array}{rcl} \text{Person} \sqcap \text{not Even} & \sqsubseteq & \text{Odd}, \\ \text{Person} \sqcap \text{not Odd} & \sqsubseteq & \text{Even}, \\ \text{Person} \sqcap \text{Even} & \sqsubseteq & \exists \text{hasParent}.(\text{Odd} \sqcap \text{Person}), \\ \text{Person} \sqcap \text{Odd} & \sqsubseteq & \exists \text{hasParent}.(\text{Even} \sqcap \text{Person}), \\ \text{hasParent} & \sqsubseteq & \text{hasChild}^-. \end{array}$$

Then, we obtain two (infinite) stable models for this knowledge base $(\mathcal{A}, \mathcal{T})$, which look very similar to the stable models obtained in Example 1.1. The two stable models correspond to the case in which Odd(mary) and to the case in which Even(mary) holds. If we add the role inclusion

$$\text{Person} \sqcap \text{not} \exists \text{hasChild} \sqsubseteq \text{Odd},$$

then the resulting knowledge base has only one stable model, namely, the one that contains Odd(mary).

In the following example of a knowledge base in *DL-Lite*$_{\mathcal{R},\text{not}}$, the stable model semantics leads to better query answers than the well-founded semantics:

Example 7.8. Consider the ABox {FiveStar(ritz)} and the TBox consisting of the following axioms:

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

Then, the well-founded semantics of this knowledge base only contains the atoms FiveStar(ritz) and Hotel(ritz), while all its stable models also contain the atoms ∃SwimOpp(ritz) and Excellent(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. This example thus demonstrates an advantage of the stable model over the well-founded semantics.

We finally consider an example that makes use of a key.

Example 7.9. Consider the ABox $\{\text{Emp}(a), \text{Emp}(b), \text{Probation}(a), \text{hasSuper}(b, c)\}$ and the TBox consisting of the following axioms where the first one says that everybody has at most one supervisor, the next two that every employer has a supervisor, unless he is a manager, and the last one that every employer in probation has a supervisor:

$$\begin{aligned} &\text{funct(hasSuper)},\\ \text{Emp} \sqcap \text{not Manager} &\sqsubseteq \exists\text{hasSuper},\\ \text{Emp} \sqcap \text{not } \exists\text{hasSuper} &\sqsubseteq \text{Manager},\\ \text{Probation} &\sqsubseteq \text{hasSuper}. \end{aligned}$$

It is not difficult to see that the ABox satisfies the key; thus, the key can be ignored for the purpose of query answering. The BCQ $\exists x.\text{Emp}(x) \wedge \text{Manager}(x)$ evaluates to false, as (i) a is not a manager, as he is in probation, and thus, by the TBox axioms, he has a supervisor, and (ii) b is not a manager, as he has a supervisor according to the ABox.

The decidability results for query answering relative to the stable model semantics can be directly applied to DLs. Here, by a covered UNBCQ Q over some knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we mean a covered UNBCQ over literals of the form $P_R(x, y)$, $\text{not}P_R(x, y)$, $P_A(x)$, and $\text{not}P_A(x)$, where R is an atomic role in $\mathcal{K}$, A is an atomic concept in $\mathcal{K}$, P_R and P_A are their translations into predicates, and x and y are variables. We often use the translations $P_R(x, y) := R(x, y)$ and $P_A(x) := A(x)$ such that a UNBCQ over $\mathcal{K}$ is essentially the same as a UNBCQ over the "schema" of $\mathcal{K}$.

Theorem 7.10. *Let $\mathcal{L}$ be DL-Lite$_{\mathcal{X},\text{not}}$, were $\mathcal{X} \in \{\mathcal{R}, \mathcal{F}, \mathcal{A}\}$, or $\mathcal{ELHI}_{\text{not}}$. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base in $\mathcal{L}$ and Q a covered UNBCQ. Furthermore we assume that, in case $\mathcal{T}$ contains functionality axioms, $\mathcal{K}$ and Q are admissible and strongly covered, respectively. Then deciding whether Q is true under the stable model semantics has the following complexity:*

(1) 2-EXPTIME*-complete in combined complexity for $\mathcal{ELHI}_{\text{not}}$, and DL-Lite$_{\mathcal{X},\text{not}}$, where $X \in \{\mathcal{R}, \mathcal{A}\}$.* 2-EXPTIME*-hardness holds even if the TBox $\mathcal{T}$ is fixed and has no functionality axioms, and Q is a UBCQ.*

(2) co-NP*-complete in data complexity. Hardness holds even if Q is a UBCQ, and $\mathcal{T}$ has no functionality axioms, in particular, if $\mathcal{T}$ is a DL-Lite$_{\text{core,not}}$ TBox.*

Proof. The upper complexity bounds follow immediately from Corollary 6.15. We now prove the lower bounds.

Lower bound in Point 1: It suffices to show the lower bound for *DL-Lite*$_{\mathcal{R},\text{not}}$, which is more restrictive than (and a subformalism of) the other two description logics.

We use Proposition 5.3 (1) and show that each set Σ of binary NF-DIDs can be polynomially transformed via a translation τ^* into a TBox $\tau^*(\Sigma)$ formulated in *DL-Lite*$_{\mathcal{R},\text{not}}$, such that (D, Σ) and $(\mathcal{A}_D, \tau^*(\Sigma))$ are equivalent with regard to UBCQ-answering, where $\mathcal{A}_D$ is the ABox corresponding to D. Before doing so, we observe that Proposition 5.3 (1) can be slightly strengthened, which will make our translation easier. In fact, by inspecting the proof of Theorem 4.5 in Reference [28], it becomes clear that the 2-EXPTIME-hardness result of Proposition 5.3 (1) already holds for *restricted-normal-form DIDs (RNF-DIDs)*, which are NF-DIDs where, in addition, all rules of type 3 according to Definition 5.2 are of the form $R(x) \rightarrow R_1(x) \vee R_2(x)$, where x is a single variable, and R, R_1, and R_2 are thus unary predicate symbols. In fact, there are only two kinds of disjunctive rules in the proof of Theorem 4.5 of Reference [28]: Rules of the form $R(x) \rightarrow R_1(x) \vee R_2(x) \vee \cdots \vee R_r(x)$, which are easily converted by use of auxiliary predicates into rules of the desired form (as already described in Appendix A of Reference [28]) and the single rule

$$conf_{\exists}(x) \rightarrow \exists y(child_1(x, y) \wedge conf_{\forall}(y)) \vee \exists y(child_2(x, y) \wedge conf_{\forall}(y)).$$

The latter can be rewritten, for example, by the following set of seven rules, where A_1, A_2, B_1, and B_2 are auxiliary predicates:

$$conf_{\exists}(x) \rightarrow A_1(x) \vee A_2(x),$$

$$A_1(x) \rightarrow \exists y B_1(x, y), \qquad A_2(x) \rightarrow \exists y B_2(x, y),$$

$$B_1(x, y) \rightarrow child_1(x, y), \qquad B_2(x, y) \rightarrow child_2(x, y),$$

$$B_1(x, y) \rightarrow conf_{\forall}(y)), \qquad B_2(x, y) \rightarrow conf_{\forall}(y)).$$

This transformation yields the desired RNF-DIDs while not altering the answer of UBCQs Q over the original schema (i.e., where no auxiliary predicate occurs in Q).

Let Σ be an arbitrary set of RNF-DIDs and consider its translation τ into a set $\tau(\Sigma)$ of NTGDs, where τ is the polynomial transformation defined in the proof of Theorem 5.5. The obtained set $\tau(\Sigma)$ is, as shown in the proof of Theorem 5.5, equivalent to Σ with respect to UBCQ answering. It contains rules of four types: Types (1) and (2) according to Definition 5.2, where, moreover, all predicates are at most binary, rules of the form $R(x) \wedge \neg R_1(x) \rightarrow R_2(x)$ (namely, a pair for each RNF-DID of type (3)), and finally clauses of the form $R(x) \wedge R_1(x) \wedge R_2(x) \rightarrow$ Fail. By inspecting the argument in the proof of Theorem 5.5, we also see that the latter can be replaced by rules of the form $R(x) \wedge \neg \bar{R}_1(x) \wedge \neg \bar{R}_2(x) \rightarrow Fail$.

Now let $\tau^*(\Sigma)$ be obtained from $\tau(\Sigma)$ as follows: Each NTGD $\sigma \in \tau(\Sigma)$ of type (1) according to Definition 5.2 is either of the form $R(x, y) \rightarrow S(x, y)$, or $R(x, y) \rightarrow S(y, x)$, or $P(x) \rightarrow Q(x)$. Then, $\tau^*(\sigma)$ is $\mathrm{R} \sqsubseteq \mathrm{S}$, $\mathrm{R} \sqsubseteq \mathrm{S}^-$, or $\mathrm{P} \sqsubseteq \mathrm{Q}$, respectively. Each clause σ of type (2) is either of the form $R(x) \rightarrow \exists y S(x, y)$, or $R(x) \rightarrow \exists y S(y, x)$, and then $\tau^*(\sigma)$ is $\mathrm{R} \sqsubseteq \exists \mathrm{S}$ or $\mathrm{R} \sqsubseteq \exists \mathrm{S}^-$, respectively. For each rule σ of the form $R(x) \wedge \neg R_1(x) \rightarrow R_2(x)$, let $\tau^*(\sigma)$ be $\mathrm{R} \sqcap \text{not}\, R_1 \sqsubseteq R_2$. Finally, if σ is of the form $R(x) \wedge \neg R_1(x) \wedge \neg R_2(x) \rightarrow$ Fail, then $\tau^*(\sigma)$ is $\mathrm{R} \sqcap \text{not}\, \mathrm{R}_1 \sqcap \text{not}\, \mathrm{R}_2 \sqsubseteq \bot$. Here, we can assume without loss of generality to have the empty concept "$\bot$" available, even though it has not be explicitly included into *DL-Lite*$_{\mathcal{R}}$. The reason is that the negation "not" can be used to define $\bot$, for example, by a single rule $\bot \sqcap \text{not}\, C \sqsubseteq C$ (just like "*Fail*" can be defined with NTGDs under the stable model semantics).

Let D be a database on a binary schema, and let $\mathcal{A}_D$ be the corresponding ABox. Let us now observe that for the TBox $\mathcal{T} = \tau^*(\Sigma)$, its translation to $D_{\mathcal{T}}$ as defined earlier in this section is equivalent with respect to UBCQ answering to $\tau(\Sigma)$, modulo predicate renaming. Thus, for each

database D, and UBCQ Q, $(\mathcal{A}_D, \sigma^*(\Sigma)) \models_{stable} Q$ iff $(D, \sigma(\Sigma)) \models_{stable} Q$ iff $(D, \Sigma) \models_{strat} Q$. Given that our reduction σ^* is computable in polynomial time, the lower bound in Point 1 is established.

Lower bound in Point 2: It suffices to prove the lower bound for $DL\text{-}Lite_{\text{core,not}}$. We give a polynomial-time reduction from the problem of deciding if an undirected graph is not 3-colorable to query answering relative to a $DL\text{-}Lite_{\text{core,not}}$ knowledge base under the stable model semantics. Suppose we are given a graph (V, E). We define an ABox

$$D \quad = \quad \{V(n) \mid n \in V\} \cup \{E(n, m) \mid (n, m) \in E\}.$$

Furthermore, we consider the following TBox-axioms:

$$\begin{aligned} V \sqcap \text{not}C_1 \sqcap \text{not}C_2 \quad &\rightarrow \quad C_3, \\ V \sqcap \text{not}C_1 \sqcap \text{not}C_3 \quad &\rightarrow \quad C_2, \\ V \sqcap \text{not}C_3 \sqcap \text{not}C_2 \quad &\rightarrow \quad C_1. \end{aligned}$$

These axioms ensure that the stable models of the knowledge base correspond to colorings of the graph where each vertex is colored by exactly one of the colors C_1, C_2, or C_3. Consider now the union of the following queries:

$$\exists x \exists y \ (C_1(x) \wedge E(x, y) \wedge C_1(y)), \tag{12}$$

$$\exists x \exists y \ (C_2(x) \wedge E(x, y) \wedge C_2(y)), \tag{13}$$

$$\exists x \exists y \ (C_3(x) \wedge E(x, y) \wedge C_3(y)). \tag{14}$$

Our claim is that the query evaluates to true over the given knowledge base iff the graph (V, E) is not 3-colorable.

Suppose first that the graph is 3-colorable, fix such a 3-coloring, and consider the set of atoms

$$\{C_1(n) \mid n \in V \text{ has color 1}\} \cup \{C_2(n) \mid n \in V \text{ has color 2}\} \cup \{C_3(n) \mid n \in V \text{ has color 3}\}.$$

Then, it is not difficult to see that this set together with D forms a stable model of the above knowledge base, and the query evaluates to false due to the definition of a 3-coloring.

Suppose now the graph is not 3-colorable and consider an arbitrary stable model I of the above knowledge base. We color the vertices of the graph by the following rule:

$$\text{color vertex } n \text{ with color } i \qquad \text{iff} \qquad C_i(n) \in I.$$

Because the graph is not colorable, there must be two adjacent vertices n_1 and n_2 such that

$$C_j(n_1) \wedge E(n_1, n_2) \wedge C_j(n_2)$$

holds for some color j. Therefore, the query evaluates to true over the given stable model. As the stable model was arbitrary, we conclude that the query evaluates to true over the knowledge base, as required. □

8 RELATED WORK

Let us discuss a number of previous approaches to extend guarded TGDs with negation. However, none of the approaches below allows one to use the stable model semantics (derived via Skolemization from the standard stable model semantics in normal logic programming) for the general class of guarded normal TGDs.

Stratified negation. Stratified negation for guarded Datalog$^{\pm}$ (with negative constraints and non-conflicting keys), and thus automatically for a number of important DLs, was introduced in Reference [33] and further extended in Reference [8] to the more expressive formalism of *weakly guarded Datalog*$^{\pm}$ (which was introduced without negation in Reference [32]). 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 1.1 and 1.2.

Well-founded negation. Negation under the **well-founded semantics (WFS)** for guarded Datalog$^{\pm}$ and covered DLs was studied in References [66, 77] for two variants of the WFS. The version of Reference [77] 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 work in Reference [5] extends this approach to sticky existential rules with nonmonotonic negation and gives an analysis of the data and combined complexity of conjunctive query answering in the resulting formalism. The second variant of the WFS, called *equality-friendly WFS* [66] does not use the UNA. Note that both variants would be unsatisfactory for Examples 1.1 and 1.2, as they would simply leave the predicates *Odd* and *Even* undefined.

Stable model negation. In Reference [88], new acyclicity conditions (similar to weak acyclicity [47, 54] and stratification conditions are presented for existential rules with negation in rule bodies, which identify restricted classes of (possibly unguarded) rule sets that have finite and/or unique stable models, and constraints are added on the input facts to further extend these classes. The semantics for these restricted classes corresponds to ours, when we impose the same restrictions. What is studied in Reference [88] are syntactic restrictions of logic programs under the same SMS considered here, which are incomparable to ours in the sense that they do not require guardedness but make other restrictions that are not required in the present article. Our work here, in contrast, is not restricted to finite and/or unique stable models. The only syntactic restriction necessary here is guardedness.

The $\mathbb{FDNC}$ programs or the more general $\mathbb{BD}$-programs in References [51, 113] combine nonmonotonic negation and rules, allowing also the use of function symbols. They stick to the classical SMS, as we do. Decidability is obtained by restricting the structure of rules to one of seven predefined forms, which have guarded rule bodies or can be transformed to guarded rules. A new decidability proof for $\mathbb{FDNC}$ programs under the stable model semantics has been provided in Reference [27]. A further important difference to our work is that results on $\mathbb{FDNC}$ and $\mathbb{BD}$-programs are limited to *atomic* queries, whereas our results apply to answering UNBCQs. Even for atomic queries, our general results for normal guarded TGDs are not subsumed by the work in References [51, 113], as we allow predicates of arbitrary arity, and as we do not need to restrict to DL-like rules. This also applies to our results on DLs, as the functional dependencies that are required by the logic *DL-Lite*$_{\mathcal{A}}$ are not treated in References [51, 113]. In addition to that, the results in References [51, 113] do not directly apply to role inclusions of the form $Q \sqsubseteq R^-$ (or, in other notation, $Q(x, y) \rightarrow R(y, x)$), i.e., role inclusions that switch the order of variables, which shows why *DL-Lite*$_{\mathcal{A},\text{not}}$ and $\mathcal{ELHI}_{\text{not}}$ knowledge bases can only be represented via a non-trivial encoding. Some aspects of References [51, 113] are not subsumed by our work either: Most notably, the stable semantics in References [51, 113] applies to disjunctive rules, which is beyond our current setting.

The work in Reference [5] considers the classical SMS (i.e., the same we consider for guarded NTGDs) for sticky existential NTGDs, which is, just as guarded NTGDs, a sub-fragment of the general set of NTGDs. The class of sticky NTGDs is incomparable to the class of guarded NTGDs in the sense that neither contains the other. Interestingly, in Reference [5], it is shown that answering even just atomic queries under sticky NTGDs is undecidable.

In Reference [114], the author defines binary frontier-guarded programs that allow for disjunction, function symbols, and negation under the stable model semantics. Here, the syntactic

restriction to at most binary predicates and by requiring rules to be frontier-guarded ensures decidability. The recent work cited in Reference [10] presents existential nonmonotonic rules, which combine ontologies and rules via existential rules and answer set programming, using skolemization to integrate the two frameworks. The decidability of query answering in this formalism is ensured via a generalized acyclicity condition. Another more recent work [4], which we already mentioned, proposes a stable model semantics for normal existential rules via a characterization in terms of second-order logic and provides a complexity analysis of query answering for the weakly-acyclic, guarded, and sticky case. This stable model semantics coincides with the SOS semantics extensively discussed in Section 3.4, originally defined (via Kripke models) by Pearce in 1996 [98], and (via second-order logic) by Ferraris, Lee, and Lifschitz [56, 57]. One key result in Reference [4] is the undecidability of UBCQ answering with guarded TGDs under the SOS semantics. A similar result undecidability result is also shown for sticky TGDs. The paper also contains a host of other interesting results, such as for example decidable restrictions such as the restriction to weakly acyclic rules [47, 54].

Hybrid approaches. Less closely related approaches are loosely and tightly coupled dl-programs [50, 87], as well as the hybrid MKNF knowledge bases [96]. 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 **K** and **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. Less closely related are also other nonmonotonic DLs, such as those based on circumscription [24] and on rational extensions [63].

Semantic conditions. Semantic conditions that guarantee decidability were described in Reference [11–13, 35, 69, 110]. Related syntactic conditions led to the Datalog$^{\pm}$ family [33, 34]. The semantic condition relevant to the present work is the *bounded treewidth* property, which is equivalent to the *tree model property* described and used in the present article.

9 CONCLUSION

We have adopted and studied the classical **stable models semantics (SMS)** for **tuple-generating dependencies (TGDs)**. We have proved that query answering for such rules is decidable, and we have determined precise complexity results for this problem for answering covered UNBCQs under various settings (combined complexity, case of bounded arities, and data complexity), determined the data complexity for answering general UNBCQs, and given upper bounds for the combined complexity and for the bounded-arity case for general UNBCQs.

On the way to these results, we have also studied guarded disjunctive existential rules with stratified negation and investigated the complexity of query answering based on this formalism, which may be of interest in its own right.

The aspect of guardedness that we exploited most is the acyclicity of chase models, as a consequence the bounded treewidth model property. This property allowed us actually to derive a decidability result not only for UNBCQ answering, but for arbitrary first-order queries. Moreover, all our complexity results hinge on the acyclic nature of chase-based models generated by guarded rules.

We have then generalized the above results by additionally allowing for negative constraints and for keys, and shown that the above decidability and complexity results all carry over to this

more general setting. We have finally shown how the above results can be used to provide a natural and decidable stable model semantics to **description logics (DLs)**, such as $\mathcal{ELHI}$ and the *DL-Lite* family of DLs. We have also provided precise combined and data complexity results for query answering under the stable model semantics for the resulting DLs.

Our approach of adopting the stable model semantics of existential rules with negation 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 unique name assumption (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. With negation, however, this significantly differs from the first-order semantics. Reconsider Example 1.1, 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, as there would exist models in which, e.g., *mary* = *f(mary)*. The Second-Order Semantics (SOS) by Reference [56] for guarded TGDs was studied in Reference [4], where it turned out that conjunctive query answering for guarded TGDs under the SOS is undecidable.

Ongoing work includes in particular the stable model semantics for (non-stratified) guarded disjunctive TGDs with negation. The semantics for this is, again, directly inherited from the corresponding stable model semantics for disjunctive logic programming with function symbols. As this can be encoded into guarded second-order logic (GSO) in a similar way as the normal stable semantics treated here, hence query answering is decidable. By a similar encoding into GSO, we also obtain decidability for the stable model semantics for disjunctive weakly guarded and weakly frontier-guarded TGDs with negation. We are currently still exploring the precise complexity of all these problems. It would also be interesting to study "modular" variants of these problems, where logic programs can make use of generalized quantifiers [49] in rule bodies.

Finally, we plan to design and implement a query answering system for guarded NTGDs (and possible extensions), which (unlike Reference [115]) exploits the full power of alternation by using deterministic versions of our alternating algorithms that cashes possibly recurring configurations (essentially, isomorphism-classes of atoms of the chase forest with their respective types), which could be achieved in analogy to Reference [48].

APPENDICES

A PROOFS OF SOME RESULTS OF SECTION 3

In this section, we prove Theorem 3.14. We first prove the following lemma:

Lemma A.1. *Let D be a database, Σ a set of NTGDs, and $T = (D, \Sigma)$. Then, each model M of T_{UNA} contains a subset M_H of its atoms that is a model of T^M, such that M_H is isomorphic to a stable Herbrand model from $SMod(D, \Sigma^M)$.*

Proof. Let M be a model of T_{UNA}. Cearly $M \models \Sigma^M$. Consider a chase sequence $\xi : I_0 = D, I_1, I_2, \ldots$ of interpretations arising during a chase computation $chase(D, \Sigma^M, \xi)$. We show by induction on i that I_i is isomorphic via a bijection f_i to a subset $f_i(I_i)$ of M. For $i = 0$, this is trivial by letting f_0 be the identity on D. Assume that the statement holds for index j. To prove that it also for $j + 1$, we distinguish two cases.

Case (a). A non-existential TGD σ of the form $\forall \mathbf{x} \forall \mathbf{y}\ (body(\mathbf{x}, \mathbf{y}) \rightarrow R(\mathbf{x}))$ is applied to I_j with some homomorphism $h : body(\mathbf{x}, \mathbf{y}) \rightarrow I_j$ to obtain a new atom $\alpha :\ R(h(\mathbf{x}))$ to yield $I_{j+1} = I_j \cup \{\alpha\}$. By the induction hypothesis, the isomorphic rule body $f_j(body(h(\mathbf{x}), h(\mathbf{y}))) = body(f_j(h(\mathbf{x})), f_j(h(\mathbf{y})))$ is in $f_j(I_j) \subseteq M$. Because $M \models \Sigma^M$, rule σ applies to $f_j(body(h(\mathbf{x}), h(\mathbf{y}))) = body(f_j(h(\mathbf{x}), f_j(h(\mathbf{y}))))$ and generates an atom $f_j(\alpha) = R(f_j(h((x))) \in M$ isomorphic to α via f_j. Therefore, f_j

is actually also an isomorphism from $I_{j+1} \to f_j(I_{j+1}) \subseteq M$, and by letting $f_{j+1} := f_j$, the induction step is proved.

Case (b). An existential TGD σ of the form $\forall \mathbf{x} \forall \mathbf{y}\ (body(\mathbf{x},\mathbf{y}) \to \exists z\, R(\mathbf{x},z))$ is applied to I_j with some homomorphism $h : body(\mathbf{x},\mathbf{y}) \to I_j$ to obtain a new atom $\alpha : R(h'(\mathbf{x}), h'(z))$, yielding $I_{j+1} = I_j \cup \{\alpha\}$, where $h' = h \cup \{(z,u)\}$ is an extension $h' : \{body(\mathbf{x},\mathbf{y}), R(\mathbf{x},z)\} \to I_{j+1}$ such that $h'(z) = u$, where $u \in \Delta_N$ is a fresh null, not occurring in I_j. Retain for later that the rule application (σ, h) is new and was not used for the chase derivation of I_j, because in the chase every rule applies only once with a given body homomorphism. By the induction hypothesis, the isomorphic rule body $f_j(body(h(\mathbf{x}), h(\mathbf{y}))) = body(f_j(h(\mathbf{x})), f_j(h(\mathbf{y})))$ is in $f_j(I_j) \subseteq M$. Because $M \models \Sigma^M$, rule σ applies to $f_j(body(h(\mathbf{x}), h(\mathbf{y}))) = body(f_j(h(\mathbf{x}), f_j(h(\mathbf{y})))$, and there must thus be an atom $Witness_\sigma(f_j(h(\mathbf{x})), f_j(h(\mathbf{y})), v)$ and an atom $R(f_j(h((x)), v) \in M$, where $v \in \Delta_N$ is a null. This null v cannot occur in $f_j(I_j)$ because of the following: Given that, as we noted, (σ, h) is a *new* application of Σ in the chase, and f_j is an isomorphism, the application $(\sigma, f_j \circ h)$ to the body $body(f_j(h(\mathbf{x}), f_j(h(\mathbf{y})))$ must be different from all TGD applications $(\sigma^*, f_j \circ h^*)$ that generate atoms in $f_j(I_j)$. Thus, by the UNA axioms, the witness v generated by this new application $(\sigma, f_j \circ h)$ cannot coincide with any of the witnesses generated by these applications, nor with any database constant. Therefore, v is a new null value that does not appear in $f_j(I_j)$, and $R(h'(\mathbf{x}), h'(z)) = R(h'(\mathbf{x}), u)$ is isomorphic to $R(f_j(h'(\mathbf{x})), v)$. Define by $f_{j+1} = f_j \cup \{(u,v)\}$. Then f_{j+1} is an isomorphism $I_{j+1} \to f(I_{j+1}) \subseteq M$, which proves the induction step.

The union $f = \bigcup_{i \geqslant 0} f_i$ is clearly an isomorphism from $chase(D, \Sigma^M, \xi)$ to $f(chase(D, \Sigma^M, \xi)) \subseteq M$. By Proposition 3.5, $chase(D, \Sigma^M)$ is also isomorphic to the unique stable Herbrand model of $SMod(D, \Sigma^M)$, which proves the lemma. □

We are now ready to prove Theorem 3.14.

Theorem 3.14. *Let D be a database, Σ a set of NTGDs, and Q a first-order Boolean query over a joint schema $\mathcal{R}$, then*

(1) $SMod(D,\Sigma) \doteq SMod_{\text{sos}}(D,\Sigma^D_{UNA})[\mathcal{R}]$ *and*
(2) $(D,\Sigma) \models_{stable} Q$ *iff* $(D,\Sigma^D_{UNA}) \models_{\text{sos}} Q$ *iff* $SOS(D,\Sigma^D_{UNA}) \models Q$.

Proof. (1) For both directions, let $T = (D,\Sigma)$, and let $\mathbf{R}$ denote the list of predicate symbols of the schema $\mathcal{R}$ of T.

For one direction, let $M \in SMod(D,\Sigma)$. We show that then there is a model $M' \in SMod_{\text{sos}}(D,\Sigma^D_{UNA})$ such that M and $M'[\mathbf{R}]$ are isomorphic. Let $\mathcal{R}^+$ denote the schema of T_{UNA}. Let M^+ over schema $\mathcal{R}^+$ result from augmenting M with the appropriate auxiliary relations for auxiliary predicates *Witness* and *EQ*. These are clearly well-defined, as M is the result of $chase(D, (\Sigma^f)^M)$, where this chase operates (i.e., assigns witnesses) according to the UNA. The Herbrand model M^+ is then also a model of T_{UNA} and therefore clearly also of $(T_{UNA})^{M^+}$. Assume there is a smaller (general) model $M^- \subset M^+$ of $(T_{UNA})^{M^+}$. Given that M^- fulfills the UNA axioms and is a subset of Herbrand model M^+, it must be itself a Herbrand model of $(T_{UNA})^{M^+}$ smaller than M^+. Then, the restriction $M^-[\mathbf{R}]$ of M^- to the predicate symbols $\mathbf{R}$ is a model of Σ^M that is smaller than M, which contradicts our assumption that $M \in SMod(D,\Sigma)$. Therefore, the model M^+ of T_{UNA} is actually a minimal model of $(T_{UNA})^{M^+}$. By Proposition 3.10, M^+ is a model of $SOS(T_{UNA})$ and thus $M^+ \in SMod_{\text{sos}}(D,\Sigma^D_{UNA})$ and because $M = M^+[\mathbf{R}]$, $M \in SMod_{\text{sos}}(D,\Sigma^D_{UNA})[\mathcal{R}]$. Thus, $M' = M$ settles this direction with the desired isomorphism being the identity.

To see the other direction, let $M \in SMod_{\text{sos}}(D,\Sigma^D_{UNA})$. We then need to show that $M[\mathbf{R}]$ is isomorphic to some model $M' \in SMod(D,\Sigma)$. From $M \in SMod_{\text{sos}}(D,\Sigma^D_{UNA})$, by definition of this set, and by Proposition 3.10, we immediately conclude that $M \models T_{UNA}$, and M is a minimal model of $(T_{UNA})^M$. By Lemma A.1, M must contain as subset a model isomorphic to a model M_0 isomorphic to some

Herbrand model of $SMod(D, \Sigma)$. Add the appropriate $Witness$ and EQ facts from M to M_0, yielding M_0^+. M_0^+ then satisfies the TGDs of Σ and the UNA Axioms and is a model of $(T_{UNA})^M$. Given that M is a *minimal* model of $(T_{UNA})^M$, it must hold that $M = M_0^+$. Therefore, $M[\mathbf{R}] = M_0$ is isomorphic to some Herbrand model of $SMod(D, \Sigma)$, which is precisely what we had to prove.

(2) First-order Boolean queries are well-known to be *generic* (see, for example, Reference [1]), which means that they are invariant under isomorphism. Statement (2) of the theorem follows immediately from this and Statement (1). □

B UPPER BOUNDS OF THEOREM 5.9

In this appendix, we give a detailed proof of the upper bounds of Theorem 5.6 for UBCQs. This is precisely Theorem 5.9, which, for convenience, we here re-state:

Theorem 5.9 (Upper Bounds of Theorem 5.6 Restricted to UBCQs). *Given as input a database D for a schema $\mathcal{R}$, a finite stratified set Σ of guarded DNTGDs over $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma) \models_{\mathsf{strat}} Q$ is:*

(1) in 2-EXPTIME *in general;*
(2) in EXPTIME*, if* $\mathrm{ar}(\mathcal{R})$ *is bounded by a constant, and Q is acyclic;*
(3) in co-NP *in data complexity, or, if* $|\mathcal{R}|$, $\mathrm{ar}(\mathcal{R})$, *and* $\mathrm{wd}(Q)$ *are bounded by a constant.*

In Section B.1, we focus on the case of answering ground atomic queries (i.e., quantifier-free CQs with a single atom) and introduce the main concepts and constructions. Section B.2 refines some of the results of Section B.1 to obtain complexity results for the case of answering UBCQs. These complexity results almost directly lead to a proof of Theorem 5.9, as shown in Section 5.2.4.

B.1 Ground Atomic Queries

This section introduces the main concepts and constructions for the proof of Theorem 5.9. To focus on the essential issues, we restrict our attention to the special case of answering ground atomic queries with respect to stratified sets of guarded DNTGDs. More precisely, we investigate the complexity of the following decision problem: Given a database D, a finite stratified set Σ of guarded DNTGDs, and a ground atom α, decide whether $(D, \Sigma) \models_{\mathsf{strat}} \alpha$. Here, an atom $\alpha = R(\mathbf{a})$ is called *ground* (w.r.t. D) if $\mathbf{a}$ is a tuple of constants from $dom(D)$; we will often just speak of a ground atom without mentioning D, since D will typically be clear from the context. We prove the following result:

Theorem B.1. *Given as input a database D for a schema $\mathcal{R}$, a finite stratified set Σ of guarded DNTGDs over $\mathcal{R}$, and a ground atom α, deciding $(D, \Sigma) \models_{\mathsf{strat}} \alpha$ has the following complexity:*

(1) 2-EXPTIME*-complete in general;*
(2) EXPTIME*-complete, if* $\mathrm{ar}(\mathcal{R})$ *is bounded by a constant;*
(3) co-NP*-complete, if* $|\mathcal{R}|$ *and* $\mathrm{ar}(\mathcal{R})$ *are bounded by a constant.*

The lower bounds of Theorem B.1 follow from known lower bounds on answering atomic queries with respect to guarded TGDs and negation-free guarded DNTGDs. In fact, answering ground atomic queries with respect to sets of guarded TGDs is 2-EXPTIME-hard in general, and EXPTIME-hard if $\mathrm{ar}(\mathcal{R})$ is bounded by a constant [32], which implies the lower bounds in the first two points. Moreover, answering atomic queries with respect to sets of negation-free guarded DNTGDs is co-NP-complete in data complexity [3] and, in particular, if $|\mathcal{R}|$ and $\mathrm{ar}(\mathcal{R})$ are bounded by a constant; this implies the lower bound in the third point.

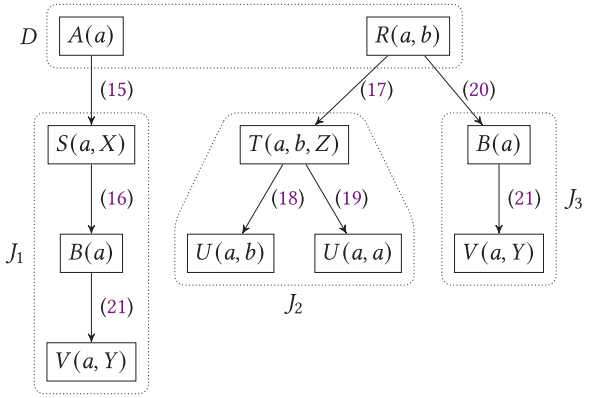


Fig. 1. Dependencies between atoms in the canonical model $\{A(a), B(a), R(a,b), S(a,X), T(a,b,Z), U(a,a), U(a,b), V(a,Y)\}$ for $D = \{A(a), R(a,b)\}$ and the set Σ consisting of the DNTGDs (15)–(22). An edge, from an atom α to an atom β indicates that β depends on α, and the edge, label indicates the DNTGD σ that was applied to obtain β based on a homomorphism that maps $guard(\sigma)$ to α.

In the following, we focus on the upper bounds of Theorem B.1. We start by giving an overview of how we obtain the upper bounds and provide a detailed proof afterwards.

B.1.1 Upper Bounds: Overview. To obtain the upper bounds of Theorem B.1, we devise an algorithm for deciding $(D, \Sigma) \not\models_{\mathsf{strat}} \alpha$. The basic idea underlying this algorithm is simple—we try to construct a canonical model J for D and Σ with $\alpha \notin J$. To this end, we start with D and exhaustively apply all applicable DNTGDs in Σ, while blocking those applications of DNTGDs that would derive α. Of course, since a canonical model for D and Σ can be of infinite size, it is impossible to always construct such a model directly. However, as we show in this section, the algorithm will be able to *verify* that such a canonical model can be constructed, implicitly constructing one if it exists.

The algorithm is based on a decomposition of canonical models, which we explain using an example. Consider the database $D = \{A(a), R(a,b)\}$ and the following stratified set Σ of guarded DNTGDs:

$$A(x) \rightarrow \exists y\, S(x,y), \tag{15}$$
$$S(x,y) \wedge U(x,x) \rightarrow B(x), \tag{16}$$
$$R(x,y) \rightarrow \exists z\, T(x,y,z), \tag{17}$$
$$T(x,y,z) \rightarrow U(x,y) \vee U(y,x), \tag{18}$$
$$T(x,y,z) \wedge A(x) \wedge U(x,y) \rightarrow U(x,x), \tag{19}$$
$$R(x,y) \wedge U(x,y) \wedge \neg U(y,x) \rightarrow B(x), \tag{20}$$
$$B(x) \rightarrow \exists y\, V(x,y), \tag{21}$$
$$V(x,y) \wedge \neg U(x,x) \rightarrow C(x). \tag{22}$$

We have $(D, \Sigma) \not\models_{\mathsf{strat}} C(a)$, as witnessed by the following canonical model for D and Σ:

$$J := \{A(a), B(a), R(a,b), S(a,X), T(a,b,Z), U(a,a), U(a,b), V(a,Y)\}. \tag{23}$$

Let us say that an atom $\beta \in J$ *depends* on an atom $\alpha \in J$ if β is obtained by a DNTGD σ in Σ with a homomorphism that maps the guard of σ to α. Thus, the atom $S(a,X)$ depends on $A(a)$, atom $B(a)$ depends on both $S(a,X)$ and $R(a,b)$, atom $T(a,b,Z)$ depends on $R(a,b)$, and so on.

Figure 1 shows the dependencies among all atoms in J; we call the forest in Figure 1 the *dependency forest* of J. Consider all atoms in J that depend on an atom in D, that is, that are obtained by an application of a DNTGD σ in Σ with a homomorphism that maps the guard of σ to D. There are only three such atoms: atom $\alpha_1 := S(a, X)$ derived using Equation (15), atom $\alpha_2 := T(a, b, Z)$ derived using Equation (17), and atom $\alpha_3 := B(a)$ derived using Equation (20). Based on these atoms, we decompose J into smaller interpretations J_0–J_3, where J_0 is the initial database D, and each J_i with $i \in \{1, 2, 3\}$ consists of all descendents of α_i in the dependency forest of J (see Figure 1):

$$J_1 := \{B(a), S(a, X), V(a, Y)\},$$
$$J_2 := \{T(a, b, Z), U(a, a), U(a, b)\},$$
$$J_3 := \{B(a), V(a, Y)\}.$$

We call J_i the *sub-model of J induced by α_i*. Now, an atom α is *not* contained in J iff, for every $i \in \{0, 1, 2, 3\}$, the atom α is not contained in J_i. One of the key insights of our analysis in Sections B.1.2 and B.1.3 is that we can recover a suitable J_i from α_i and a small amount of auxiliary information, without access to J or any of the other interpretations J_j with $j \neq i$. This leads to a *self-reduction*, essentially translating a question of the form "Can we obtain a canonical model J from D and Σ such that $\alpha \notin J$?" to simpler questions of the form "Can we obtain a canonical model J_i from α_i and Σ such that $\alpha \notin J_i$?"

Implementing this self-reduction requires much care. Due to disjunctions that may occur in the heads of DNTGDs, there is no straightforward deterministic approach of generating a canonical model J for D and Σ with $\alpha \notin J$, as in the case of guarded TGDs. Thus, *a priori* it is not clear how to compute the children $\alpha_1, \ldots, \alpha_n$ of atoms in D in the dependency forest of such a model. All we can do is to guess relevant portions of a canonical model and to verify that these indeed belong to a canonical model for D and Σ with the desired properties. Our algorithm starts by guessing a set T of all ground literals (i.e., literals ℓ with $dom(\ell) \subseteq dom(D)$) that it suspects to be true in a canonical model J for D and Σ with $\alpha \notin J$. Since α is ground, we must have $\neg\alpha \in T$. The algorithm also guesses a total order $\preceq$ on the positive literals in T that it suspects to be the order of deriving these atoms during the construction of J. Assume for the moment that the algorithm guesses correctly, that is, T is indeed the set of ground literals that are true in a canonical model J for D and Σ, and the atoms in T are derived in the specified order. Then, T provides us with valuable information. For instance, using T we can determine the list of all pairs of DNTGDs $\sigma \in \Sigma$ and homomorphisms h such that, during the construction of J, the DNTGD σ is applied with h in such a way that h maps $guard(\sigma)$ to D. Once we have access to this list, we can guess a list of all atoms $\alpha_1, \ldots, \alpha_n$ in J that are generated by applications of those DNTGDs along with the list $S_1, \ldots, S_n$ of sets S_i of atoms required to generate the sub-model J_i of J induced by α_i. Recursively, we can then verify that each atom in T is in D or is derived in some J_i, and that none of the atoms that occur negatively in T is in D or is derived in any J_i. This ensures that T is indeed the set of all ground literals that are true in some canonical model J for D and Σ. As $\neg\alpha \in T$, we also have $\alpha \notin J$.

To illustrate, let us revisit the example database $D = \{A(a), R(a, b)\}$ and set Σ consisting of DNTGDs (15)–(22). The algorithm's guess for T and $\preceq$ could be

$$T = \{A(a), B(a), R(a, b), U(a, a), U(a, b)\} \cup \{\neg\beta \mid \beta \notin J,\ dom(\beta) \subseteq \{a, b\}\} \tag{24}$$

and

$$A(a) \prec R(a, b) \prec U(a, b) \prec U(a, a) \prec B(a), \tag{25}$$

where J is the canonical model from Equation (23). The list of DNTGDs and homomorphisms would consist of DNTGD (15) with homomorphism $x \mapsto a$, DNTGD (17) with homomorphism $x, y \mapsto a, b$, and DNTGD (20) with homomorphism $x, y \mapsto a, b$. Finally, the algorithm's guess

for the atoms α_i and sets S_i could be $S(a, X)$, $T(a, b, Z)$, and $B(a)$ together with $S_1 = \{U(a, a)\}$, $S_2 = \{A(a)\}$, and $S_3 = \emptyset$. Clearly, each atom in T is in D or is derived in some J_i, and none of the atoms that occur negatively in T is in D or any of the J_i.

A number of technical issues remain to be solved. For instance, for each $i \in \{1, \ldots, n\}$, we can only verify that there is a sub-model J_i of *some* canonical model induced by α_i that contains the required ground atoms and avoids "forbidden" ones (i.e., those that occur negatively in T). *A priori* there is no guarantee that the sub-models can be combined to a canonical model for D and Σ. To ensure that the sub-models can later be combined into such a model, we have to guarantee that each DNTGD is applied at most once with each homomorphism and head atom *across different sub-models*. Thus, if a DNTGD σ is applied with homomorphism h and head atom β during the construction of a sub-model J_i, then σ cannot be applied with h and β during the construction of a different sub-model J_j with $j \neq i$. This is important to guarantee that the chase sequences for the different sub-models can be merged into chase sequences for the union of these sub-models. Unfortunately, this also means that we have to give up the property that each J_i is a model of Σ. For instance, going back to our example, we have to ensure that there is *exactly one* application of the DNTGD (21) with homomorphism $x \mapsto a$ and head atom $V(x, y)$—either in J_1 or in J_3, but not in both. But then, one of J_1 and J_3 will not contain the atom $V(a, Y)$, implying that J_1 or J_3 will not be a model of Σ. To deal with this problem, we define the notion of a *relativized canonical model*, which relaxes the notion of a canonical model and in particular allows us to enforce or prevent certain applications of DNGDs. These relativized canonical models are technically not models of Σ, but it will turn out that their combination will result in a canonical model.

The following subsections provide a detailed account of the self-reduction described here. We start by introducing the notion of a relativized canonical model in Section B.1.2, followed by a detailed description of the self-reduction in Section B.1.3. Finally, Section B.1.4 applies the self-reduction to obtain an algorithm for answering atomic queries w.r.t. stratified sets of guarded DNTGDs within the desired time bounds.

B.1.2 Relativized Canonical Models. The self-reduction outlined in Section B.1.1 crucially depends on a relaxation of the notion of a canonical model that allows us to combine "sub-models" of a canonical model J to a larger "sub-model" of J. The notion of a *relativized canonical model* that we are going to introduce in this section is such a relaxation. The idea is that a relativized canonical model is a sub-interpretation J' that is embedded into a canonical model J and is derived by an exhaustive application of DNTGDs from an initial set I of atoms in J. Atoms in J that are not listed in I may only be used as *side atoms* (i.e., non-guard atoms), unless they are derived by some DNTGD during the construction of J'. For the database $D = \{A(a), R(a, b)\}$ and the set Σ consisting of DNTGDs (15)–(22), the interpretation J given in Equation (23) and the interpretations J_1, J_2, and J_3 shown in Figure 1 are relativized canonical models (generated from D, $\{S(a, X)\}$, $\{T(a, b, Z)\}$, and $\{B(a)\}$, respectively, and embedded into J).

So, we can think of a relativized canonical model J' as being embedded into a canonical model J. Apart from the set I of start atoms, a complete specification of J' requires full information about its environment within J such as the set T of all ground literals w.r.t. I that are true in J, the set S of atoms in T that may not be derived in J' but can be used as side atoms in applications of DNTGDs, and possible requirements on applications of DNTGDs to the atoms in T (applications that must be enforced or blocked). The notion of a Σ-*environment* introduced next provides this information. The first component of a Σ-environment is a type, which corresponds to the aforementioned set T.

Definition B.2 (Type). A *type* over a schema $\mathcal{R}$ ($\mathcal{R}$-*type*, for short) is a set T of $\mathcal{R}$-literals with arguments from $\Delta \cup \Delta_N$ such that for each $\mathcal{R}$-atom α at most one of α and $\neg\alpha$ is in T. We let T^+

be the set of all positive literals in T, and set $T^- := \{\alpha \mid \neg\alpha \in T\}$. An interpretation I for $\mathcal{R}$ *realizes* T if $T^+ \subseteq I$ and $T^- \cap I = \emptyset$.

An $\mathcal{R}$-type T is *complete* if for all $\mathcal{R}$-atoms α with $dom(\alpha) \subseteq dom(T)$ we have $\alpha \in T$ or $\neg\alpha \in T$. Recall that T is *over* an interpretation I if $I \subseteq T$ and $dom(T) = dom(I)$. ◁

In addition to types T, a Σ-environment also contains restrictions on applications of DNTGDs to the atoms contained in T. To capture such restrictions, we introduce the notion of a Σ-rule.

Definition B.3 (Σ-trigger, Σ-rule). Let I be an interpretation for a schema $\mathcal{R}$, let Σ a set of guarded DNTGDs over $\mathcal{R}$, and let T be an $\mathcal{R}$-type.

- A Σ-*trigger* is a pair (σ, h) consisting of a DNTGD $\sigma \in \Sigma$ and a homomorphism h on the variables in the body of σ. A Σ-*trigger on* T is a Σ-trigger (σ, h) such that $h(body^+(\sigma)) \subseteq T^+$ and $h(body^-(\sigma)) \cap T^+ = \emptyset$.
- A Σ-*rule* is a triple $r = (\sigma, h, \beta)$ such that (σ, h) is a Σ-trigger and $\beta \in head(\sigma)$. Let h_r be the extension of h that assigns to each variable in β that does not occur in $body(\sigma)$ a fresh null[9]. We also let $\sigma_r := \sigma$ and $\beta_r := h_r(\beta)$. A Σ-*rule on* T is a Σ-rule $r = (\sigma, h, \beta)$ such that (σ, h) is a Σ-trigger on T and $\beta_r \notin T^-$.
- If R is a set of Σ-rules, then $tr(R) := \{(\sigma, h) \mid (\sigma, h, \beta) \in R\}$, and $R(\sigma, h) := \{\beta \mid (\sigma, h, \beta) \in R\}$ for each $(\sigma, h) \in tr(R)$. ◁

Note that if T is complete and (σ, h) is a Σ-trigger on T, then $h(body^+(\sigma)) \subseteq T^+$ and $h(body^-(\sigma)) \subseteq T^-$. We are now ready to define the notion of a Σ-environment.

Definition B.4 (Σ-environment). Consider an interpretation I for a schema $\mathcal{R}$, and a set Σ of guarded DNTGDs over $\mathcal{R}$ together with a stratification s of Σ. A Σ-*environment for* I is a tuple $E = (T, S, R^+, R^-, \preceq)$ consisting of an $\mathcal{R}$-type T, a set $S \subseteq T^+$, disjoint sets R^+ and R^- of Σ-rules on T, and a total order $\preceq$ on T^+ that respects s. Here, $\preceq$ *respects* s if for all $\mathcal{R}$-atoms $R(\mathbf{a}), S(\mathbf{b}) \in T^+$ with $s(R) < s(S)$ we have $R(\mathbf{a}) \prec S(\mathbf{b})$. We let $E_\downarrow := (S, R^+, R^-)$. ◁

Intuitively, a Σ-environment provides information on how a canonical model J embeds a "sub-model" J' that is generated from a set I of start atoms. The set T is the set of all ground literals w.r.t. I that are true in J, the set S provides "side atoms" from J that may not be derived in J' but can be used in its derivation, the sets R^+ and R^- contain Σ-rules that must be applied (R^+) or that are blocked (R^-), and $\preceq$ is the order in which the atoms in T^+ are derived during the construction of J. The sets R^+, R^- and the total order $\preceq$ are particularly important later in Section B.1.3 when we combine chase sequences for different "sub-models" of J to chase sequences for a larger "sub-model" of J. The sets R^+ and R^- ensure that each applicable DNTGD will be applied, but at most once for each homomorphism and head atom. The total order $\preceq$ ensures that the chase sequences can be combined in such a way that the order of deriving the atoms in T^+ is preserved.

We now turn to the definition of relativized canonical models. Like canonical models, relativized canonical models are derived by suitable chase sequences. In the case of relativized canonical models, these chase sequences work relative to Σ-environments. We will use the following terminology and notation to give a precise description of this variant of the chase:

Definition B.5. Let I, S be interpretations for a schema $\mathcal{R}$, and let Σ be a set of guarded DNTGDs over $\mathcal{R}$. A Σ-trigger (σ, h) is *active on* I *relative to* S if (σ, h) is a Σ-trigger on $I \cup S$ and $h(guard(\sigma)) \in I$. A Σ-rule $r = (\sigma, h, \beta)$ is *applicable to* I *relative to* S if (σ, h) is active on I relative to S. If r is applicable to I relative to S, then the *result of applying* r *to* I *relative to* S is the interpretation

[9] We assume that the null assigned to each existentially quantified variable z in the head of σ is uniquely determined by σ, h, and z.

$I' := I \cup \{\beta_r\}$. We write $I \vdash_r^S I'$ if r is applicable to I relative to S, and I' is the result of applying r to I relative to S. ◁

Instead of "(σ, h) is active on I relative to S," we will often say that "*σ is applicable to I with h relative to S*," or that "*σ is applicable to I relative to S*" if h is irrelevant in the given context. Similarly, instead of saying that "I' is the result of applying r to I relative to S," we will often say that "I' is the *result of applying σ to I* relative to S," specifying h, β, or both h and β as required.

Definition B.6 (Relativized Chase Sequence). Consider interpretations I, S for a schema $\mathcal{R}$, a stratified set Σ of guarded DNTGDs over $\mathcal{R}$, a semi-positive set $\Sigma_0 \subseteq \Sigma$, sets R^+ and R^- of Σ-rules, and a total order $\leqslant$ on the set of all $\mathcal{R}$-atoms.

- A *chase sequence of I with Σ_0 relative to (S, R^+, R^-)* is a potentially infinite sequence $I_0, I_1, I_2, \ldots$ of interpretations such that $I_0 = I$, each I_{i+1} is a result of applying a DNTGD in Σ_0 to I_i relative to S, and the following conditions are satisfied for each Σ_0-trigger (σ, h):
  - —if $(\sigma, h) \notin tr(R^+) \cup tr(R^-)$ and (σ, h) is active on some I_i relative to S, then there exists an atom $\beta \in head(\sigma)$ and an index $j \geqslant 0$ such that $I_j \vdash^S_{\sigma,h,\beta} I_{j+1}$;
  - —if $(\sigma, h) \in tr(R^+)$, then for all $\beta \in R^+(\sigma, h)$ there exists a $j \geqslant 0$ such that $I_j \vdash^S_{\sigma,h,\beta} I_{j+1}$;
  - —if $(\sigma, h) \in tr(R^-)$, then there are no $\beta \in R^-(\sigma, h)$ and $j \geqslant 0$ such that $I_j \vdash^S_{\sigma,h,\beta} I_{j+1}$;
  - —for each atom $\beta \in head(\sigma)$, there exists at most one index $j \geqslant 0$ with $I_j \vdash^S_{\sigma,h,\beta} I_{j+1}$.

  We call $I_0 \cup I_1 \cup I_2 \cup \cdots$ the *result* of the chase sequence. The chase sequence is *$\leqslant$-aware* if for all chase steps $I_i \vdash_r^S I_{i+1}$, either $\beta_r \in I_i \cup S$ or for all $\alpha \in h_r(body^+(\sigma_r))$ we have $\alpha < \beta_r$.
- Let $chase(I, \Sigma_0 \mid S, R^+, R^-, \leqslant)$ be the set of all interpretations that are the result of some $\leqslant$-aware chase sequence of I with Σ_0 relative to (S, R^+, R^-). ◁

Example B.7. Let Σ be the stratified set of guarded DNTGDs given by (15)–(22). Let $S = \{A(a), U(a, a)\}$ and $R = \{r\}$, where $r = (\sigma, x \mapsto a, V(x, y))$ and σ is the DNTGD (21). We assume that $h_r(y) = Y$, so $\beta_r = V(a, Y)$. Then,

$$\{S(a, X)\},\ \{S(a, X), B(a)\},\ \{S(a, X), B(a), V(a, Y)\} \tag{26}$$

is a chase sequence of $I := \{S(a, X)\}$ with Σ relative to $(S, R, \varnothing)$. The interpretation $\{S(a, X), B(a)\}$ is the result of applying DNTGD (16) to $\{S(a, X)\}$ relative to S (note that the application of Equation (16) uses the atom $U(a, a)$ as a side atom, but not as a guard). Similarly, the interpretation $\{S(a, X), B(a), V(a, Y)\}$ is the result of applying Equation (21) to $\{S(a, X), B(a)\}$ relative to S. Note that for all Σ-triggers (σ, h) that are active on $\{S(a, X), B(a), V(a, Y)\}$ relative to S, we have applied a corresponding Σ-rule. Moreover, we have satisfied the requirement that r must be applied. Notice that the chase sequence is $\leqslant$-aware for

$$S(a, X) < U(a, a) < B(a) < V(a, Y) \tag{27}$$

(it would not be $\leqslant$-aware if we swap the positions of $U(a, a)$ and $B(a)$). This shows that

$$J_1 := \{S(a, X), B(a), V(a, Y)\} \tag{28}$$

belongs to $chase(I, \Sigma \mid S, R, \varnothing, \leqslant)$.

Now observe that Equation (26) is *not* a chase sequence of I with Σ relative to $(S, \varnothing, R)$. Indeed, the last chase step that generates $\{S(a, X), B(a), V(a, Y)\}$ violates the requirement that r must not be applied. If we omit the final interpretation from Equation (26), then we obtain a chase sequence of I with Σ relative to $(S, \varnothing, R)$. This shows that

$$J_2 := \{S(a, X), B(a)\} \tag{29}$$

belongs to $chase(I, \Sigma \mid S, \varnothing, R, \leqslant)$. Notice that J_2 is not a model of Σ.

Let Σ be a set of DNTGDs over a schema $\mathcal{R}$, and let s be a stratification of Σ. Recall the definition of the sets $\Sigma_s^{(i)}$ from Section 4.1. A *chase order w.r.t. s* is a total order $\leqslant$ on the set of all $\mathcal{R}$-atoms that respects s (i.e., for all $\mathcal{R}$-atoms $R(\mathbf{a}), S(\mathbf{b})$ with $s(R) < s(S)$ we have $R(\mathbf{a}) < S(\mathbf{b})$). Using this terminology, we are now ready to define the notion of a relativized canonical model.

Definition B.8 (Relativized Canonical Model). Consider an interpretation I for a schema $\mathcal{R}$, a set Σ of guarded DNTGDs over $\mathcal{R}$, a stratification s of Σ of length l, and a Σ-environment $E = (T, S, R^+, R^-, \preceq)$ for I.

- A *canonical model for I and Σ relative to E* is an interpretation J with the property that $J \cup S$ realizes T, and there exists a chase order $\leqslant$ w.r.t. s such that $\leqslant$ extends $\preceq$ and J can be obtained by $\leqslant$-aware chase sequences of I with Σ relative to $E_\downarrow$. That is, $J = J_l$, where $J_0 = I$, and $J_i \in \mathit{chase}(J_{i-1}, \Sigma_s^{(i)} \mid E_\downarrow, \leqslant)$ for each $i \in \{1, \ldots, l\}$.
- Let $\mathit{CMod}(I, \Sigma \mid E)$ be the set of all canonical models for I and Σ relative to E. ◁

Example B.9. Let us revisit Example B.7. Let $\mathcal{R}$ be the schema of Σ. Consider the $\mathcal{R}$-type

$$T := \{A(a), B(a), S(a, X), U(a, a)\} \cup \{\neg\beta \mid \beta \notin J,\ \mathit{dom}(\beta) \subseteq \{a, X\}\},$$

where J is the interpretation from Equation (23), and the total order $\preceq$ on T^+ given by

$$A(a) \prec S(a, X) \prec U(a, a) \prec B(a).$$

Then, $E_1 := (T, S, R, \varnothing, \preceq)$ and $E_2 := (T, S, \varnothing, R, \preceq)$ are Σ-environments for I. The interpretation J_1 given by Equation (28) is a canonical model for I with Σ relative to E_1. To see this, note that $J_1 \cup S = T^+ \cup \{V(a, Y)\}$, which implies that $J_1 \cup S$ realizes T. Furthermore, the total order $\leqslant$ given by Equation (27) extends $\preceq$ (assuming $A(a) < S(a, X)$) and respects the stratification of Σ, and we have $J_1 \in \mathit{chase}(I, \Sigma \mid (E_1)_\downarrow, \leqslant)$. Similarly, the interpretation J_2 given by Equation (29) is a canonical model for I with Σ relative to E_2. It is not difficult to see that J_1 and J_2 are in fact the unique canonical models for I with Σ relative to E_1 and E_2, respectively.

Example B.10. Consider once more the stratified set Σ of guarded DNTGDs given by Equations (15)–(22). Let $E = (T, \varnothing, \varnothing, \varnothing, \preceq)$ be the Σ-environment for $D = \{A(a), R(a, b)\}$, where T and $\preceq$ are defined as in Equations (24) and (25), respectively. Then, the interpretation J given by Equation (23) belongs to $\mathit{CMod}(D, \Sigma \mid E)$.

We conclude this section by providing a reduction from the problem of answering atomic queries w.r.t. stratified sets of guarded DNTGDs to the problem of deciding the existence of canonical models relative to a particular environment. This corresponds to the first step of our algorithm for answering atomic queries w.r.t. stratified sets of guarded DNTGDs. The lemma also shows that we can always assume a complete Σ-environment. Here, we say that a Σ-environment $E = (T, S, R^+, R^-, \preceq)$ for I is *complete* if T is complete. Moreover, we say that a Σ-environment $E^* = (T^*, S^*, (R^*)^+, (R^*)^-, \preceq^*)$ for I *extends* E (or is an *extension* of E) if $T \subseteq T^*$ and $\preceq \subseteq \preceq^*$, and all other components are invariant (i.e., $S = S^*$, $R^+ = (R^*)^+$, and $R^- = (R^*)^-$). A complete Σ-environment E^* that extends a Σ-environment E is also called a *completion* of E.

Lemma B.11. *Let I be an interpretation for a schema $\mathcal{R}$, and let Σ be a stratified set of guarded DNTGDs over $\mathcal{R}$.*

(1) For every atom α that is ground w.r.t. I, we have that $(I, \Sigma) \not\models_{\mathsf{strat}} \alpha$ iff $\alpha \notin I$ and $\mathit{CMod}(I, \Sigma \mid E_{I,\alpha})$ is non-empty, where $E_{I,\alpha} := (I \cup \{\neg\alpha\}, \varnothing, \varnothing, \varnothing, \preceq)$ for an arbitrary total order $\preceq$ on I.
(2) If E is a Σ-environment for I, then $\mathit{CMod}(I, \Sigma \mid E)$ is non-empty iff there exists a completion E^* of E such that $\mathit{CMod}(I, \Sigma \mid E^*)$ is non-empty.

Proof. We first prove part 1. Fix an atom α that is ground w.r.t. I, and let s be a stratification of Σ of length l. To prove the "only if" direction, assume that $(I,\Sigma) \not\models_{\mathsf{strat}} \alpha$. Then, there exists a $J \in \mathit{CMod}(I,\Sigma)$ with $\alpha \notin J$. The latter implies $\alpha \notin I$. Since $J \in \mathit{CMod}(I,\Sigma)$, there are interpretations $J_0 := I$ and $J_i \in \mathit{chase}(J_{i-1}, \Sigma_s^{(i)})$, for $i \in \{1,\dots,l\}$, such that $J = J_l$. Let $\preceq$ be an arbitrary total order on I. It is straightforward to extend $\preceq$ to a chase order $\leqslant$ w.r.t. s that reflects the order of deriving the atoms in J by the chase sequences for the individual J_i. Then, $J_i \in \mathit{chase}(J_{i-1}, \Sigma_s^{(i)} \mid E_{I,\alpha\downarrow}, \leqslant)$ for each $i \in \{1,\dots,l\}$. Since J realizes $I \cup \{\neg\alpha\}$, this proves $J \in \mathit{CMod}(I, \Sigma \mid E_{I,\alpha})$.

For the "if" direction, assume that $\alpha \notin I$ and that $\mathit{CMod}(I,\Sigma \mid E_{I,\alpha})$ is non-empty. Let $J \in \mathit{CMod}(I,\Sigma \mid E_{I,\alpha})$. Then, J realizes $I \cup \{\neg\alpha\}$, and there exists a chase order $\leqslant$ w.r.t. s such that $\leqslant$ extends $\preceq$ and J can be obtained by $\leqslant$-aware chase sequences of I with Σ relative to $E_\downarrow$. In particular, $J = J_l$, where $J_0 = I$, and $J_i \in \mathit{chase}(J_{i-1}, \Sigma_s^{(i)})$ for each $i \in \{1,\dots,l\}$, which implies $J \in \mathit{CMod}(I,\Sigma)$. Moreover, the fact that J realizes $I \cup \{\neg\alpha\}$ implies $\alpha \notin J$, so $(I,\Sigma) \not\models_{\mathsf{strat}} \alpha$.

We now turn to part 2. Fix a Σ-environment $E = (T, S, R^+, R^-, \preceq)$ for I. The "if" direction is trivial. For the "only if" direction, let $J \in \mathit{CMod}(I,\Sigma \mid E)$. Then $J \cup S$ realizes T, and there is a chase order $\leqslant$ w.r.t. s such that $\leqslant$ extends $\preceq$ and J can be obtained by $\leqslant$-aware chase sequences of I with Σ relative to $E_\downarrow$. Let $E^* := (T^*, S, R^+, R^-, \preceq^*)$, where T^* consists of all $\mathcal{R}$-literals ℓ with $\mathit{dom}(\ell) \subseteq \mathit{dom}(I)$ and $J \cup S \models \ell$, and $\preceq^*$ is the restriction of $\leqslant$ to $(T^*)^+$. Then, E^* is a complete Σ-environment for I that extends E. Moreover, $J \cup S$ realizes T^*, $\leqslant$ extends $\preceq^*$, and J can be obtained by $\leqslant$-aware chase sequences of I with Σ relative to $E^*_\downarrow$, which implies $J \in \mathit{CMod}(I,\Sigma \mid E^*)$. □

B.1.3 The Self-reduction. This section provides a detailed description of the self-reduction outlined in Section B.1.1, which corresponds to the key computation step of our algorithm for answering atomic queries w.r.t. stratified sets of guarded DNTGDs. The input consists of an interpretation I and a Σ-environment $E = (T, S, R^+, R^-, \preceq)$ for I. The goal is to reduce the question "$\mathit{CMod}(I,\Sigma \mid E) \neq \emptyset$?" to simpler questions of the form "$\mathit{CMod}(I_i,\Sigma \mid E_i) \neq \emptyset$?," where each I_i consists of a single atom. For instance, in Example B.10, we could reduce the question "$\mathit{CMod}(D,\Sigma \mid E) \neq \emptyset$?" to three simpler questions:

(1) $\mathit{CMod}(\{S(a,X)\},\Sigma \mid E_1) \neq \emptyset$?
(2) $\mathit{CMod}(\{T(a,b,Z)\},\Sigma \mid E_2) \neq \emptyset$?
(3) $\mathit{CMod}(\{B(a)\},\Sigma \mid E_3) \neq \emptyset$?

Here, the E_i are suitable Σ-environments that describe how the interpretation J_i given in Figure 1 is embedded into the canonical model J in Equation (23). For example, E_1 could be as in Example B.9. (In general, we would also have to make sure that E_3 blocks the application of DNTGD (21) with homomorphism $x \mapsto a$, because this DNTGD will already be applied in J_1 with the same homomorphism and head atom.)

At the surface of it, the reduction is quite straightforward: We first guess all Σ-rules r on T that are applied in the underlying canonical model J and divide these into a set R_0 of Σ-rules that are applicable to I relative to S, and a set R of all remaining Σ-rules. In the above example,

$$R_0 = \{\underbrace{(\sigma_1, x \mapsto a, S(x,y))}_{r_1}, \underbrace{(\sigma_2, xy \mapsto ab, T(x,y,z))}_{r_2}, \underbrace{(\sigma_3, xy \mapsto ab, B(x))}_{r_3}\}, \tag{30}$$

$$R = \{\underbrace{(\sigma_4, x \mapsto a, V(x,y))}_{r_4}\}, \tag{31}$$

where σ_1, σ_2, σ_3, and σ_4 correspond to the DNTGDs given by Equations (15), (17), (20), and (21). The atoms β_r with $r \in R_0$ are the children of atoms in I in the dependency forest of J (cf. Section B.1.1).

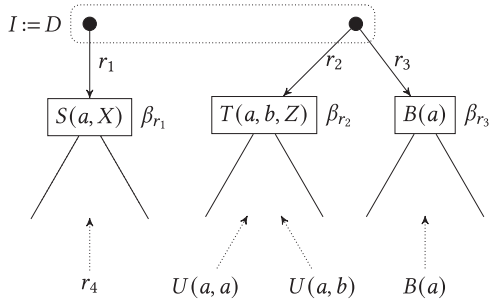


Fig. 2. Illustration of a self-reduction step using the example in Section B.1.1. We first guess all the Σ-rules r_1, r_2, r_3, r_4 that are applicable in the underlying canonical model with their guards mapped to an atom over $dom(I)$. We split these into two groups: Σ-rules r_1, r_2, r_3 whose guards are mapped to I, and the remaining Σ-rule r_4 whose guard is not mapped to I. We then assign r_4 and each of the ground atoms $U(a, a)$, $U(a, b)$, $B(b)$ that have not been derived yet to one of the Σ-rules r_1, r_2, r_3. The idea is that r_4 will be applied in the subtree rooted at β_{r_1} and that each atom assigned to r_i will be derived in the subtree rooted at β_{r_i}.

We assign each atom in T^+ that still needs to be derived (i.e., those atoms in $T^+ \setminus (I \cup S)$) and each Σ-rule in R to a Σ-rule in R_0. The idea is that an atom assigned to a Σ-rule $r \in R_0$ will eventually be derived in a relativized model J_r generated from β_r, and that a Σ-rule $r' \in R$ assigned to a Σ-rule $r \in R_0$ will be applied during the construction of J_r. Returning to the above example, we could assign r_4 to r_1, $U(a, a)$ and $U(a, b)$ to r_2, and $B(a)$ to r_3. See Figure 2 for an illustration. It now remains to ask for each $r \in R_0$ if the atoms assigned to r can indeed be generated from β_r in such a way that the Σ-rules in R that have been assigned to r are applied (and the other rules in $R_0 \cup R$ are not). This corresponds to $|R_0|$ new questions "$CMod(I_r, \Sigma \mid E_r) \neq \emptyset$?," one for each $r \in R_0$.

The following notion of a Σ-*expansion* captures all relevant information about a self-reduction step such as the sets R_0 and R of Σ-rules, the assignments of atoms in $T^+ \setminus (I \cup S)$ and Σ-rules in R to the Σ-rules in R_0, and additional technical conditions that guarantee soundness and completeness of the reduction. In the definition, we use the term Σ-*rule on* T *with guard* I to refer to a Σ-rule on T with $h_r(guard(\sigma_r)) \in I$. We also define the inverse $f^{-1}\colon Y \to 2^X$ for a mapping $f\colon X \to Y$ by $f^{-1}(y) := \{x \in X \mid f(x) = y\}$.

Definition B.12 (Σ-expansion). Let I be an interpretation for a schema $\mathcal{R}$, let Σ be a set of guarded DNTGDs over $\mathcal{R}$, let s be a stratification of Σ, and let $E = (T, S, R^+, R^-, \preceq)$ be a complete Σ-environment for I.

A Σ-*expansion* of I relative to E is a tuple $(R_0, R, \leqslant, \rho, \delta)$ consisting of a set R_0 of Σ-rules on T with guard I, a set R of Σ-rules on T with guard $T^+ \setminus I$, a total order $\leqslant$ on $T^+ \cup \{\beta_r \mid r \in R_0\}$ that extends $\preceq$ and respects s, and mappings $\rho\colon T^+ \setminus (I \cup S) \to R_0$ and $\delta\colon R \to R_0$ with the following properties:

(1) $R^+ \subseteq R_0 \cup R$, $R^- \cap (R_0 \cup R) = \emptyset$, and for each Σ-trigger (σ, h) on T with $(\sigma, h) \notin tr(R^+) \cup tr(R^-)$, there exists an atom $\beta \in head(\sigma)$ with $(\sigma, h, \beta) \in R_0 \cup R$.
(2) For each atom $\alpha \in T^+ \setminus (I \cup S)$, we have $dom(\alpha) \subseteq dom(\beta_{\rho(\alpha)})$.
(3) Let $\alpha_1, \ldots, \alpha_n$ be the atoms in $T^+ \cup \{\beta_r \mid r \in R_0\}$, sorted increasingly w.r.t. $\leqslant$. Then, for each $i \in \{1, \ldots, n\}$, we have $\alpha_i \in I$, or there exists a Σ-rule $r \in R_0$ such that either
   (a) $h_r(body^+(\sigma_r)) \subseteq \{\alpha_1, \ldots, \alpha_{i-1}\} \cup S$ and $\alpha_i = \beta_r$, or
   (b) $\alpha_i \in \rho^{-1}(r) \setminus \{\beta_r\}$. ◁

Example B.13. Consider the database D, the set Σ of guarded DNTGDs, and the Σ-environment E from Example B.10. Note that the set S of "side atoms" in E is empty. Let $I := D$, and let R_0

and R be as defined in Equations (30) and (31). We define ρ and δ so they assign the atoms in $T^+ \setminus (I \cup S) = \{U(a,a), U(a,b), B(a)\}$ and the Σ-rules in R to the Σ-rules in R_0 as indicated in Figure 2: $\rho(U(a,a)) = \rho(U(a,b)) = r_2$, $\rho(B(a)) = r_3$, and $\delta(r_4) = r_1$. We also extend the total order $\preceq$ on T^+ to the total order $\leqslant$ on $T^+ \cup \{S(a,X), T(a,b,Z), B(a)\}$ with

$$A(a) \prec R(a,b) \prec S(a,X) \prec T(a,b,Z) \prec U(a,b) \prec U(a,a) \prec B(a).$$

It is straightforward to verify that $(R_0, R, \leqslant, \rho, \delta)$ is a Σ-expansion of I relative to E.

Example B.14. Let Σ be the stratified set of guarded DNTGDs given by Equations (15)–(22). Let σ_1–σ_6 be given by DNTGDs (15) and (17)–(21). Consider the Σ-environment $E := (T, S, R^+, R^-, \preceq)$ for $I = \{T(a,b,Z)\}$, where

- $T = \{A(a), B(a), R(a,b), T(a,b,Z), U(a,a), U(a,b)\} \cup \{\neg\beta \mid \beta \notin J,\ dom(\beta) \subseteq \{a,b,Z\}\}$ with J as in (23),
- $S = \{A(a), B(a), R(a,b)\}$,
- $R^+ = \emptyset$,
- $R^- = \{(\sigma_1, x \mapsto a, S(x,y)),\ (\sigma_2, xy \mapsto ab, T(x,y,z)),\ (\sigma_5, xy \mapsto ab, B(x)),$ $(\sigma_6, x \mapsto a, V(x,y))\}$, and
- $\preceq$ is the total order on T^+ with $A(a) \prec R(a,b) \prec T(a,b,Z) \prec U(a,b) \prec U(a,a) \prec B(a)$.

Then, $(\{r_1, r_2\}, \emptyset, \preceq, \rho, \delta)$ is a Σ-expansion of I relative to E, where $r_1 = (\sigma_3, xyz \mapsto abZ, U(x,y))$, $r_2 = (\sigma_4, xyz \mapsto abZ, U(x,x))$, and ρ is such that $\rho(U(a,b)) = r_1$ and $\rho(U(a,a)) = r_2$.

The Reduction Lemma below constitutes the main result of this section. It uses the following notation: Given an $\mathcal{R}$-type T and an atom $\alpha \in T^+$, we define $T[\alpha] := \{\ell \in T \mid dom(\ell) \subseteq dom(\alpha)\}$.

Lemma B.15 (Reduction Lemma). *Let I and J be interpretations for $\mathcal{R}$, let Σ be a stratified set of guarded DNTGDs over $\mathcal{R}$, and let $E = (T, S, R^+, R^-, \preceq)$ be a complete Σ-environment for I. Then, the following are equivalent:*

(1) $J \in CMod(I, \Sigma \mid E)$.

(2) There exists a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of I relative to E with the following properties: For each $r \in R_0$, let $E_r := (T_r, S_r, R_r^+, R_r^-, \preceq_r)$, where $T_r := T[\beta_r] \cup \{\beta_r\}$, $S_r := T^+[\beta_r] \setminus \rho^{-1}(r)$, $R_r^+ := \delta^{-1}(r)$, R_r^- is the set of all Σ-rules on $T[\beta_r]$ that do not occur in R_r^+, and $\preceq_r$ is the restriction of $\leqslant_R$ to T_r^+. Then, there exist interpretations $J_r \in CMod(\{\beta_r\}, \Sigma \mid E_r)$, for $r \in R_0$, such that

$$J = I \cup \bigcup_{r \in R_0} J_r. \tag{32}$$

Moreover, $dom(I) \cap dom(J_r) \subseteq dom(\beta_r)$ and $dom(J_r) \cap dom(J_{r'}) = dom(\beta_r) \cap dom(\beta_{r'})$, if $r \neq r'$.

Proof. Throughout this proof, we fix a stratification s of Σ, and we denote the length of s by l.

(1 ⇒ 2) Let $J \in CMod(I, \Sigma \mid E)$. Then there exists a chase order $\leqslant$ w.r.t. s that extends $\preceq$, and there exist interpretations $J_0 := I$, $J_1 \in chase(J_0, \Sigma_s^{(1)} \mid E_\downarrow, \leqslant)$, …, $J_l \in chase(J_{l-1}, \Sigma_s^{(l)} \mid E_\downarrow, \leqslant)$ such that $J = J_l$ and $J \cup S$ realizes T. For each $i \in \{1, \ldots, l\}$, fix a $\leqslant$-aware chase sequence C_i of J_{i-1} with $\Sigma_s^{(i)}$ relative to $E_\downarrow$ such that the result of C_i is J_i.

Let $\hat{R}$ be the set of all Σ-rules (σ, h, β) on T such that for some $i \in \{1, \ldots, l\}$, there exists a chase step in C_i that applies σ with h and β relative to S. The set R_0 of the desired Σ-expansion consists of all triples $(\sigma, h, \beta) \in \hat{R}$ with $h(guard(\sigma)) \in I$. Similarly, the set R consists of all triples $(\sigma, h, \beta) \in \hat{R}$ with $h(guard(\sigma)) \notin I$. Note that $R^+ \subseteq \hat{R} = R_0 \cup R$ and $R^- \cap (R_0 \cup R) = R^- \cap \hat{R} = \emptyset$. Moreover, for each Σ-trigger (σ, h) on T with $(\sigma, h) \notin tr(R^+) \cup tr(R^-)$, there exists an atom $\beta \in head(\sigma)$ with $(\sigma, h, \beta) \in \hat{R} = R_0 \cup R$.

We now define the mappings $\rho\colon T^+ \setminus (I \cup S) \to R_0$ and $\delta\colon R \to R_0$ of the Σ-expansion. The construction will also yield an interpretation J_r and a set S'_r for each $r \in R_0$, and it will guarantee the following properties:

(P1) For all $r \in R_0$, we have that J_r can be obtained by $\leqslant$-aware chase sequences of $\{\beta_r\}$ with Σ relative to (S'_r, R^+_r, R^-_r).
(P2) For all atoms β generated by one of the chase sequences C_i, $i \in \{1, \ldots, l\}$, there exists an $r \in R_0$ with $\beta \in J_r \setminus S'_r$ and $dom(\beta) \subseteq dom(J_r)$. Furthermore, if $\beta \in T^+ \setminus (I \cup S)$, then $\rho(\beta)$ is such an r.
(P3) For all $r \in R_0$, we have $dom(I) \cap dom(J_r) \subseteq dom(\beta_r)$.
(P4) For all distinct $r, r' \in R_0$, we have $dom(J_r) \cap dom(J_{r'}) = dom(\beta_r) \cap dom(\beta_{r'})$.
(P5) Let $\alpha_1, \ldots, \alpha_n$ be the atoms in $T^+ \cup \{\beta_r \mid r \in R_0\}$, sorted increasingly w.r.t. $\leqslant$. Then, for each $i \in \{1, \ldots, n\}$, we have $\alpha_i \in I$, or there exists a Σ-rule $r \in R_0$ such that either
  (a) $h_r(body^+(\sigma_r)) \subseteq \{\alpha_1, \ldots, \alpha_{i-1}\} \cup S$ and $\alpha_i = \beta_r$, or
  (b) $\alpha_i \in \rho^{-1}(r) \setminus \{\beta_r\}$.

The construction proceeds in $l + 1$ stages. In stage 0, we let J_r and S_r be empty for each $r \in R_0$, and we let ρ and δ be the empty mappings. Suppose now that stage $i - 1$ has been completed for some $i \in \{1, \ldots, l\}$. Then stage i proceeds as follows: Let $r_1, r_2, \ldots$ be the sequence of Σ-rules that are applied in C_i, listed in the same order as they are applied in C_i. We consider these Σ-rules one after the other, starting with r_1. Let $r_j = (\sigma, h, \beta)$ be the current Σ-rule. We pick a Σ-rule $r \in R_0$ as follows:

Case 1: $r_j \in R_0$. In this case, we let $r := r_j$.
Case 2: $r_j \notin R_0$. In this case, there exists a Σ-rule $r \in R_0$ with $h(guard(\sigma)) \in J_r$. If $r_j \in R$, then we pick any such r. If $r_j \notin R$, then r is unique (because $dom(h(guard(\sigma)))$ contains an element that does not occur in $dom(T)$), and we pick this unique r.

Now, if $\beta_{r_j} \in T^+ \setminus (I \cup S \cup \bigcup_{r' \in R_0} J_{r'})$, then we let $\rho(\beta_{r_j}) = r$. If $r_j \in R$, then we also let $\delta(r_j) = r$. Finally, we add all atoms in $h(body^+(\sigma)) \setminus J_r$ to S'_r, and the atom β_{r_j} to J_r. This completes the construction. It is straightforward to verify that (P1)–(P5) are satisfied.

Let $\leqslant_R$ be the restriction of $\leqslant$ to $T^+ \cup \{\beta_r \mid r \in R_0\}$. Then, $(R_0, R, \leqslant_R, \rho, \delta)$ is a Σ-expansion of I relative to E: we have already argued that condition 1 of Definition B.12 is satisfied, and it is easy to see that the remaining conditions of Definition B.12 follow from (P2), (P3), and (P5).

From (P2), we obtain Equation (32), so it remains to show that for each $r \in R_0$ we have $J_r \in CMod(\{\beta_r\}, \Sigma \mid E_r)$. Let $r \in R_0$. Note that $dom(S'_r) \subseteq dom(J_r) \cap dom(I \cup S)$, which implies $dom(S'_r) \subseteq dom(\beta_r)$ due to (P3) and the assumption that $dom(S) \subseteq dom(I)$. Since T is a complete $\mathcal{R}$-type over I, this implies $S'_r \subseteq T^+[\beta_r]$. Furthermore, $S'_r \cap \rho^{-1}(r) = \emptyset$, since by (P2) we have $\alpha \notin S'_r$ for all $\alpha \in \rho^{-1}(r)$. Consequently, $S'_r \subseteq T^+[\beta_r] \setminus \rho^{-1}(r) = S_r$. By property (P1), this implies that each J_r is obtained by $\leqslant$-aware chase sequences on $\{\beta_r\}$ with Σ relative to (S_r, R^+, R^-). Since $J_r \cup S_r$ realizes $T^+[\beta_r]$, we conclude that $J_r \in CMod(\{\beta_r\}, \Sigma \mid E_r)$.

(2 ⇒ 1) Let $(R_0, R, \leqslant_R, \rho, \delta)$ be a Σ-expansion of I relative to E, and for each $r \in R_0$ let $J_r \in CMod(\{\beta_r\}, \Sigma \mid E_r)$. For all $r, r' \in R_0$, we assume that $dom(J_r) \cap dom(I) \subseteq dom(\beta_r)$, and $dom(J_r) \cap dom(J_{r'}) = dom(\beta_r) \cap dom(\beta_{r'})$ if $r \neq r'$. We show that

$$J := I \cup \bigcup_{r \in R_0} J_r$$

belongs to $CMod(I, \Sigma \mid E)$.

It is easy to see that $J \cup S$ realizes T. Indeed, we have $T^+ \subseteq J \cup S$, because for each $\alpha \in T^+ \setminus (I \cup S)$ with $\rho(\alpha) = r$, we have $\alpha \in T^+[\beta_r] \cap \rho^{-1}(r) \subseteq T^+_r \setminus S_r \subseteq J_r$ where $\alpha \in T^+[\beta_r]$ follows from

condition 2 in Definition B.12 and the last inclusion follows from the fact that $J_r \cup S_r$ realizes T_r. Furthermore, we have $T^- \cap (J \cup S) = \emptyset$. To see this, suppose, to the contrary, that there exists an atom $\alpha \in T^- \cap (J \cup S)$. If $\alpha \in I \cup S$, then since T is a complete $\mathcal{R}$-type over I and $dom(S) \subseteq dom(T)$, we have $\alpha \in T^+$, which violates that T is an $\mathcal{R}$-type. Hence, $\alpha \in J \setminus (I \cup S)$, which implies that $\alpha \in J_r$ for some $r \in R_0$. Since $\alpha \in T^-$ and $dom(T) \subseteq dom(I)$, we have $\alpha \in dom(J_r) \cap dom(I) \subseteq dom(\beta_r)$. This implies that $\alpha \in T^-[\beta_r]$ and $\alpha \in T^+[\beta_r]$ (since T is a complete $\mathcal{R}$-type over I), a contradiction.

It remains to show that there is a chase order $\leqslant$ w.r.t. s such that $\leqslant$ extends $\preceq$, and J can be obtained by $\leqslant$-aware chase sequences of I with Σ relative to $E_\downarrow$. For each $r \in R_0$, let $\leqslant_r$ be a chase order w.r.t. s such that $\leqslant_r$ extends $\preceq_r$, and J_r can be obtained by $\leqslant_r$-aware chase sequences of $\{\beta_r\}$ with Σ relative to $(E_r)_\downarrow$. Let $\leqslant$ be any chase order w.r.t. s that extends $\preceq$ and is consistent with each $\leqslant_r$ on $J_r \cup S_r$. Such a chase order exists, since the chase orders $\leqslant_r$ are consistent on T^+, and since T is a complete $\mathcal{R}$-type over I. Note that each J_r can be obtained by $\leqslant$-aware chase sequences on $\{\beta_r\}$ with Σ relative to $(E_r)_\downarrow$ (since $\leqslant$ is compatible with $\leqslant_r$ on $J_r \cup S_r$), which means that there are interpretations $J_{r,0} := \{\beta_r\}$ and $J_{r,i} \in chase(J_{r,i-1}, \Sigma_s^{(i)} \mid (E_r)_\downarrow, \leqslant)$, for $i \in \{1, \ldots, l\}$, such that $J_{r,l} = J_r$. Define

$$J_0 := I \qquad \text{and} \qquad J_i := I \cup \bigcup_{r \in R_0} J_{r,i}.$$

We show that $J_i \in chase(J_{i-1}, \Sigma_s^{(i)} \mid E_\downarrow, \leqslant)$ for each $i \in \{1, \ldots, l\}$. Since $J = I \cup \bigcup_{r \in R_0} J_r = I \cup \bigcup_{r \in R_0} J_{r,l} = J_l$, this proves that J can be obtained by $\leqslant$-aware chase sequences on I with Σ relative to $E_\downarrow$.

Let $i \in \{1, \ldots, l\}$, and let $\gamma_1, \gamma_2, \gamma_3, \ldots$ be the list of all atoms in $J_i \setminus (J_{i-1} \cup S)$ sorted increasingly w.r.t. $\leqslant$. For each $t \geqslant 0$, let $K_t := J_{i-1} \cup \{\gamma_j \mid 1 \leqslant j \leqslant t\}$. We show that $K_0, K_1, K_2, \ldots$ is a $\leqslant$-aware chase sequence of J_{i-1} with $\Sigma_s^{(i)}$ relative to $E_\downarrow$. First note that $K_0 = J_{i-1}$. Next, consider any $t \geqslant 1$. We show that K_t is the result of applying a DNTGD in $\Sigma_s^{(i)}$ to K_{t-1} relative to S.

CLAIM 1. *There exists a Σ-rule $r \in R_0$ such that either*

(1) $\sigma_r \in \Sigma_s^{(i)}$, $h_r(body^+(\sigma_r)) \subseteq K_{t-1} \cup S$ and $\gamma_t = \beta_r$; or
(2) $\gamma_t \in J_{r,i}$ and $\gamma_t \neq \beta_r$.

PROOF. Assume first that $\gamma_t \in T^+ \cup \{\beta_r \mid r \in R_0\}$. Let $\alpha_1, \ldots, \alpha_n$ be as in condition 3 of Definition B.12, and assume that $\gamma_t = \alpha_i$. Since $\gamma_t \notin I$, there exists a Σ-rule $r \in R_0$ such that either

- $h_r(body^+(\sigma_r)) \subseteq \{\alpha_1, \ldots, \alpha_{i-1}\} \cup S$ and $\alpha_i = \beta_r$, or
- $\alpha_i \in \rho^{-1}(r) \setminus \{\beta_r\}$.

In the first case, we have $\sigma_r \in \Sigma_s^{(i)}$ (because $\alpha_i \in J_i \setminus J_{i-1}$) and $\{\alpha_1, \ldots, \alpha_{i-1}\} \cup S \subseteq K_{t-1} \cup S$, which implies 1. In the second case, we have $\alpha_i \in J_{r,i}$ (because $\alpha_i \in \rho^{-1}(r)$ and $\alpha_i \in J_i$) and $\alpha_i \neq \beta_r$, which implies 2.

Now assume that $\gamma_t \notin T^+ \cup \{\beta_r \mid r \in R_0\}$. Since $\gamma_t \notin J_{i-1}$, there exists a Σ-rule $r \in R_0$ with $\gamma_t \in J_{r,i}$. Furthermore, since $\gamma_t \notin \{\beta_r \mid r \in R_0\}$, we also have $\gamma_t \neq \beta_r$. □

Let $r \in R_0$ be as guaranteed by the claim. If $\sigma_r \in \Sigma_s^{(i)}$, $h_r(body^+(\sigma_r)) \subseteq K_{t-1} \cup S$ and $\gamma_t = \beta_r$, then we can apply r to K_{t-1} relative to S, and the result of this application is K_t. In the following, we assume that $\gamma_t \in J_{r,i}$ and $\gamma_t \neq \beta_r$. Let C be a $\leqslant$-aware chase sequence of J_{i-1} with $\Sigma_s^{(i)}$ relative to $(E_r)_\downarrow$. Since $\gamma_t \in J_{r,i} \setminus J_{i-1}$ and $\gamma_t \neq \beta_r$, there exists a chase step in C that generates γ_t. Consider the first such chase step in C, and assume that it applies a DNTGD σ with a homomorphism h and an atom β relative to S_r. Note that $S_r \subseteq S \cup \bigcup_{r' \in R_0 \setminus \{r\}} J_{r'}$. Since C is $\leqslant$-aware and $\gamma_t \notin I \cup S$, we

have $\alpha < \gamma_t$ and hence $\alpha \in K_{t-1} \cup S$ for each $\alpha \in h(body^+(\sigma))$. It follows that we can apply σ to K_{t-1} with h and β relative to S, and that the result of this application is K_t.

Note that there might be DNTGDs $\sigma \in \Sigma_s^{(i)}$ and homomorphisms h such that σ was not applied with h in the sequence $K_0, K_1, K_2, \ldots$. These correspond to applications of DNTGDs with homomorphisms that generate one of the atoms in $J_i \cup S$, thus we can safely apply any such DNTGD σ with any such homomorphism h as soon as all atoms in $h(body^+(\sigma))$ have been generated. The properties of a Σ-expansion and of each J_r make sure that the resulting chase sequence is $\leqslant$-aware of J_{i-1} with $\Sigma_s^{(i)}$ relative to $E_\downarrow$. □

B.1.4 Proof of Theorem B.1. We now combine the results from the previous two subsections to obtain a proof of the upper bounds in Theorem B.1. To this end, we describe an algorithm that decides $(D, \Sigma) \not\models_{\text{strat}} \alpha$ within the desired time bounds. The algorithm uses an algorithm for the following problem as a subroutine:

ACHECK	
Input:	a stratified set Σ of guarded DNTGDs over a schema $\mathcal{R}$, an $\mathcal{R}$-atom β, and a Σ-environment E for $\{\beta\}$
Question:	$CMod(\{\beta\}, \Sigma \mid E) \neq \emptyset$?

Lemma B.11 (2) and Lemma B.15 (the Reduction Lemma) lead to the following:

Theorem B.16. ACHECK *is in* 2-EXPTIME. *It is in* EXPTIME *if* $\text{ar}(\mathcal{R})$ *is bounded by a constant, and in* PTIME *if* $|\mathcal{R}|$ *and* $\text{ar}(\mathcal{R})$ *are bounded by a constant.*

Proof. We describe an alternating Turing machine for ACHECK. To this end, we use the following family of mappings to canonize objects such as atoms or Σ-environments that involve elements from $\Delta \cup \Delta_N$: Given an atom $\alpha = R(a_1, \ldots, a_n)$, let $can_\alpha : \{a_1, \ldots, a_n\} \to \{1, \ldots, n\}$ be defined by $can_\alpha(a) := \min\{i \leqslant n \mid a_i = a\}$. We extend can_α in the obvious way to objects built from the constants and nulls in $\{a_1, \ldots, a_n\}$. For atoms $\beta = S(b_1, \ldots, b_m)$ with $dom(\beta) \subseteq \{a_1, \ldots, a_n\}$, we set $can_\alpha(\beta) := S(can_\alpha(b_1), \ldots, can_\alpha(b_m))$ and $can_\alpha(\neg\beta) := \neg can_\alpha(\beta)$, and for sets L of literals with $dom(L) \subseteq \{a_1, \ldots, a_n\}$, we set $can_\alpha(L) := \{can_\alpha(\ell) \mid \ell \in L\}$. If $r = (\sigma, h, \beta)$ is a Σ-rule such that the range of h is in $\{a_1, \ldots, a_n\}$, then we let $can_\alpha(r) := (\sigma, can_\alpha \circ h, \beta)$. And given a Σ-environment $E = (T, S, R^+, R^-, \preceq)$ with $dom(T) \subseteq \{a_1, \ldots, a_n\}$, we define $can_\alpha(E) := (can_\alpha(T), can_\alpha(S), can_\alpha(R^+), can_\alpha(R^-), can_\alpha(\preceq))$, where $can_\alpha(R^*) := \{can_\alpha(r) \mid r \in R^*\}$ for $* \in \{+, -\}$, and $can_\alpha(\preceq) := \{(can_\alpha(\beta), can_\alpha(\gamma)) \mid (\beta, \gamma) \in \preceq\}$.

We use configurations (β, E, c), where β is an atom, $E = (T, S, R^+, R^-, \preceq)$ is a Σ-environment for $\{\beta\}$, and c assigns to each atom $\alpha \in T^+ \setminus (\{\beta\} \cup S)$ (respectively, to each Σ-rule $r \in R^+$) a counter $c(\alpha) \in \{0, 1, \ldots, d^*\}$ (respectively, $c(r) \in \{0, 1, \ldots, d^*\}$), for a constant d^* as in Section 5.2.2, which will also be specified specified later in this proof. The goal in such a configuration is to verify that $CMod(\{\beta\}, \Sigma \mid E)$ is non-empty. Intuitively, the role of the counter is to prevent the machine from postponing the derivation of an atom in $T^+ \setminus (\{\beta\} \cup S)$ or the application of a Σ-rule in R^+ indefinitely. To keep the space bounded, we store β and E in canonized form, which means that $\beta = can_\beta(\beta)$ and $E = can_\beta(E)$. Whenever we generate a new configuration (β', E', c'), we first apply $can_{\beta'}$ to both β' and E' to canonize the configuration. It is straightforward to verify that $CMod(\{\beta'\}, \Sigma \mid E')$ is non-empty iff $CMod(\{can_{\beta'}(\beta')\}, \Sigma \mid can_{\beta'}(E'))$ is non-empty. We define

$$d^* := |\{(can_\beta(\beta), can_\beta(E)) \mid \beta \text{ is an } \mathcal{R}\text{-atom, and } E \text{ is a } \Sigma\text{-environment for } \{\beta\}\}|,$$

where $\mathcal{R}$ refers to the schema of the given set Σ. (A concrete value for d^* was already stated in Section 5.2.2.)

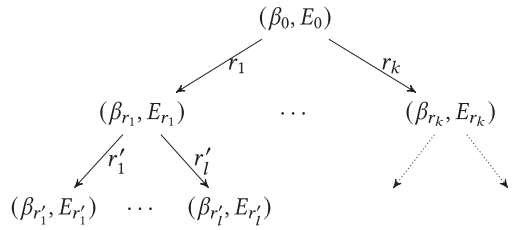


Fig. 3. Construction of the self-reduction tree for $J_0 \in \mathit{CMod}(\{\beta_0\}, \Sigma \mid E_0)$. The root is labeled by (β_0, E_0). The children of a node (β, E) correspond to the atoms β_r, Σ-environments E_r, and interpretations $J_r \in \mathit{CMod}(\{\beta_r\}, \Sigma \mid E_r)$ guaranteed by the Reduction Lemma (Lemma B.15).

Now, given as input a stratified set Σ of guarded DNTGDs, an atom β_0, and a Σ-environment $E_0 = (T, S, R^+, R^-, \preceq)$ for $\{\beta_0\}$, the machine starts in configuration (β_0, E_0, c_0), where c_0 assigns the integer d^* to all atoms in $T^+ \setminus (\{\beta_0\} \cup S)$ and to all Σ-rules in R^+. If the current configuration is (β, E, c) with $E = (T, S, R^+, R^-, \preceq)$, then it proceeds as follows:

(1) If $c(\alpha) = 0$ for some $\alpha \in T^+ \setminus (\{\beta\} \cup S)$, or if $c(r) = 0$ for some $r \in R^+$, then reject.
(2) Guess a completion $E^* = (T^*, S, R^+, R^-, \preceq^*)$ of E according to Lemma B.11 (2). Extend c to $(T^*)^+ \setminus (\{\beta\} \cup S)$ by setting $c(\alpha) := d^*$ for each $\alpha \in (T^*)^+ \setminus (\{\beta\} \cup S)$ with $\alpha \notin T^+$.
(3) Guess a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of $\{\beta\}$ relative to E^* according to Lemma B.15. If no such Σ-expansion exists, then reject.
(4) For each $r \in R_0$, do the following:
   (a) Compute $E_r = (T_r, S_r, R_r^+, R_r^-, \preceq_r)$ as defined in Lemma B.15.
   (b) For each $\alpha \in T_r^+ \setminus (\{\beta\} \cup S_r)$, let $c_r(\alpha) := c(\alpha) - 1$. For each $r' \in R_r^+$, let $c_r(r') := c(r') - 1$ if $r' \in R^+$, and $c_r(r') := d^*$ if $r' \notin R^+$.
(5) Accept iff for each $r \in R_0$ the computation starting in configuration (β_r, E_r, c_r) is accepting.

It is straightforward to check that the space required by the machine is exponential in general, polynomial if $\mathrm{ar}(\mathcal{R})$ is bounded, and logarithmic if $\mathrm{ar}(\mathcal{R})$ and $|\mathcal{R}|$ are bounded. This implies the desired time bounds.

It remains to prove that the machine accepts the input Σ, β_0, and E_0 iff $\mathit{CMod}(\{\beta_0\}, \Sigma \mid E_0)$ is non-empty.

"Only If" From Lemma B.11 (2) and Lemma B.15, it is clear that if the input is accepted, then $\mathit{CMod}(\{\beta_0\}, \Sigma \mid E_0)$ is non-empty. The counters guarantee that for each configuration (β, E) with $E = (T, S, R^+, R^-, \preceq)$, we derive all atoms $\alpha \in T^+ \setminus (\{\beta\} \cup S)$ and apply all Σ-rules $r \in R^+$ within at most d^* steps, where a step refers to a transition between two individual configurations.

"If" The proof of this direction is based on the concept of a *self-reduction tree*. Let β be an $\mathcal{R}$-atom, $E = (T, S, R^+, R^-, \preceq)$ a Σ-environment, and $J \in \mathit{CMod}(\{\beta\}, \Sigma \mid E)$. By induction, we define the self-reduction tree $\mathit{SR}_J(\beta, E)$ for J as follows: The root of $\mathit{SR}_J(\beta, E)$ is labeled by (β, E). We assume that E is complete; otherwise, we replace it by a completion E^* of E with $J \in \mathit{CMod}(\{\beta\}, \Sigma \mid E^*)$, which exists by Lemma B.11 (2). By Lemma B.15, there is a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of $\{\beta\}$ relative to E such that J is the union of $\{\beta\}$ and interpretations $J_r \in \mathit{CMod}(\{\beta_r\}, \Sigma \mid E_r)$, where r ranges over the Σ-rules in R_0. For each $r \in R_0$, we insert $\mathit{SR}_{J_r}(\beta_r, E_r)$ into $\mathit{SR}_J(\beta, E)$, and add an r-labeled edge, from the root of $\mathit{SR}_J(\beta, E)$ to the root of $\mathit{SR}_{J_r}(\beta_r, E_r)$. See Figure 3 for an illustration. To simplify the presentation, in the following we will often identify a node with its label:

Let $u = (\beta_u, E_u)$ be a node in $\mathit{SR}_J(\beta, E)$, where $E_u = (T_u, S_u, R_u^+, R_u^-, \preceq_u)$. Note that for each atom $\alpha \in T_u^+ \setminus (\{\beta_u\} \cup S_u)$, there exists a unique path $u_0, u_1, \ldots, u_n$ from $u_0 = u$ to a descendent

$u_n = (\beta', E')$ such that $\beta' = \alpha$, and for each $i \in \{1, \ldots, n-1\}$, we have $u_i = (\beta_i, E_i)$, $E_i = (T_i^+, S_i, R_i^+, R_i^-, \preceq_i)$, and $\alpha \in T_i^+ \setminus (\{\beta_i\} \cup S_i)$. We call this the *witness path for α at u*. Similarly, for each Σ-rule $r \in R^+$, there exists a unique path $u_0, u_1, \ldots, u_n$ from $u_0 = u$ to a descendent $u_n = (\beta', E')$ such that the edge, from u_{n-1} to u_n is labeled r, and for each $i \in \{1, \ldots, n-1\}$, we have $u_i = (\beta_i, E_i)$, $E_i = (T_i^+, S_i, R_i^+, R_i^-, \preceq_i)$, and $r \in R_i^+$. We call this the *witness path for r at u*. For each $\alpha \in T_u^+ \setminus (\{\beta_u\} \cup S_u)$ and each $r \in R_u^+$, let $d_J(u, \alpha)$ be the length of the witness path for α at u, and let $d_J(u, r)$ be the length of the witness path for r at u. Let $d_J(u)$ be the maximum of $d_J(u, \alpha)$ and $d_J(u, r)$, where α ranges over the atoms in $T_u^+ \setminus (\{\beta_u\} \cup S_u)$ and r ranges over the Σ-rules in R_u^+.

We show that if $CMod(\{\beta\}, \Sigma \mid E)$ is non-empty, then there always exists a $J \in CMod(\{\beta\}, \Sigma \mid E)$ such that $d_J(u) \leqslant d^*$ for all nodes u in $SR_J(\beta, E)$. This implies the "if" direction.

Suppose that $CMod(\{\beta\}, \Sigma \mid E)$ is non-empty. We construct the desired interpretation J by induction. For the base case, we pick an arbitrary $J_0 \in CMod(\{\beta\}, \Sigma \mid E)$. In the induction step, we are given $J_i \in CMod(\{\beta\}, \Sigma \mid E)$. If $d_{J_i}(u) \leqslant d^*$ for all nodes u in $SR_{J_i}(\beta, E)$, then J_i is the desired interpretation J, and we are done. Otherwise, we construct J_{i+1} as follows: Let $u = (\beta_u, E_u)$ be a node at minimal depth in $SR_{J_i}(\beta, E)$ such that $d_{J_i}(u) > d^*$. Assume $E_u = (T_u, S_u, R_u^+, R_u^-, \preceq_u)$. We apply the following reduction steps:

Reduction Step 1: Suppose that there exists an $\alpha \in T_u^+ \setminus (\{\beta_u\} \cup S_u)$ with $d_{J_i}(u, \alpha) > d^*$. Let $u_0, u_1, \ldots, u_n$ with $n = d_{J_i}(u, \alpha)$ be the witness path for α at u. Given $u_p = (\beta_p, E_p)$ and $u_q = (\beta_q, E_q)$, we write $u_p \sim u_q$ if there exists a bijective mapping $f\colon dom(\beta_p) \to dom(\beta_q)$ such that

- $f(a) = a$ for all $a \in dom(\beta_u) \cap dom(\beta_{u_p})$,
- $f^{-1}(a) = a$ for all $a \in dom(\beta_u) \cap dom(\beta_{u_q})$, and
- $f(\beta_p) = \beta_q$ and $f(E_p) = E_q$, where $f(E_p)$ is obtained from E_p by replacing in E_p each occurrence of an element $a \in dom(\beta_{u_p})$ by $f(a)$.

Intuitively, $u_p \sim u_q$ means that u_p and u_q are "indistinguishable" from the perspective of u. Now, by our choice of d^*, there exist $p < q$ such that $u_p \sim u_q$. Replace the subtree rooted at u_p by an isomorphic copy of the subtree rooted at u_q. This yields the self-reduction tree $SR_{J'}(\beta, E)$ for a new interpretation $J' \in CMod(\{\beta\}, \Sigma \mid E)$, with a shorter witness path for α at u.

Reduction Step 2: Suppose that there exists an $r \in R_u^+$ with $d_{J_i}(u, r) > d^*$. Let $u_0, u_1, \ldots, u_n$ with $n = d_{J_i}(u, \alpha)$ be the witness path for r at u. By our choice of d^*, there exist $p < q$ such that $u_p \sim u_q$. As in reduction step 1, we replace the subtree rooted at u_p by an isomorphic copy of the subtree rooted at u_q, which yields the self-reduction tree $SR_{J'}(\beta, E)$ for a new interpretation $J' \in CMod(\{\beta\}, \Sigma \mid E)$, with a shorter witness path for r at u.

We stress that none of these reduction steps increases the length of a witness path for any atom $\alpha' \in T_u^+ \setminus (\{\beta_u\} \cup S_u)$ or any Σ-rule $r' \in R_u^+$. This could only happen if the witness path for α' (respectively, r') leads through u_p but not through u_q. However, $u_p \sim u_q$ and $dom(\alpha') \subseteq dom(\beta_u)$ (respectively, r' is a Σ-rule on T_u), so α' (respectively, r') would be contained in both $T_{u_p}^+ \setminus (\{\beta_{u_p}\} \cup S_{u_p})$ and $T_{u_q}^+ \setminus (\{\beta_{u_q}\} \cup S_{u_q})$ (respectively, $R_{u_p}^+$ and $R_{u_q}^+$), which means that the witness path for α' (respectively, r') must visit u_q and thus yields a contradiction. By the same reasoning, we have $d_{J'}(v) \leqslant d_{J_i}(v)$ for all nodes v that are not descendents of u.

Exhaustive application of the above reduction steps leads to an interpretation J_{i+1} with $J_{i+1} \in CMod(\{\beta\}, \Sigma \mid E)$ and $d_{J_{i+1}}(u) \leqslant d^*$. Note that $SR_{J_i}(\beta, E)$ and $SR_{J_{i+1}}(\beta, E)$ are identical, except for the subtrees rooted at u. Furthermore, $d_{J_{i+1}}(v) \leqslant d_{J_i}(v)$ for all nodes v that are not descendents of u. This guarantees that the sequence $J_0, J_1, J_2, \ldots$ converges to an interpretation $J \in CMod(\{\beta\}, \Sigma \mid E)$ such that $d_J(v) \leqslant d^*$ for all nodes v of $SR_J(\beta, E)$. □

We are now ready to give a proof of Theorem B.1.

Proof of Theorem B.1. The lower bounds follow from known lower bounds on answering atomic queries with respect to guarded TGDs and negation-free guarded DNTGDs, as explained at the beginning of Section B.1. It remains to prove the upper bounds. To this end, we describe a Turing machine that checks $(D, \Sigma) \not\models_{\mathsf{strat}} \alpha$ within the desired time bounds.

Given D, Σ, and α, the machine first computes $E_{D,\alpha}$ as defined in Lemma B.11 (1). It then guesses a completion E^* of $E_{D,\alpha}$ according to Lemma B.11 (2), and a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of D relative to E^* according to Lemma B.15. In the general case and in the case that $\mathrm{ar}(\mathcal{R})$ is bounded by a constant, we replace the non-deterministic guessing steps by loops over all possible completions and Σ-expansions. The machine accepts the input iff $\mathit{CMod}(\{\beta_r\}, \Sigma \mid E_r) \neq \emptyset$ for each $r \in R_0$, where E_r is defined as in Lemma B.15. From Lemma B.11 and Lemma B.15, it is clear that this will be the case iff $(D, \Sigma) \not\models_{\mathsf{strat}} \alpha$.

Theorem B.16 implies that $\mathit{CMod}(\{\beta_r\}, \Sigma \mid E_r) \neq \emptyset$ can be checked in 2-EXPTIME in the general case, in EXPTIME if $\mathrm{ar}(\mathcal{R})$ is bounded, and in PTIME if both $|\mathcal{R}|$ and $\mathrm{ar}(\mathcal{R})$ are bounded. This leads to the upper bounds stated in Theorem B.1. □

B.2 UCBQs

We now generalize the results of Section B.1 to prove that the complexity bounds stated in Theorem 5.9 also hold in the case of UBCQs. Most of the heavy lifting for the proof has been done already in Section B.1, and all that remains is to generalize some of the concepts and results of Section B.1.

We start with some basic notation and concepts regarding BCQs. Let Q be a BCQ. By $\mathit{var}(Q)$, we denote the set of all variables that occur in Q. A *partial match* of Q in an interpretation I is a partial mapping $h\colon \mathit{var}(Q) \to \mathit{dom}(I)$ such that for all atoms α in Q we have $h(\alpha) \in I$ if h is defined on all variables in α. Given a partial match h of Q in I, let $\mathit{dom}(h)$ be the set of all variables $x \in \mathit{var}(Q)$ such that $h(x)$ is defined. We will occasionally identify a partial match h of Q in I with the (total) mapping $h\colon \mathit{dom}(h) \to \mathit{dom}(I)$. Given $V \subseteq \mathit{var}(Q)$, we let $Q[V]$ be the subquery of Q induced by all atoms of Q that contain at least one variable from V. Note that $Q[V]$ is a BCQ if V is non-empty.

Let h be a partial match of Q in I, let Σ be a stratified set of guarded DNTGDs, and let R_0 be a set of Σ-rules. A *decomposition* for (Q, h) over R_0 is a tuple $(V_r)_{r \in R_0}$ such that $\bigcup_{r \in R_0} V_r = \mathit{var}(Q) \setminus \mathit{dom}(h)$, $V_r \cap V_{r'} = \emptyset$ for all distinct $r, r' \in R_0$, there is no atom in Q that contains a variable from V_r and a variable from $V_{r'}$ for distinct $r, r' \in R_0$, and for each $r \in R_0$ the range of $h|_{\mathit{var}(Q[V_r])}$ is in $\mathit{dom}(\beta_r)$.

Next, we generalize the concept of a Σ-environment. Let $\hat{Q}$ be a UBCQ. A $(\Sigma, \hat{Q})$-*environment* for I is a pair $(E, \mathcal{Q})$ consisting of a Σ-environment $E = (T, S, R^+, R^-, \preceq)$ for I and a set $\mathcal{Q}$ of pairs (Q, h), where Q is a subquery of a disjunct of $\hat{Q}$ and h is a partial match of Q in T^+. Intuitively, the additional component $\mathcal{Q}$ provides information about partial matches of some of the disjuncts of $\hat{Q}$ into an underlying canonical model J. If there is a partial match h' of some disjunct of $\hat{Q}$ in J, then the component Q in $(Q, h) \in \mathcal{Q}$ corresponds to the subquery of that disjunct induced by all atoms that are mapped into the "submodel" of J generated from I, and h is the restriction of h' to the variables in Q. A $(\Sigma, \hat{Q})$-environment $(E, \mathcal{Q})$ is *complete* if E is complete and $\mathcal{Q}$ consists of all pairs (Q, h) such that Q is a subquery of a disjunct of $\hat{Q}$ and h is a partial match of Q in T^+. A $(\Sigma, \hat{Q})$-environment $(E^*, \mathcal{Q}^*)$ *extends* a $(\Sigma, \hat{Q})$-environment $(E, \mathcal{Q})$ if E^* extends E and $\mathcal{Q} \subseteq \mathcal{Q}^*$. A complete $(\Sigma, \hat{Q})$-environment $(E^*, \mathcal{Q}^*)$ that extends a $(\Sigma, \hat{Q})$-environment $(E, \mathcal{Q})$ is also called a *completion* of $(E, \mathcal{Q})$.

The last ingredient to our proof of the complexity bounds for answering UBCQs are the following generalizations of Lemma B.11 and Lemma B.15: These lemmas use $\mathit{CMod}(I, \Sigma \mid (E, \mathcal{Q}))$ to

denote the set of all $J \in \mathit{CMod}(I, \Sigma \mid E)$ such that for all $(Q, h) \in \mathbf{Q}$ there is no match of Q in J that extends h.

Lemma B.17. *Let I be an interpretation for a schema $\mathcal{R}$, let Σ be a stratified set of guarded DNTGDs over $\mathcal{R}$, and let $\hat{Q}$ be a UBCQ over $\mathcal{R}$.*

(1) We have $(I, \Sigma) \not\models_{\mathsf{strat}} \hat{Q}$ iff $\mathit{CMod}(I, \Sigma \mid (E_I, \mathbf{Q}_{I,\hat{Q}}))$ is non-empty, where $E_I := (I, \varnothing, \varnothing, \varnothing, \preceq)$ for an arbitrary total order $\preceq$ on I, and $\mathbf{Q}_{I,\hat{Q}}$ is the set of all pairs (Q, h) with Q a disjunct of $\hat{Q}$ and h the empty mapping.

(2) If $(E, \mathbf{Q})$ is a $(\Sigma, \hat{Q})$-environment for I, then $\mathit{CMod}(I, \Sigma \mid (E, \mathbf{Q}))$ is non-empty iff there exists a completion $(E^*, \mathbf{Q}^*)$ of $(E, \mathbf{Q})$ such that $\mathit{CMod}(I, \Sigma \mid (E^*, \mathbf{Q}^*))$ is non-empty.

Proof. Point 2 is a straightforward generalization of Lemma B.11 (2). For point 1, if $(I, \Sigma) \not\models_{\mathsf{strat}} \hat{Q}$, then there is an interpretation $J \in \mathit{CMod}(I, \Sigma)$ with $J \not\models \hat{Q}$. The proof that $J \in \mathit{CMod}(I, \Sigma \mid E_I)$ is similar to the proof of the first part of Lemma B.11 (1). Let $(Q, h) \in \mathbf{Q}_{I,\hat{Q}}$. Then, since $J \not\models \hat{Q}$, there is no match of Q in J (that extends the empty mapping h). This implies $J \in \mathit{CMod}(I, \Sigma \mid (E_I, \mathbf{Q}_{I,\hat{Q}}))$.

For the converse, let $J \in \mathit{CMod}(I, \Sigma \mid (E_I, \mathbf{Q}_{I,\hat{Q}}))$. Then, $J \in \mathit{CMod}(I, \Sigma)$. Since for every $(Q, h) \in \mathbf{Q}_{I,\hat{Q}}$, there is no match of Q in J that extends the empty mapping h, we have $J \models \hat{Q}$. □

Lemma B.18 (Reduction Lemma for UBCQs). *Let I be an interpretation for $\mathcal{R}$, let Σ be a stratified set of guarded DNTGDs over $\mathcal{R}$, let $\hat{Q}$ be a UBCQ over $\mathcal{R}$, and let $(E, \mathbf{Q})$ be a complete $(\Sigma, \hat{Q})$-environment for I, where $E = (T, S, R^+, R^-, \preceq)$. Then, the following are equivalent:*

(1) $\mathit{CMod}(I, \Sigma \mid (E, \mathbf{Q}))$ is non-empty.

(2) There exists a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of I relative to E and $(\Sigma, \hat{Q})$-environments $(E_r, \mathbf{Q}_r)$ for $r \in R_0$, where E_r is defined as in Lemma B.15, such that the following are true:

(a) For each $r \in R_0$, the set $\mathit{CMod}(\{\beta_r\}, \Sigma \mid (E_r, \mathbf{Q}_r))$ is non-empty.

(b) For each pair $(Q, h) \in \mathbf{Q}$ and each decomposition $(V_r)_{r \in R_0}$ for (Q, h) over R_0, there exists an $r \in R_0$ such that $(Q[V_r], h|_{\mathit{var}(Q[V_r])}) \in \mathbf{Q}_r$.

Proof. *(1 ⇒ 2)* Let $J \in \mathit{CMod}(I, \Sigma \mid (E, \mathbf{Q}))$. Then, $J \in \mathit{CMod}(I, \Sigma \mid E)$, so by Lemma B.15, there exists a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of I relative to E and interpretations $J_r \in \mathit{CMod}(\{\beta_r\}, \Sigma \mid E_r)$, for $r \in R_0$, such that $J = I \cup \bigcup_{r \in R_0} J_r$. We now show that there are sets $\mathbf{Q}_r$ for $r \in R_0$ such that

- $J_r \in \mathit{CMod}(\{\beta_r\}, \Sigma \mid (E_r, \mathbf{Q}_r))$ for each $r \in R_0$, and
- condition (b) is satisfied.

It suffices to show that for all $(Q, h) \in \mathbf{Q}$ and all decompositions $(V_r)_{r \in R_0}$ for (Q, h) over R_0, there exists an $r \in R_0$ such that the mapping $h|_{\mathit{var}(Q[V_r])}$ cannot be extended to a match of $Q[V_r]$ in J_r. Consider a pair $(Q, h) \in \mathbf{Q}$ and a decomposition $(V_r)_{r \in R_0}$ for (Q, h) over R_0. For a contradiction, suppose that for all $r \in R_0$ the mapping $h|_{\mathit{var}(Q[V_r])}$ can be extended to a match h_r of $Q[V_r]$ in J_r. Since h and the mappings h_r agree on all common variables, we can combine these mappings into a match h' of Q in J. However, since $h' \supseteq h$, this contradicts the fact that $J \in \mathit{CMod}(I, \Sigma \mid (E, \mathbf{Q}))$.

(2 ⇒ 1) Let $(R_0, R, \leqslant_R, \rho, \delta)$ be a Σ-expansion of I relative to E. For each $r \in R_0$, let $(E_r, \mathbf{Q}_r)$ be a $(\Sigma, \hat{Q})$-environment such that E_r is defined as in Lemma B.15 and both (a) and (b) are satisfied. For each $r \in R_0$, let $J_r \in \mathit{CMod}(\{\beta_r\}, \Sigma \mid (E_r, \mathbf{Q}_r))$. Without loss of generality, we may assume that for all $r, r' \in R_0$ we have $\mathit{dom}(I) \cap \mathit{dom}(J_r) \subseteq \mathit{dom}(\beta_r)$, and $\mathit{dom}(J_r) \cap \mathit{dom}(J_{r'}) = \mathit{dom}(\beta_r) \cap \mathit{dom}(\beta_{r'})$ if $r \neq r'$, which can be achieved by consistently replacing each null in $\mathit{dom}(J_r) \setminus \mathit{dom}(I)$ by a fresh one. By Lemma B.15,

$$J := I \cup \bigcup_{r \in R_0} J_r$$

belongs to $CMod(I, \Sigma \mid E)$. To show that $J \in CMod(I, \Sigma \mid (E, \mathcal{Q}))$, it suffices to show that for all $(Q, h) \in \mathcal{Q}$ there is no match h' of Q in J that extends h.

For a contradiction, assume that there exists a pair $(Q, h) \in \mathcal{Q}$ and a match h' of Q in J that extends h. Since $(E, \mathcal{Q})$ is complete, we may assume that h is defined on all variables $x \in var(Q)$ with $h'(x) \in dom(T^+)$. For each $r \in R_0$, let $V_r := \{x \in var(Q) \mid h'(x) \in dom(J_r) \setminus dom(T^+)\}$. Then, $(V_r)_{r \in R_0}$ is a decomposition for (Q, h) over R_0. Now, for every $r \in R_0$, the mapping $h'|_{var(Q[V_r])}$ is a match of $Q[V_r]$ in J_r, and $h'|_{var(Q[V_r])}$ extends $h|_{var(Q[V_r])}$. This implies that (b) does not hold and yields the desired contradiction. □

We now combine the previous two lemmas to obtain our upper bounds on the complexity of answering UBCQs with respect to stratified sets of guarded DNTGDs. As in the case of atomic queries, we first provide upper bounds on the complexity of a suitable problem that we will use as an "oracle":

QCHECK
Input: a stratified set Σ of guarded DNTGDs over a schema $\mathcal{R}$, an $\mathcal{R}$-atom β, a UBCQ Q over $\mathcal{R}$, and a (Σ, Q)-environment $(E, \mathcal{Q})$ for $\{\beta\}$
Question: $CMod(\{\beta\}, \Sigma \mid (E, \mathcal{Q})) \neq \emptyset$?

Theorem B.19. QCHECK *is in* 2-EXPTIME. *It is in* PTIME *if* $|\mathcal{R}|$, $\mathrm{ar}(\mathcal{R})$, *and* $\mathrm{wd}(Q)$ *are bounded by a constant.*

Proof. We describe an alternating Turing machine for QCHECK. This machine is similar to the one for ACHECK constructed in the proof of Theorem B.16. Essentially, the only difference is that here we use the reduction lemma for UBCQs (Lemma B.18) instead of the reduction lemma for atomic queries (Lemma B.15).

As in the proof of Theorem B.16, we use mappings can_α to canonize objects such as atoms, literals, sets of literals, and (Σ, Q)-environments that are built from the elements in $dom(\alpha)$. We use configurations $(\beta, E, \mathcal{Q}, c)$, where β is an atom, $(E, \mathcal{Q})$ with $E = (T, S, R^+, R^-, \preceq)$ is a (Σ, Q)-environment for $\{\beta\}$, and c is a mapping that maps each atom in $T^+ \setminus (\{\beta\} \cup S)$ and each rule in R^+ to an integer in $\{0, 1, \ldots, d^*\}$, where d^* (whose concrete value was already given in Section 5.2.2) is formally defined as follows:

$$d^* := |\{(can_\beta(\beta), can_\beta(E), can_\beta(\mathcal{Q})) \mid \beta \text{ is an } \mathcal{R}\text{-atom, and } (E, \mathcal{Q}) \text{ is a } (\Sigma, Q)\text{-environment for } \{\beta\}\}|.$$

The goal in such a configuration is to verify that $CMod(\{\beta\}, \Sigma \mid (E, \mathcal{Q}))$ is non-empty. As before, we represent β, E, and $\mathcal{Q}$ in canonical form.

Now, given as input a stratified set Σ of guarded DNTGDs, an atom β_0, and a (Σ, Q)-environment $(E_0, \mathcal{Q}_0)$ with $E_0 = (T, S, R^+, R^-, \preceq)$, the machine starts in configuration $(\beta_0, E_0, \mathcal{Q}_0, c_0)$, where c_0 assigns the integer d^* to all atoms in $T^+ \setminus (\{\beta_0\} \cup S)$ and to all Σ-rules in R^+. If the current configuration is $(\beta, E, \mathcal{Q}, c)$ with $E = (T, S, R^+, R^-, \preceq)$, then it proceeds as follows:

(1) If $c(\alpha) = 0$ for some $\alpha \in T^+ \setminus (\{\beta\} \cup S)$, or if $c(r) = 0$ for some $r \in R^+$, then reject.
(2) Guess a completion $E^* = (T^*, S, R^+, R^-, \preceq^*)$ of E according to Lemma B.17 (2). Extend c to $(T^*)^+ \setminus (\{\beta\} \cup S)$ by setting $c(\alpha) := d^*$ for each $\alpha \in (T^*)^+ \setminus (\{\beta\} \cup S)$ with $\alpha \notin T^+$.
(3) Guess a Σ-expansion $(R_0, R, \preccurlyeq_R, \rho, \delta)$ of $\{\beta\}$ relative to E^* and compute the corresponding complete (Σ, Q)-environments $(E_r, \mathcal{Q}_r)$ for $r \in R_0$ according to Lemma B.18. If no such Σ-expansion exists, then reject.
(4) For each $r \in R_0$, do the following:
   (a) Compute $E_r = (T_r, S_r, R_r^+, R_r^-, \preceq_r)$ as defined in Lemma B.15.

(b) For each $\alpha \in T_r^+ \setminus (\{\beta\} \cup S_r)$, let $c_r(\alpha) := c(\alpha) - 1$. For each $r' \in R_r^+$, let $c_r(r') := c(r') - 1$ if $r' \in R^+$, and $c_r(r') := d^*$ if $r' \notin R^+$.

(5) Accept iff for each $r \in R_0$ the computation starting in configuration $(\beta_r, E_r, \boldsymbol{Q}_r, c_r)$ is accepting.

It is straightforward to check that the space required by the machine is exponential in general, and logarithmic if $\text{ar}(\mathcal{R})$, $|\mathcal{R}|$, and $\text{wd}(Q)$ are bounded by a constant. The proof of correctness is a straightforward modification of the proof of Theorem B.16, where the nodes in the self-reduction tree are labeled by triples $(\beta, E, \boldsymbol{Q})$ instead of pairs (β, E). This is to ensure that reduction steps that are applied in the proof of the "if" direction preserve "non-satisfaction" of Q. □

Theorem B.19 now leads to tight bounds for answering UBCQs with respect to stratified sets of guarded DNTGDs, and we are finally able to accomplish the proof of Theorem 5.9.

Theorem 5.9 (Upper Bounds of Theorem 5.6 Restricted to UBCQs). *Given as input a database D for a schema $\mathcal{R}$, a finite stratified set Σ of guarded DNTGDs over $\mathcal{R}$, and a covered UNBCQ Q over $\mathcal{R}$, deciding $(D, \Sigma) \models_{\text{strat}} Q$ is:*

(1) in 2-EXPTIME *in general;*
(2) in EXPTIME, *if* $\text{ar}(\mathcal{R})$ *is bounded by a constant, and Q is acyclic;*
(3) in co-NP *in data complexity, or, if* $|\mathcal{R}|$, $\text{ar}(\mathcal{R})$, *and* $\text{wd}(Q)$ *are bounded by a constant.*

Proof. To prove the upper bounds, we describe a Turing machine TM that verifies $(D, \Sigma) \not\models_{\text{strat}} Q$ within the desired time bounds. Given D, Σ, and Q as input, the machine TM first computes E_D and $\boldsymbol{Q}_{D,Q}$ as defined in Lemma B.17 (1). TM then guesses a completion $(E^*, \boldsymbol{Q}^*)$ of $(E_D, \boldsymbol{Q}_{D,Q})$ according to Lemma B.17 (2), and a Σ-expansion $(R_0, R, \leqslant_R, \rho, \delta)$ of D relative to E^* as well as (Σ, Q)-environments $(E_r, \boldsymbol{Q}_r)$ for $r \in R_0$ according to Lemma B.18. In the general case, we replace these non-deterministic steps by loops over all completions, Σ-expansions, and (Σ, Q)-environments $(E_r, \boldsymbol{Q}_r)$. The machine accepts the input iff $\mathit{CMod}(\{\beta_r\}, \Sigma \mid (E_r, \boldsymbol{Q}_r)) \neq \emptyset$ for each $r \in R_0$. From Lemma B.17 and Lemma B.18, it follows that the machine accepts the input iff $(D, \Sigma) \not\models_{\text{strat}} Q$. Theorem B.19 implies that $\mathit{CMod}(\{\beta_r\}, \Sigma \mid (E_r, \boldsymbol{Q}_r)) \neq \emptyset$ can be checked in 2-EXPTIME in general, and in PTIME if $|\mathcal{R}|$, $\text{ar}(\mathcal{R})$, and $\text{wd}(Q)$ are bounded. This implies the upper bounds given in points (1) and (2).

If $\text{ar}(\mathcal{R})$ is bounded, and Q is acyclic, then we simulate Q by adding to Σ a polynomial number of guarded TGDs that imply a goal predicate $\mathit{Goal}()$ in a canonical model J iff Q is true in J. It is easy to see that this can be done in such a way that the resulting set Σ' of guarded DNTGDs is stratified and contains at most $|Q|$ additional predicates whose arities are at most $\text{ar}(\mathcal{R})$. Then, $(D, \Sigma) \not\models_{\text{strat}} Q$ iff $(D, \Sigma') \not\models_{\text{strat}} \mathit{Goal}()$. Thus, the upper bound in point (2) follows from Theorem B.1. □

C DETAILS AND PROOFS FOR SECTION 6

We now prove Theorems 6.13 and 6.14. We first prove Theorem 6.13 by induction on the definition of the canonical model of stratified sets of DNTGDs via the construction of the chase (see Definition 4.10). To this end, we need to extend the chase construction by the application of EGDs as follows: An EGD σ on $\mathcal{R}$ of the form $\Phi(\mathbf{x}) \rightarrow x_i = x_j$ is *applicable* to a database D for $\mathcal{R}$ if there exists a homomorphism $\eta\colon \Phi(\mathbf{x}) \rightarrow D$ such that $\eta(x_i)$ and $\eta(x_j)$ are different and not both constants. If $\eta(x_i)$ and $\eta(x_j)$ are different constants in Δ or cannot be unified via homomorphisms, then there is a *hard violation* of σ, and the chase *fails*. Otherwise, the result of the application of σ to D is the database $h(D)$ obtained from D by replacing every occurrence of an element $e \in \{\eta(x_i), \eta(x_j)\}$ in D by $\theta(\eta(x_i))$, where θ is a most general homomorphism that unifies $\eta(x_i)$ and $\eta(x_j)$ such that $\theta(\eta(x_i))$ is the smallest null in the

lexicographic order under all such homomorphisms. Note that h is a homomorphism, but not necessarily an endomorphism of D, since $h(D)$ is not necessarily a subset of D. But for the special class of DNTGDs and EGDs that we define in this section, h is actually an endomorphism of D.

The *chase* of a database D, in the presence of a stratified set Σ_T of DNTGDs and a set Σ_E of EGDs, denoted $CMod(D, \Sigma_T \cup \Sigma_E,)$, is computed by iteratively applying (1) a single DNTGD once, according to the standard order, and (2) the EGDs, as long as they are applicable (i.e., until a fixpoint is reached).

Example C.1. Consider the following set $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ of DNTGDs and EGDs:

$$\begin{array}{lrcl} \sigma_1: & r(x,y) & \rightarrow & \exists z\, s(x,y,z), \\ \sigma_2: & s(x,y,z) & \rightarrow & y = z, \\ \sigma_3: & r(x,y), s(z,y,y) & \rightarrow & x = y. \end{array}$$

Let D be the database $\{r(a,b)\}$. In the computation of $chase(D,\Sigma)$, we first apply σ_1 and add the fact $s(a,b,z_1)$, where z_1 is a null. Then, the application of σ_2 on $s(a,b,z_1)$ yields $z_1 = b$, thus turning $s(a,b,z_1)$ into $s(a,b,b)$. Now, we apply σ_3 on $r(a,b)$ and $s(a,b,b)$, and by equating $a = b$, the chase fails; this is a hard violation, since $a, b \in \Delta$.

We are now ready to prove Theorem 6.13.

Theorem 6.13. *Let $\mathcal{R}$ be a relational schema, let Σ_T be a stratified set of guarded DNTGDs on $\mathcal{R}$, and let Σ_K be a set of keys on $\mathcal{R}$. Then: If Σ_K is SNC with Σ_T, then Σ_K is separable from Σ_T relative to UBCQs.*

Proof. Intuitively, the main idea behind the proof is to show by induction on the construction of the iterative chase that each application of a key induces a homomorphism that does not change atoms whose negations are required later by other rule applications. The positions in K_∞ are the only positions whose values change when such a homomorphism is applied. Since no negative literal in a rule body contains a variable that occurs at a position in K_∞ in the positive part of the rule body, such a homomorphism then leaves negative literals unchanged.

Formally, suppose that, in the process of constructing the oblivious chase for a stratified set Σ_T of guarded DNTGDs, a DNTGD σ fires, deriving an atom $\mathbf{a}$ with the predicate r. Take any key $\kappa \in \Sigma_K$ for r, where $\mathbf{K}$ is the corresponding set of positions of r. Then, as the set $\mathbf{H}_\sigma$ of positions in $\mathbf{a}$ occupied by universally quantified variables is not a proper superset of the set $\mathbf{K}$, there are only two possible cases: (i) $\mathbf{K} = \mathbf{H}_\sigma$: In this case, κ is the only key that can fire. By our particular chase order, this key is immediately applied and just eliminates the new atom $\mathbf{a}$ generated by σ, because $\mathbf{a}$ has fresh nulls in all positions but those of $\mathbf{K}$. (ii) At least one position in $\mathbf{K}$ is occupied by an existentially quantified variable in $\mathbf{a}$: Then, the new fact $\mathbf{a}$ generated by the application of σ contains a fresh null in a position in $\mathbf{K}$, and therefore it cannot violate the key κ. It follows that the oblivious chase only eliminates some facts generated by some DNTGDs, and we now intuitively have to make sure that these facts are actually isomorphic to what has been derived before, as otherwise removing them would make a difference, as illustrated by Example 6.10.

Let D be a database, Σ_T be a stratified set of guarded DNTGDs, and κ be a key such that κ is SNC with Σ_T. Let $D \models \kappa$. Then, by (i) and (ii) above, constructing $CMod(D, \Sigma_T \cup \Sigma_K)$ converges to possibly infinite fixpoints without ever producing a hard violation. Furthermore, suppose that, in the process of constructing $CMod(D, \Sigma_T \cup \Sigma_K)$, the key κ is violated while deriving the atom $\mathbf{a}$ from an element S of the chase constructed thus far, as in (i) above. Let h be the retraction of S induced by applying κ on S. It is then not difficult to see that for all $\alpha \in dom(S)$ that do not occur in S at a position of K_∞, we have $h(\alpha) = \alpha$. Consequently, for all $\sigma \in \Sigma_T$, all mappings μ with

$S \models \mu(body(\sigma))$, and all $\alpha \in \mu(body^-(\sigma))$, we have $h(\alpha) = \alpha$. In particular, $h(S) \models h(\mu(body(\sigma)))$. Hence, there is a homomorphism from $S \in CMod(D, \Sigma_T)$ to some $S' \in CMod(D, \Sigma_T \cup \Sigma_K)$ (which is in fact an endomorphism from S to S') that is defined by the union of all substitutions performed by those keys that are applied. Conversely, there is also a homomorphism from any $S \in CMod(D, \Sigma_T \cup \Sigma_K)$ to some $S' \in CMod(D, \Sigma_T)$, as there is always some $S' \in CMod(D, \Sigma_T)$ with $S \subseteq S'$. Thus, for every UBCQ Q, it holds that Q is true in all $S \in CMod(D, \Sigma_T \cup \Sigma_E)$ iff Q is true in all $S \in CMod(D, \Sigma_T)$. In summary, Σ_K is separable from Σ_T relative to UBCQs. □

We finally prove Theorem 6.14.

Theorem 6.14. *Let $\mathcal{R}$ be a relational schema, let Σ_T be a stratified set of guarded DNTGDs on $\mathcal{R}$, and let Σ_K be a set of keys on $\mathcal{R}$. Then: If Σ_K is SNC with Σ_T, then Σ_K is separable from Σ_T relative to strongly covered UNBCQs.*

Proof. The result follows from Theorem 6.13, applied to the UBCQ Q' relative to Σ'_T and Σ_K, where Q' and Σ'_T are obtained from Q and Σ_T by replacing every negative literal $\neg\alpha$ in Q by α' and by adding the corresponding rule $\beta \wedge \neg\alpha \rightarrow \alpha'$ to Σ_T, respectively, where β is a cover of $\neg\alpha$ in Q, and α' replaces the predicate R in α by a fresh predicate R' that simulates the complement of R-predicates. It is then not difficult to verify that K_∞ relative to Σ'_T and Σ_K is obtained from K_∞ relative to Σ_T and Σ_K by eventually adding the positions of fresh predicates, which occur only in Q', but not in rule bodies in Σ'_T. Thus, as Q is strongly covered, Σ_K is SNC with Σ'_T iff Σ_K is SNC with Σ_T. □

ACKNOWLEDGMENTS

We thank the referees for their valuable comments and suggestions. We thank Pedro Cabalar and David Pearce for helpful elucidations on Equilibrium Logic and its relationship to the Second-Order Stable-Model Semantics (SOS).

REFERENCES

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
[2] Mario Alviano, Carmine Dodaro, Wolfgang Faber, Nicola Leone, and Francesco Ricca. 2013. WASP: A native ASP solver based on constraint learning. In *Proceedings of the LPNMR*. Springer, 54–66.
[3] Mario Alviano, Wolfgang Faber, Nicola Leone, and Marco Manna. 2012. Disjunctive Datalog with existential quantifiers: Semantics, decidability, and complexity issues. *Theor. Pract. Log. Prog.* 12, 4/5 (2012), 701–718.
[4] Mario Alviano, Michael Morak, and Andreas Pieris. 2017. Stable model semantics for tuple-generating dependencies revisited. In *Proceedings of the PODS*. 377–388.
[5] Mario Alviano and Andreas Pieris. 2015. Default negation for non-guarded existential rules. In *Proceedings of the PODS*. 79–90.
[6] Hajnal Andréka, István Németi, and Johan van Benthem. 1998. Modal languages and bounded fragments of predicate logic. *J. Phil. Logic* 27 (1998), 217–274.
[7] Marcelo Arenas, Pablo Barceló, Leonid Libkin, and Filip Murlak. 2014. *Foundations of Data Exchange*. Cambridge University Press.
[8] Marcelo Arenas, Georg Gottlob, and Andreas Pieris. 2014. Expressive languages for querying the Semantic Web. In *Proceedings of the PODS*. 14–16.
[9] Franz Baader, Sebastian Brandt, and Carsten Lutz. 2005. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the IJCAI*. 364–369.
[10] Jean-François Baget, Laurent Garcia, Fabien Garreau, Claire Lefèvre, Swan Rocher, and Igor Stéphan. 2018. Bringing existential variables in answer set programming and bringing non-monotony in existential rules: Two sides of the same coin. *Ann. Math. Artif. Intell.* 82, 1–3 (2018), 3–41.
[11] Jean-François Baget, Michel Leclère, and Marie-Laure Mugnier. 2010. Walking the decidability line for rules with existential variables. In *Proceedings of the KR*. 466–476.
[12] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. 2009. Extending decidable cases for rules with existential variables. In *Proceedings of the IJCAI*. 677–682.

[13] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. 2011. Walking the complexity lines for generalized guarded existential rules. In *Proceedings of the IJCAI*. 712–717.
[14] Catriel Beeri and Moshe Y. Vardi. 1981. The implication problem for data dependencies. In *Proceedings of the ICALP*. 73–85.
[15] Catriel Beeri and Moshe Y. Vardi. 1984. A proof procedure for data dependencies. *J. ACM* 31, 4 (1984), 718–741.
[16] Luigi Bellomarini, Ruslan R. Fayzrakhmanov, Georg Gottlob, Andrey Kravchenko, Eleonora Laurenza, Yavor Nenov, Stéphane Reissfelder, Emanuel Sallinger, Evgeny Sherkhonov, and Lianlong Wu. 2018. Data science with Vadalog: Bridging machine learning and reasoning. In *Proceedings of the MEDI*. Springer, 3–21.
[17] Luigi Bellomarini, Georg Gottlob, Andreas Pieris, and Emanuel Sallinger. 2018. Swift logic for big data and knowledge graphs. In *Proceedings of the SOFSEM*. Springer, 3–16.
[18] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. 2018. The Vadalog system: Datalog-based reasoning for knowledge graphs. *Proc. VLDB Endow.* 11, 9 (2018), 975–987.
[19] Gerald Berger, Georg Gottlob, Andreas Pieris, and Emanuel Sallinger. 2019. The space-efficient core of Vadalog. In *Proceedings of the PODS*. 270–284.
[20] Philip A. Bernstein and Dah-Ming W. Chiu. 1981. Using semi-joins to solve relational queries. *J. ACM* 28, 1 (1981), 25–40.
[21] Philip A. Bernstein and Nathan Goodman. 1981. Power of natural semijoins. *SIAM J. Comput.* 10, 4 (1981), 751–771.
[22] Leopoldo Bertossi, Georg Gottlob, and Reinhard Pichler. 2018. Datalog: Bag semantics via set semantics. *arXiv preprint arXiv:1803.06445* (2018).
[23] Meghyn Bienvenu and Riccardo Rosati. 2013. Tractable approximations of consistent query answering for robust ontology-based data access. In *Proceedings of the IJCAI*. 775–781.
[24] Piero Bonatti, Marco Faella, Carsten Lutz, Luigi Sauro, and Frank Wolter. 2015. Decidability of circumscribed description logics revisited. In *Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation*. Springer, 112–124.
[25] Piero A. Bonatti. 2004. Reasoning with infinite stable models. *Artif. Intell.* 156, 1 (2004), 75–111.
[26] Piero A. Bonatti. 2008. Erratum to: Reasoning with infinite stable models [*Artif. Intell.* 156(1):75–111, 2004]. *Artif. Intell.* 172, 15 (2008), 1833–1835.
[27] Pier A. Bonatti. 2011. On the decidability of FDNC programs. *Intelligenza Artificiale* 5, 1 (2011), 89–93.
[28] Pierre Bourhis, Marco Manna, Michael Morak, and Andreas Pieris. 2016. Guarded-based disjunctive tuple-generating dependencies. *ACM T. Database Syst.* 41, 4 (2016), 1–45.
[29] Pierre Bourhis, Michael Morak, and Andreas Pieris. 2013. The Impact of disjunction on query answering under guarded-based existential rules. In *Proceedings of the IJCAI*. 796–802.
[30] Loreto Bravo and Leopoldo Bertossi. 2005. Deductive databases for computing certain and consistent answers from mediated data integration systems. *J. Appl. Log.* 3, 2 (2005), 329–367.
[31] Pedro Cabalar, Jorge Fandinno, Luis Farinas Del Cerro, and David Pearce. 2018. Functional ASP with intensional sets: Application to Gelfond-Zhang aggregates. *Theor. Pract. Log. Prog.* 18, 3/4 (2018), 390–405.
[32] Andrea Calì, Georg Gottlob, and Michael Kifer. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.* 48 (2013), 115–174.
[33] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.* 14 (2012), 57–83.
[34] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. 2010. Datalog+/−: A family of logical knowledge representation and query languages for new applications. In *Proceedings of the LICS*. 228–242.
[35] Andrea Calì, Georg Gottlob, and Andreas Pieris. 2012. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.* 193 (2012), 87–128.
[36] Andrea Calì, Domenico Lembo, and Riccardo Rosati. 2003. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of the PODS*. 260–271.
[37] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. 2011. The MASTRO system for ontology-based data access. *Seman. Web* 2, 1 (2011), 43–53.
[38] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.* 39, 3 (2007), 385–429.
[39] Ashok K. Chandra, Harry R. Lewis, and Johann A. Makowsky. 1981. Embedded implicational dependencies and their inference problem. In *Proceedings of the STOC*. 342–354.
[40] Ashok K. Chandra and Moshe Y. Vardi. 1985. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. Comput.* 14 (1985), 671–677.

[41] Jack Clearman, Ruslan R. Fayzrakhmanov, Georg Gottlob, Yavor Nenov, Stéphane Reissfelder, Emanuel Sallinger, and Evgeny Sherkhonov. 2019. Feature engineering and explainability with Vadalog: A recommender systems application. In *Proceedings of the Datalog 2.0 2019 — 3rd International Workshop on the Resurgence of Datalog in Academia and Industry*.

[42] Bruno Courcelle. 1990. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.* 85, 1 (1990), 12–75.

[43] Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-order Logic: A Language-theoretic Approach*. Vol. 138. Cambridge University Press.

[44] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv.* 33, 3 (2001), 374–425.

[45] Alin Deutsch, Alan Nash, and Jeff B. Remmel. 2008. The chase revisisted. In *Proceedings of the PODS*. 149–158.

[46] Alin Deutsch and Val Tannen. 2001. Optimization properties for classes of conjunctive regular path queries. In *Proceedings of the DBLP*. 21–39. DOI: https://doi.org/10.1007/3-540-46093-4_2

[47] Alin Deutsch and Val Tannen. 2003. Reformulation of XML queries and constraints. In *Proceedings of the ICDT*. 225–241.

[48] Wolfgang Dvořák, Georg Gottlob, Reinhard Pichler, and Stefan Woltran. 2009. Alternation as a programming paradigm. In *Proceedings of the PPDP*. 61–72.

[49] Thomas Eiter, Georg Gottlob, and Helmut Veith. 1997. Modular logic programming and generalized quantifiers. In *Proceedings of the LPNMR*. Springer, 290–309.

[50] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. 2008. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.* 172, 12/13 (2008), 1495–1539.

[51] Thomas Eiter and Mantas Šimkus. 2010. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.* 11, 2 (2010).

[52] Ronald Fagin. 1983. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM* 30, 3 (1983), 514–550.

[53] Ronald Fagin, Laura M. Haas, Mauricio Hernández, Renée J. Miller, Lucian Popa, and Yannis Velegrakis. 2009. CLIO: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*. Springer, 198–236.

[54] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124.

[55] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. 2005. Data exchange: Getting to the core. *ACM Trans. Datab. Syst.* 30, 1 (2005), 174–210.

[56] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. 2007. A new perspective on stable models. In *Proceedings of the IJCAI*. 372–379.

[57] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. 2011. Stable models and circumscription. *Artif. Intell.* 175, 1 (2011), 236–263.

[58] Ariel Fuxman, Phokion G. Kolaitis, Renée J. Miller, and Wang Chiew Tan. 2006. Peer data exchange. *ACM Trans. Datab. Syst.* 31, 4 (2006), 1454–1498.

[59] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. 2011. Potassco: The Potsdam answer set solving collection. *AI Commun.* 24, 2 (2011), 107–124.

[60] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. 2007. *clasp*: A conflict-driven answer set solver. In *Proceedings of the LPNMR*. Springer, 260–265.

[61] Gaetano Geck, Bas Ketsman, Frank Neven, and Thomas Schwentick. 2019. Parallel-correctness and containment for conjunctive queries with union and negation. *ACM Trans. Comput. Log.* 20, 3 (2019), 1–24.

[62] Michael Gelfond and Vladimir Lifschitz. 1988. The stable model semantics for logic programming. In *Proceedings of the ICLP/SLP*. 1070–1080.

[63] Laura Giordano and Daniele Theseider Dupré. 2016. ASP for minimal entailment in a rational extension of $\mathcal{SROEL}$. *Theor. Pract. Log. Prog.* 16, 5/6 (2016), 738–754.

[64] M. Elisabeth Goncalves and Erich Grädel. 2000. Decidability issues for action guarded logics. In *Proceedings of the DL*. 123–132.

[65] Georg Gottlob, Erich Grädel, and Helmut Veith. 2002. Datalog LITE: A deductive query language with linear time model checking. *ACM Trans. Comput. Log.* 3, 1 (2002), 42–79.

[66] Georg Gottlob, André Hernich, Clemens Kupke, and Thomas Lukasiewicz. 2012. Equality-friendly well-founded semantics and applications to description logics. In *Proceedings of the AAAI*. 757–764.

[67] Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2001. The complexity of acyclic conjunctive queries. *J. ACM* 48, 3 (2001), 431–498.

[68] Georg Gottlob, Marco Manna, Michael Morak, and Andreas Pieris. 2012. On the complexity of ontological reasoning under disjunctive existential rules. In *Proceedings of the MFCS*. 1–18.
[69] Georg Gottlob, Marco Manna, and Andreas Pieris. 2013. Combining decidability paradigms for existential rules. *Theor. Pract. Log. Prog.* 13, 4/5 (2013), 877–892.
[70] Georg Gottlob, Marco Manna, and Andreas Pieris. 2020. Multi-head guarded existential rules over fixed signatures. In *Proceedings of the KR*. 445–454.
[71] Georg Gottlob and Alan Nash. 2008. Efficient core computation in data exchange. *J. ACM* 55, 2 (2008).
[72] Erich Grädel, Colin Hirsch, and Martin Otto. 2002. Back and forth between guarded and modal logics. *ACM Trans. Comput. Log.* 3, 3 (2002), 418–463.
[73] Marc H. Graham. 1979. *On the Universal Relation*. Technical Report, University of Toronto, Computer Systems Research Group.
[74] Gösta Grahne and Adrian Onet. 2011. On conditional chase termination. In *Proceedings of the AMW* (2011), 46.
[75] B. Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res.* 47 (2013), 741–808.
[76] André Hernich. 2012. Computing universal models under guarded TGDs. In *Proceedings of the ICDT*. 222–235.
[77] André Hernich, Clemens Kupke, Thomas Lukasiewicz, and Georg Gottlob. 2013. Well-founded semantics for extended Datalog and ontological reasoning. In *Proceedings of the PODS*. 225–236.
[78] David S. Johnson and Anthony C. Klug. 1984. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.* 28, 1 (1984), 167–189.
[79] Nikolaos Konstantinou, Edward Abel, Luigi Bellomarini, Alex Bogatu, Cristina Civili, Endri Irfanie, Martin Koehler, Lacramioara Mazilu, Emanuel Sallinger, Alvaro A. A. Fernandes et al. 2019. VADA: An architecture for end user informed data preparation. *J. Big Data* 6, 1 (2019), 74.
[80] Markus Krötzsch and Sebastian Rudolph. 2011. Extending decidable existential rules by joining acyclicity and guardedness. In *Proceedings of the IJCAI*.
[81] Markus Krötzsch and Sebastian Rudolph. 2011. *Revisiting Acyclicity and Guardedness Criteria for Decidability of Existential Rules*. Technical Report 3011. Institute AIFB, Karlsruhe Institute of Technology.
[82] Markus Krötzsch and Sebastian Rudolph. 2013. *On the Relationship of Joint Acyclicity and Super-weak Acyclicity*. Technical Report 3037. Institute AIFB, Karlsruhe Institute of Technology.
[83] Nicola Leone, Marco Manna, Giorgio Terracina, and Pierfrancesco Veltri. 2012. Efficiently computable Datalog$^{\exists}$ programs. In *Proceedings of the KR*.
[84] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. 2006. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.* 7, 3 (2006), 499–562.
[85] John J. LeTourneau. 1968. *Decision Problems Related to the Concept of Operation*. Phd. Thesis. University of California, Berkeley.
[86] John W. Lloyd. 1984. *Foundations of Logic Programming*. Springer.
[87] Thomas Lukasiewicz. 2010. A novel combination of answer set programming with description logics for the semantic web. *IEEE Trans. Knowl. Data Eng.* 22, 11 (2010), 1577–1592.
[88] Despoina Magka, Markus Krötzsch, and Ian Horrocks. 2013. Computing stable models for nonmonotonic existential rules. In *Proceedings of the IJCAI*. 1031–1038.
[89] Despoina Magka, Markus Krötzsch, and Ian Horrocks. 2014. A rule-based ontological framework for the classification of molecules. *J. Biomed. Semant.* 5, 1 (2014), 17.
[90] Marco Maratea, Francesco Ricca, and Pierfrancesco Veltri. 2010. DLV^{MC}: Enhanced model checking in DLV. In *Proceedings of the JELIA*. Springer, 365–368.
[91] Victor W. Marek and Jeffrey B. Remmel. 2003. On the expressibility of stable logic programming. *Theor. Pract. Log. Prog.* 3, 4/5 (2003), 551–567.
[92] Victor W. Marek, Anil Nerode, and Jeffrey Remmel. 1992. How complicated is the set of stable models of a recursive logic program? *Ann. Pure Appl. Log.* 56, 1-3 (1992), 119–135.
[93] Bruno Marnette. 2009. Generalized schema-mappings: From termination to tractability. In *Proceedings of the PODS*. 13–22.
[94] Mostafa Milani, Leopoldo Bertossi, and Sina Ariyan. 2014. Extending contexts with ontologies for multidimensional data quality assessment. In *Proceedings of the 30th International Conference on Data Engineering Workshops*. 242–247.
[95] John C. Mitchell. 1983. The implication problem for functional and inclusion dependencies. *Inf. Contr.* 56, 3 (1983), 154–173.
[96] Boris Motik and Riccardo Rosati. 2010. Reconciling description logics and rules. *J. ACM* 57, 5 (2010).
[97] Adrian Constantin Onet. 2012. *The Chase Procedure and Its Applications*. Ph.D. Dissertation. Concordia University.

[98] David Pearce. 1996. A new logical characterisation of stable models and answer sets. In *Proceedings of the NMELP*. Springer, 57–70.
[99] David Pearce. 2006. Equilibrium logic. *Ann. Math. Artif. Intell.* 47, 1/2 (2006), 3.
[100] David Pearce and Agustin Valverde. 2004. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proceedings of the JELIA*. Springer, 147–160.
[101] David Pearce and Agustin Valverde. 2005. A first order nonmonotonic extension of constructive logic. *Stud. Log.* 80, 2 (2005), 321–346.
[102] David Pearce and Agustín Valverde. 2008. Quantified equilibrium logic and foundations for answer set programs. In *Proceedings of the ICLP*. Springer, 546–560.
[103] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking data to ontologies. *J. Data Seman.* 10 (2008), 133–173.
[104] Michael O. Rabin. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. Am. Math. Soc.* 141 (1969), 1–35.
[105] Thoralf Räsch. 2002. Introduction to guarded logics. In *Automata, Logics, and Infinite Games*. Springer, 321–341.
[106] Domenico Fabio Savo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Vittorio Romagnoli, Marco Ruzzi, and Gabriele Stella. 2010. MASTRO at work: Experiences on ontology-based data access. *Proc. DL* 573 (2010), 20–31.
[107] John S. Schlipf. 1995. The expressive powers of the logic programming semantics. *J. Comput. Syst. Sci.* 51, 1 (1995), 64–86.
[108] Saharon Shelah. 1975. The monadic theory of order. *Ann. Math.* 102, 3 (1975), 379–419.
[109] Robert E. Tarjan and Mihalis Yannakakis. 1984. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.* 13, 3 (1984), 566–579.
[110] Michaël Thomazo. 2011. *Conjunctive Query Answering under Existential Rules — Decidability, Complexity, and Algorithms*. Ph.D. Dissertation. University of Montpellier, France.
[111] Ron van der Meyden. 1998. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*. Springer, 307–356.
[112] Moshe Y. Vardi. 1982. The complexity of relational query languages. In *Proceedings of the STOC*. 137–146.
[113] Mantas Šimkus. 2010. *Nonmonotonic Logic Programs with Function Symbols*. Ph.D. Dissertation. TU Wien, Austria.
[114] Mantas Šimkus. 2015. Binary frontier-guarded ASP with function symbols. In *Proceedings of the RuleML*. 311–327.
[115] Hai Wan, Guohui Xiao, Chenglin Wang, Xianqiao Liu, Junhong Chen, and Zhe Wang. 2020. Query answering with guarded existential rules under stable model semantics. In *Proceedings of the AAAI*.
[116] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-based data access: A survey. In *Proceedings of the IJCAI*. 5511–5519.
[117] Clement Tak Yu and Meral Z. Ozsoyoglu. 1979. An algorithm for tree-query membership of a distributed query. In *Proceedings of the COMPSAC*. 306–312.

Received February 2018; revised May 2020; accepted January 2021