Equality-Friendly Well-Founded Semantics and Applications to Description Logics

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

[1] Department of Computer Science, University of Oxford, UK
{firstname.lastname}@cs.ox.ac.uk
[2] Oxford-Man Institute of Quantitative Finance, University of Oxford, UK
[3] Institut für Informatik, Humboldt-Universität zu Berlin, Germany
hernich@informatik.hu-berlin.de

Abstract. We tackle the problem of defining a well-founded semantics (WFS) for Datalog rules with existentially quantified variables in their heads and negations in their bodies. In particular, we provide a WFS for the recent Datalog$^{\pm}$ family of ontology languages, which covers several important description logics (DLs). To do so, we generalize Datalog$^{\pm}$ by non-stratified nonmonotonic negation in rule bodies, and we define a WFS for this generalization via guarded fixed point logic. We refer to this approach as *equality-friendly WFS*, since it has the advantage that it does not make the unique name assumption (UNA); this brings it close to OWL and its profiles as well as typical DLs, which also do not make the UNA. We prove that for guarded Datalog$^{\pm}$ with negation under the equality-friendly WFS, conjunctive query answering is decidable, and we provide precise complexity results for this problem. From these results, we obtain precise definitions of the standard WFS extensions of $\mathscr{EL}$ and of members of the *DL-Lite* family, as well as corresponding complexity results for query answering.

1 Introduction

The recent Datalog$^{\pm}$ family of ontology languages [7] extends plain Datalog by the possibility of existential quantification in rule heads and other features, and simultaneously restricts the rule syntax to achieve tractability. The following example illustrates how description logic (DL) knowledge bases are expressed in Datalog$^{\pm}$.

Example 1 (Literature) The knowledge that every conference paper is an article and that every scientist is the author of at least one paper can be expressed in DL by the TBox axioms $\mathit{ConferencePaper} \sqsubseteq \mathit{Article}$ and $\mathit{Scientist} \sqsubseteq \exists \mathit{isAuthorOf}$, respectively, while the knowledge that John is a scientist can be expressed by the ABox axiom $\mathit{Scientist}(\mathit{john})$. In Datalog$^{\pm}$, the former are encoded as the rule $\mathit{ConferencePaper}(X) \rightarrow \mathit{Article}(X)$ and the rule $\mathit{Scientist}(X) \rightarrow \exists Y\, \mathit{isAuthorOf}(X,Y)$, respectively, and the latter is encoded by an identical fact in the database. Furthermore, the TBox axiom that encodes that conference papers are not journal papers, can be expressed in Datalog$^{\pm}$ by the negative constraint $\mathit{ConferencePaper} \wedge \mathit{JournalPaper} \rightarrow \bot$. A simple Boolean conjunctive query (BCQ) asking whether John authors a paper is $\exists X\, \mathit{isAuthorOf}(\mathit{john}, X)$. ■

The Datalog$^{\pm}$ languages bridge an apparent gap in expressive power between database query languages and DLs as ontology languages, extending the well-known Dat-

alog language in order to embed DLs. They also allow for transferring important concepts and proof techniques from database theory to DLs. For example, it was so far not clear how to enrich tractable DLs by the feature of nonmonotonic negation. By the results of [7], DLs can be enriched by stratified negation via mappings from DLs to Datalog$^{\pm}$ with stratified negation, which is defined and studied in that paper. Given that stratified negation is quite limited, we wondered whether the richer and more expressive well-founded negation could be defined for Datalog$^{\pm}$. The well-founded semantics (WFS) for normal (logic) programs [25] is one of the most widely used semantics for nonmonotonic normal programs, it is the standard semantics for such programs for database applications, and it is thus especially under a data-oriented perspective of great importance for the Web. Having many nice features, the WFS is defined for all normal programs (i.e., logic programs with the possibility of negation in rule bodies), has a polynomial data tractability, approximates the answer set semantics, and coincides with the canonical model in case of stratified normal programs.

In this paper, we concentrate on the important problem of defining a WFS for (unrestricted) normal Datalog$^{\pm}$, i.e., Datalog with existentially quantified variables in rule heads and negations in rule bodies. This new semantics is called the *equality-friendly WFS (EFWFS)*, since it has the crucial advantage that it does not make the unique name assumption (UNA); this brings it close to OWL and its profiles as well as typical DLs, which also do not make the UNA.

Since (unrestricted) normal Datalog$^{\pm}$ generalizes positive Datalog$^{\pm}$, consistency checking and query answering in it is in general undecidable. However, it turns out that the guarded fragment of normal Datalog$^{\pm}$ can be translated to *guarded fixed point logic*, which is a well-studied decidable formalism. Through this translation, we thus obtain the decidability of consistency checking and query answering in guarded normal Datalog$^{\pm}$. Furthermore, we obtain upper complexity bounds, which are then also shown to be tight. Guarded Datalog$^{\pm}$ covers in particular the DLs $\mathscr{EL}$ and *DL-Lite*$_{\mathscr{R}}$ (which is underlying the OWL 2 QL profile). Therefore, our decidability results and upper complexity bounds carry over to these DLs. The following example illustrates how the WFS can be extended to such DLs.

Example 2 (Holidays) Consider an ABox containing the three holiday destinations $Dest(d_1)$, $Dest(d_2)$, and $Dest(d_3)$. Suppose that any destination that offers the opportunity to swim needs either direct access to the beach ($Beach(x)$) or a bus connection to some beach ($BeachBus(x,y)$). That is, at destinations where swimming is possible, we want to make sure that never both not $Beach$ and not $\exists BeachBus$ hold. This can be achieved by the following two rules:

$$Dest \sqcap Swimming \sqcap \mathrm{not}\, Beach \sqsubseteq \exists BeachBus; \tag{1}$$

$$Dest \sqcap Swimming \sqcap \mathrm{not}\, \exists BeachBus \sqsubseteq Beach. \tag{2}$$

Observe also that not $Swimming(d)$ would immediately imply the facts not $Beach(d)$ and not $\exists BeachBus(d)$, since it would make it impossible that either of the two axioms could be applied to derive new facts about d. ■

The following example shows a case where not making the UNA is more appropriate than making it under the WFS.

Example 3 (Company) Suppose we are given certain facts about employees and their employers. The following two concept membership axioms state that John and Sam are employees: $\mathit{Employee}(\mathit{John})$, $\mathit{Employee}(\mathit{Sam})$. To these axioms, we add a concept inclusion axiom that maintains that every employee must have an employer: $\mathit{Employee} \sqsubseteq \exists.\mathit{hasEmployer}$. Finally, we would like to test whether or not *John* and *Sam* work in the same company, which is expressed by the query $\exists x(\mathit{hasEmployer}(\mathit{John},x) \sqcap \text{not}\, \mathit{hasEmployer}(\mathit{Sam},x))$. Then, under the UNA, equality between all individuals (including new ones) is minimized, and we evaluate the query to true, which is not the case without UNA, where different Skolem terms may be interpreted by the same object. ■

2 Preliminaries

In this section, we briefly recall some basics on Datalog$^{\pm}$ [7].

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), and (ii) an infinite set of variables $\mathcal{V}$ (used in queries and constraints). We denote by $\mathbf{X}$ sequences of variables $X_1,\ldots,X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *relation names* (or *predicate symbols*, or simply *predicates*). A *term* t is a constant or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1,...,t_n)$, where P is an n-ary predicate, and $t_1,...,t_n$ are terms. A conjunction of atoms is often identified with the set of all its atoms.

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

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

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathscr{R}$, and a set of TGDs Σ on $\mathscr{R}$, the set of *models* of D and Σ, denoted $mods(D,\Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q,D,\Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D,\Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q,D,\Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [5], even with fixed schema and TGDs [6].

Negative Constraints. Another crucial ingredient of Datalog$^{\pm}$ for ontological modeling are *negative constraints* (or simply *constraints*), which are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ is a conjunction of atoms (not necessarily guarded), called its *body*. We usually omit the universal quantifiers, and we implicitly assume that all sets of constraints are finite here.

Normal TGDs and BCQs. Normal TGDs are TGDs that may also contain (default-) negated atoms in their bodies: Given a relational schema $\mathscr{R}$, a *normal TGD (NTGD)* σ has the form $\forall \mathbf{X} \forall \mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X},\mathbf{Z})$, where $\Phi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathscr{R}$, and $\Psi(\mathbf{X},\mathbf{Z})$ is a conjunction of atoms over $\mathscr{R}$. We usually omit the universal quantifiers. As for standard TGDs, w.l.o.g., $\Psi(\mathbf{X},\mathbf{Z})$ is a singleton atom. We say σ is *satisfied* in a database D for $\mathscr{R}$ iff, whenever there is a homomorphism h for all the variables and constants in the body of σ that maps (i) positive literals of $\Phi(\mathbf{X},\mathbf{Y})$ to atoms of D and (ii) no negated atom of $\Phi(\mathbf{X},\mathbf{Y})$ to an atom of D, then there is an extension h' of h that maps the head atom to an atom of D. We call σ *guarded* iff it contains a positive body atom that contains all universally quantified variables of σ. W.l.o.g., constants in the body of guarded σ occur only in guards.

We next add negation to BCQs. A *normal Boolean conjunctive query (NBCQ)* Q is an existentially closed conjunction of atoms and negated atoms $\exists \mathbf{X}\, p_1(\mathbf{X}) \wedge \cdots \wedge p_m(\mathbf{X}) \wedge \neg p_{m+1}(\mathbf{X}) \wedge \cdots \wedge \neg p_{m+n}(\mathbf{X})$, where $m \geqslant 1$, $n \geqslant 0$, and the variables of the p_i's are among $\mathbf{X}$. Q is *covered* if for every negative atom α in Q there is a positive atom in Q containing every argument in α. In the sequel, w.l.o.g., NBCQs contain no constants.

3 Equality-Friendly WFS for Datalog$^{\pm}$

In this section, we first recall the well-founded semantics (WFS) of normal programs. As a central new contribution, we then introduce a WFS of normal Datalog$^{\pm}$ programs without the UNA.

WFS of Normal Programs. The WFS [25] is the most widely used semantics for nonmonotonic programs, it is the standard semantics for such programs for database applications, and it is thus especially under a data-oriented perspective of great importance for the Web. For our purposes, it is enough to recall the WFS of *function-free ground* normal programs, and we refer to [25] for the general case.

We first give some preliminary definitions. A function-free normal program is a finite set of rules r of the form $\beta_1, \ldots, \beta_n, \neg\beta_{n+1}, \ldots, \neg\beta_{n+m} \rightarrow \alpha$, where $\alpha, \beta_1, \ldots, \beta_{n+m}$ are atoms and $m,n \geqslant 0$. We call α the *head* of r, denoted $H(r)$, while the conjunction $\beta_1, \ldots, \beta_n, \neg\beta_{n+1}, \ldots, \neg\beta_{n+m}$ constitutes its *body*. Let $B(r) = B^+(r) \cup B^-(r)$, where

$B^+(r)=\{\beta_1,\ldots,\beta_n\}$, and $B^-(r)=\{\beta_{n+1},\ldots,\beta_{n+m}\}$. The program is *ground* if it contains no variables. Let P be a function-free ground normal program. The *Herbrand base* of P, denoted HB_P, is the set of all atoms that can be constructed from the predicate symbols and the constants appearing in P. For all $S \subseteq HB_P$, we let $\neg.S = \{\neg.a \mid a \in S\}$. We denote by $Lit_P = HB_P \cup \neg.HB_P$ the set of all literals with predicate symbols and constants from P. A *(three-valued) interpretation* relative to P is any set $I \subseteq Lit_P$ that is *consistent* (i.e., there is no atom $a \in HB_P$ with $\{a, \neg a\} \subseteq I$).

The WFS has many different equivalent definitions (see also [4]). We here recall the one based on unfounded sets, via the operators U_P, T_P, and W_P. A set $U \subseteq HB_P$ is an *unfounded set* of P relative to $I \subseteq Lit_P$ iff for every $a \in U$ and every rule $r \in P$ with $H(r)=a$, either (i) $\neg b \in I \cup \neg.U$ for some atom $b \in B^+(r)$, or (ii) $b \in I$ for some atom $b \in B^-(r)$. There exists the greatest unfounded set of P relative to I, denoted $U_P(I)$. Intuitively, it collects all those atoms that cannot become true when extending I with further information. We are now ready to define the two operators T_P and W_P on interpretations $I \subseteq Lit_P$ relative to P by:

$$T_P(I)=\{H(r) \mid r \in P, B^+(r) \cup \neg.B^-(r) \subseteq I\}; \quad W_P(I)=T_P(I) \cup \neg.U_P(I).$$

Since W_P is monotonic, it has a least fixed point, denoted $lfp(W_P)$, which is the *well-founded semantics (WFS)* of P, denoted $WFS(P)$. Intuitively, starting with $I=\emptyset$, rules are applied to obtain new positive and negated facts (via $T_P(I)$ resp. $\neg.U_P(I)$). This is repeated until no longer possible.

EFWFS of Normal Datalog$^\pm$ Programs. We relate normal Datalog$^\pm$ programs to sets of function-free ground normal programs, and define their equality-friendly WFS (EFWFS) as the set of well-founded models of the associated normal programs.

The basic idea is as follows. If we do not make the UNA, different constants in a normal Datalog$^\pm$ program P may represent the same value. Thus, P may turn out to be any of the programs P' obtained from P by identifying constants. Furthermore, in every such program P', existential quantifiers may introduce one or more value, which, since we do not make the UNA, does not have to be "fresh", but can be any constant. Hence, without the UNA, the meaning of P may be captured by the set of all normal programs P'' obtained from P by identifying values, and replacing TGDs in P by arbitrary instances, at least one for each possible variable assignment for its body. It is then natural to consider the well-founded models of all those programs P'' as the semantics of P.

More precisely, an *instance* of a normal TGD $\Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X},\mathbf{Z})$ is a rule of the form $\Phi(\mathbf{a},\mathbf{b}) \rightarrow \Psi(\mathbf{a},\mathbf{c})$, where $\mathbf{a},\mathbf{b},\mathbf{c}$ are tuples of constants. Let $P = D \cup \Sigma$ be a normal Datalog$^\pm$ program, where D is a database and Σ a set of normal TGDs. An *instance* $\mathscr{I}$ of P is a normal program consisting of all facts in D, and instances of TGDs in Σ such that for all TGDs $\Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X},\mathbf{Z})$ in Σ, and all interpretations $\mathbf{a},\mathbf{b}$ for $\mathbf{X},\mathbf{Y}$, there is at least one $\mathbf{c}$ such that $\Phi(\mathbf{a},\mathbf{b}) \rightarrow \Psi(\mathbf{a},\mathbf{c})$ is in $\mathscr{I}$. Let $\mathscr{I}(P)$ be the set of all instances of P, and $dom(P)$ the set of all constants in P. For any $\mu\colon dom(P) \rightarrow \Delta$, let P_μ be the program obtained from P by replacing all constants c with $\mu(c)$. Then, the *equality-friendly well-founded semantics* of P, denoted $EFWFS(P)$, is the set $\{\mathrm{WFS}(P'') \mid \mu\colon dom(P) \rightarrow \Delta, P'' \in \mathscr{I}(P_\mu)\}$. Query-answering for an NBCQ Q is now defined by saying that Q evaluates to *Yes* in $EFWFS(P)$ iff Q evaluates to *Yes* in all elements of $EFWFS(P)$. Here, an NBCQ Q evaluates to *Yes* in a

three-valued interpretation I iff there is a homomorphism that maps all elements of Q^+ into atoms in I and all elements of Q^- into negated atoms in I.

Example 4 Consider the following program P:

$$\begin{aligned} &\rightarrow A(0); && A(X) \rightarrow \exists Y_1, Y_2\,(R(X, Y_1, Y_2)); \\ R(X_1, X_2, X_3), \neg A(X_3) &\rightarrow S(0); && A(X), \neg S(X) \rightarrow \exists Y_1, Y_2\,(R(Y_1, Y_2, X)). \end{aligned}$$

Obviously, $A(0)$ is in the EFWFS of P. Furthermore, because of the second rule, every possible well-founded model of P contains $R(0, c_1, c_2)$. But we cannot assume $c_2 \neq 0$. If $c_2 = 0$, all possible applications of the third rule are blocked, and thus $\neg S(0)$ would be derived (the last rule may then still add another atom $R(d_1, d_2, 0)$, but this does not affect the overall result). Otherwise, in case $c_2 \neq 0$, the third rule would yield $S(0)$, because clearly we have $\neg A(c_2)$. Therefore, neither $S(0)$ nor its negation is in $EFWFS(P)$ and the only atoms that are always in $EFWFS(P)$ are $A(0)$ and $R(0, c_1, c_2)$ for some constants c_1 and c_2 (so, the query $\exists X_1, X_2\,(R(0, X_1, X_2))$ would evaluate to true). The picture changes, if we add the constraint $R(X_1, X_2, X_3),\ R(X_3, X_2, X_1) \rightarrow \bot$. Then, c_2 generated by the second rule must be different from 0, and so we always obtain $S(0)$, and we get $\neg R(d_1, d_2, 0)$, for all d_1, d_2, by the last rule. ■

4 Translation into Guarded Fixed Point Logic

The EFWFS can be characterized in terms of *guarded fixed point logic*. This characterization turns out to be more convenient for reasoning about the EFWFS of guarded normal Datalog$^{\pm}$ programs.

Guarded fixed point logic (GFP), introduced by Grädel and Walukiewicz [13], simultaneously restricts and extends first-order logic by enforcing a certain quantification pattern, and allowing for inductively defining relations, while having a satisfiability problem of moderate complexity.

Let $\mathscr{R}$ be a relational schema. The set of formulas of GFP over $\mathscr{R}$ is built from atomic formulas over $\mathscr{R}$ (including equality atoms) using Boolean combinations, and the following two additional formula formation rules:

I. If α is an atomic formula over $\mathscr{R}$ containing the variables in $\mathbf{X}$, and ψ is a GFP formula over $\mathscr{R}$ whose free variables occur in α, then $\exists \mathbf{X}\,(\alpha \wedge \psi)$ and $\forall \mathbf{X}\,(\alpha \rightarrow \psi)$ are GFP formulas over $\mathscr{R}$. The formula α is called *guard*.

II. Let R be a k-ary predicate, $\mathbf{X}$ a k-tuple of variables, and $\psi(R, \mathbf{X})$ a GFP formula over $\mathscr{R} \cup \{R\}$ whose free variables occur in $\mathbf{X}$, and where R appears only positively (in the scope of an even number of negation symbols) and not in guards. Then, $[\mathbf{lfp}_{R,\mathbf{X}}\,\psi](\mathbf{X})$ and $[\mathbf{gfp}_{R,\mathbf{X}}\,\psi](\mathbf{X})$ are GFP formulas over $\mathscr{R}$ with free variables $\mathbf{X}$.

As for the semantics of the formulas in II, given a database D for $\mathscr{R}$, ψ defines an operator $F\colon \mathscr{P}(dom(D)^k) \rightarrow \mathscr{P}(dom(D)^k)$ with $F(S) := \{\mathbf{a} \mid D \models \psi(S, \mathbf{a})\}$. This operator is monotone and thus has a least fixed point $lfp(F)$ and a greatest fixed point $gfp(F)$. Then, $D \models [\mathbf{lfp}_{R,\mathbf{X}}\,\psi](\mathbf{a})$ iff $\mathbf{a} \in lfp(F)$, and $D \models [\mathbf{gfp}_{R,\mathbf{X}}\,\psi](\mathbf{a})$ iff $\mathbf{a} \in gfp(F)$.

Example 5 ([13]) The following GFP sentence says that the binary relation E is well-founded (i.e., no element is the endpoint of an infinite E-path): $\forall x, y\,(E(x, y) \rightarrow [\mathbf{lfp}_{W,x}$ $\forall y(E(y, x) \rightarrow W(y))](x))$. ■

We are now ready to describe the translation of guarded normal Datalog$^{\pm}$ into GFP. Let $P = D \cup \Sigma$ be a fixed guarded normal Datalog$^{\pm}$ program, where D is a database, and Σ is a set of guarded NTGDs. Without loss of generality, we assume that P contains only a single predicate R;[4] let k be its arity. We construct a GFP sentence $\mathsf{efwfs}(P)$ whose models closely correspond to the databases in EFWFS(P). The key is to "existentially quantify" all the instances of NTGDs that we use to compute the WFS, and to mimic the fixed-point definition of the WFS using those instances.

Technically, we represent instances of an NTGD $\sigma \in \Sigma$ using a $2k$-ary predicate S_σ. If $R(\mathbf{U})$ is the guard of σ, and $R(\mathbf{V})$ is its head, then a tuple $(\mathbf{a},\mathbf{b})$ in S_σ encodes the instance of σ obtained by substituting $\mathbf{a}$ and $\mathbf{b}$ for $\mathbf{U}$ and $\mathbf{V}$, respectively. For example, if σ is the NTGD $R(X,Y,0), \neg R(X,1,1) \rightarrow \exists Z\, R(Z,X,2)$, then the atom $S_\sigma((a,b,0),(c,a,2))$ represents the instance $R(a,b,0), \neg R(a,1,1) \rightarrow R(c,a,2)$ of σ.

We also use k-ary predicates T^*, C, T, F, and R, where T^* is intended to encode a superset of all true atoms; C holds all permutations of tuples in T^*; T and F respectively hold the set of all true atoms and the set of all false atoms (the latter relativized to C); and R corresponds to the predicate R of the initial database D. It is not hard to construct GFP sentences that enforce the desired interpretation of the predicates S_σ, T^*, C, and R. Based on those GFP sentences, it is then possible to construct GFP sentences ψ_T and ψ_F that enforce the desired interpretations of T and F. The sentence ψ_T basically does nothing else than defining the (set of all positive literals of the) least fixed point of $W_{P'}$, where P' consists of all instances of NTGDs in Σ represented by the tuples in the predicates S_σ, while ψ_F defines the greatest fixed point of a certain operator, yielding the greatest unfounded set $U_{P'}(T)$ (relativized to C).

For an NBCQ Q over $\{R\}$, let Q^* be the BCQ obtained by replacing every positive literal $R(\mathbf{X})$ in Q with $T(\mathbf{X})$, and every negative literal $\neg R(\mathbf{X})$ in Q with $F(\mathbf{X})$.

Lemma 6 *For all covered NBCQs Q over the schema of P, we have* EFWFS$(P) \models Q$ *iff* $\mathsf{efwfs}(P) \models Q^*$.

5 Complexity

We now take a look at the complexity of answering covered NBCQs over guarded normal Datalog$^{\pm}$ programs $P = D \cup \Sigma$. More precisely, we consider the *combined complexity*, measured in terms of the overall input, including P and the query.

In Section 4, we characterized the EFWFS of guarded normal Datalog$^{\pm}$ programs via a translation into guarded fixed point logic GFP. From the fact that satisfiability for GFP sentences is in 2-EXPTIME (and in EXPTIME in case of bounded arities) [13], we obtain the following:

Theorem 7 *Deciding* EFWFS$(P) \models Q$, *where $P = D \cup \Sigma$ is a guarded normal Datalog$^{\pm}$ program and Q a covered NBCQ, is 2-EXPTIME-complete in the general case, and EXPTIME-complete in the case of bounded arities and acyclic Q. Hardness holds even with respect to atomic queries.*

[4] It is an easy task to transform a guarded normal Datalog$^{\pm}$ program into this form, since multiple predicates can be simulated by constants and a single predicate. For example, $A(x,y) \wedge B(x)$ may be replaced by $R(a,x,y) \wedge R(b,x,x)$, where a and b are extra constant symbols.

We remark that Theorem 7 carries over to *unions of covered NBCQs*, that is, queries Q of the form $\bigvee_{i=1}^{n} Q_i$, where each Q_i is a covered NBCQ.

6 WFS for OWL 2 QL

All the DLs of the *DL-Lite* family in [8,22] and the DL $\mathscr{EL}$ [3] can be embedded into Datalog$^{\pm}$ [7]. In particular, this holds for *DL-Lite*$_{\mathscr{R}}$, which forms the theoretical basis of the QL profile of the Web ontology language OWL 2. Both our equality-friendly WFS (EFWFS) for normal Datalog$^{\pm}$ and the OWL 2 QL profile do not make the UNA. Our work in this paper thus paves the way for an extension of OWL 2 QL with nonmonotonic negation under the EFWFS.

The following definition extends *DL-Lite*$_{\mathscr{R}}$[5] (underlying the OWL 2 QL profile) and *DL-Lite*$_{\mathscr{R},\sqcap}$ [8,22], and $\mathscr{EL}$ [3] with nonmonotonic negation under the EFWFS.

Definition 8 Recall that a *DL-Lite*$_{\mathscr{R},\sqcap}$ knowledge base consists of a pair $(\mathscr{T},\mathscr{A})$, where the TBox $\mathscr{T}$ is a finite set of concept and role inclusion axioms $U_1 \sqcap \cdots \sqcap U_n \sqsubseteq V$, and the ABox $\mathscr{A}$ is a finite set of concept and role membership axioms $C(a)$ and $R(a,b)$, respectively. A *DL-Lite*$_{\mathscr{R},\sqcap,\text{not}}$ knowledge base $(\mathscr{T},\mathscr{A})$ consists of a finite set of inclusion axioms $\mathscr{T}$ and a finite set of membership axioms $\mathscr{A}$, where:

- Any *DL-Lite*$_{\mathscr{R},\sqcap}$ inclusion axiom is a *DL-Lite*$_{\mathscr{R},\sqcap,\text{not}}$ inclusion axiom.
- If $U_1 \sqcap \cdots \sqcap U_n \sqsubseteq V$ and $U'_1 \sqcap \cdots \sqcap U'_m \sqsubseteq V$ with $n,m > 0$ are both either concept or role inclusion axioms in *DL-Lite*$_{\mathscr{R},\sqcap}$, and V is positive (i.e., not of the form $V = \neg V'$), then $U_1 \sqcap \cdots \sqcap U_n \sqcap \text{not}\, U'_1 \sqcap \cdots \sqcap \text{not}\, U'_m \sqsubseteq V$ is a *DL-Lite*$_{\mathscr{R},\sqcap,\text{not}}$ concept or role inclusion axiom, respectively. Here, the U_i's and U'_i's contain no conjunction, and $\text{not}\, U'_i$ is the negation as failure (as opposed to the classical "$\neg$" in *DL-Lite*).
- For any concept A, any role R, and any individuals a,b, the expressions $A(a)$ and $R(a,b)$ are concept and role membership axioms, respectively.

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

Finally, we define $\mathscr{EL}_{\text{not}}$ as the extension of $\mathscr{EL}$ that allows formulas of the form $\text{not}\, C$ for atomic concepts $C = A$ and for concepts $C = \exists R.B$ to occur in top-level conjunctions on the left-hand side of TBox-axioms. ■

The semantics of *DL-Lite*$_{\mathscr{R},\text{not}}$ (resp., *DL-Lite*$_{\mathscr{R},\sqcap,\text{not}}$) is defined by translating a given *DL-Lite*$_{\mathscr{R},\text{not}}$ (resp., *DL-Lite*$_{\mathscr{R},\sqcap,\text{not}}$) knowledge base KB into a normal Datalog$^{\pm}$ program P_{KB} and by computing the well-founded semantics of P_{KB}. The details of the translation of *DL-Lite*$_{\mathscr{R},\text{not}}$ (resp., *DL-Lite*$_{\mathscr{R},\sqcap,\text{not}}$) into Datalog$^{\pm}$ are an extension of the translation of *DL-Lite*$_{\mathscr{R}}$ (resp., *DL-Lite*$_{\mathscr{R},\sqcap}$) given in [7]. Similarly, it is possible to translate $\mathscr{EL}_{\text{not}}$ into our formalism. Note that the $\text{sameAs}(a,b)$ and $\text{differentFrom}(a,b)$ constraints (specifying that the two individuals a and b are the same and different, respectively) that may be contained in a given knowledge base (over an ontology language without UNA) can be easily enforced by adding appropriate equalities $a = b$ and inequalities $\neg(a = b)$ to the guarded fixed point sentence $\mathsf{efwfs}(P_{KB})$.

[5] Note that although *DL-Lite*$_{\mathscr{R}}$ adopts the UNA, it actually does not require it, since making this assumption would have no impact on the semantic consequences of a *DL-Lite*$_{\mathscr{R}}$ ontology.

Example 9 (Holidays (cont'd)) Consider again Example 2. The two axioms (1) and (2) are translated into the following normal Datalog$^{\pm}$ rules:

$$p_{Dest}(X), p_{Swimming}(X), \neg p_{Beach}(X) \rightarrow \exists Y.p_{BB}(X,Y);$$
$$p_{Dest}(X), p_{Swimming}(X), \neg p^{\exists}_{BB}(X) \rightarrow p_{Beach}(X);$$
$$p_{BB}(X,Y) \rightarrow p^{\exists}_{BB}(X),$$

where $p^{\exists}_{BB}(X)$ is a new predicate for $\exists BeachBus$.

Suppose next that we are additionally given a database with holiday destinations $Dest(d_1)$, $Dest(d_2)$, and $Dest(d_3)$, which we want to be different, i.e., we assume that differentFrom(d_1,d_2), differentFrom(d_2,d_3), and differentFrom(d_1,d_3). We want to formalize the idea that any destination where one can swim should have a beach or a bus to a location with a beach — otherwise one has to take delight in walking. This can be achieved by considering the rules (1) and (2) along with the additional rule

$$Dest \sqcap \mathrm{not}\, Beach \sqcap \mathrm{not}\, \exists BeachBus \sqsubseteq WalkingOnly. \tag{3}$$

Furthermore, we may assume that at any place where one is not confined to only walking, one can also swim:

$$Dest \sqcap \mathrm{not}\, WalkingOnly \sqsubseteq Swimming. \tag{4}$$

Consider the following further facts: $WalkingOnly(d_1)$ and $BeachBus(d_2,d_3)$. Clearly, $WalkingOnly(d_1)$ implies that $Swimming(d_1)$ cannot be derived — because rule (4), the only rule that can derive $Swimming(d_1)$, requires the negation of $WalkingOnly(d_1)$ to be true. Thus, the WFS of the knowledge base includes not$Swimming(d_1)$. This is in contrast to what would happen if we interpreted not as "classical" negation: in the latter case, we could not derive anything from $WalkingOnly(d_1)$, as axiom (4) would trivially be satisfied for d_1. Other facts that are derived for d_1 are not$Beach(d_1)$ and not$\exists BeachBus(d_1)$ (= not$R(x,y)$ for any y), because the fact not$Swimming(d_1)$ implies that rules (1) and (2) cannot be used to derive $Beach(d_1)$ or $\exists BeachBus(d_1)$ (note that we use the fact that d_1 has to be different from d_2 and d_3, because of our ABox assumptions). Concerning the destination d_2, the WFS contains the following atoms: not$Beach(d_2)$, not$WalkingOnly(d_2)$, and $Swimming(d_2)$. ■

The complexity of answering covered NBCQs for any of our *DL-Lite*$_{\mathrm{not}}$ logics and for $\mathscr{EL}_{\mathrm{not}}$ can now be determined using Theorem 7. Note that Theorem 10 also yields immediate bounds for the complexity of standard DL problems such as instance and satisfiability checking.

Theorem 10 *Let $\mathscr{L}$ be any of DL-Lite$_{\mathscr{R},\sqcap,\mathrm{not}}$, DL-Lite$_{\mathscr{R},\mathrm{not}}$ or $\mathscr{EL}_{\mathrm{not}}$. Then, given a knowledge base $KB = (\mathscr{T},\mathscr{A})$ and an acyclic (resp., general) covered NBCQ Q, deciding whether Q is true under the EFWFS of KB is in EXPTIME (resp., 2-EXPTIME) for combined complexity.*

7 Related Work

There is already a substantial amount of work on combining rules and ontologies. The main direction of research so far has been to combine rules and ontologies into dl-programs consisting of a knowledge base together with a set of rules. This combination can be carried out in a loose or tight fashion. Representatives of the former are in particular the dl-programs in [10]. The hybrid MKNF knowledge bases in [21] are also close in spirit. Some representatives of tight integrations of rules and ontologies include the works by Rosati [23] and Lukasiewicz [18]. SWRL [15] and WRL [2] also belong to this category. For several of the above combinations of rules and ontologies, a well-founded semantics has been defined: [11], [19], [17], and [9] define a well-founded semantics for the loosely integrated dl-programs in [10], for the tightly integrated dl-programs in [18], for the hybrid MKNF knowledge bases in [21], and for an integration of rules and ontologies that is close in spirit to Rosati's approach [23], respectively.

We achieve the combination of rules and ontologies by a reduction from description logics to logic programming formalisms. Obviously our work is based on the earlier work on Datalog$^{\pm}$. Based on the same idea of translating ontologies into logic programming rules and hence closely related to our work are [1], [24], [14], and [16]. Probably the closest relationship to our work has the paper on FDNC-rules by [12] where the stable semantics is used in order to obtain a rule-based formalism with negation-as-failure that allows for the formulation of ontological knowledge.

8 Summary and Outlook

We have defined the equality-friendly WFS for Datalog with existentially quantified variables in rule heads and negations in rule bodies. Via a translation of its guarded fragment to *guarded fixed point logic*, we have then proved the decidability in the guarded case, and obtained complexity results for this case. These are important contributions in their own right. In addition, since the approach does not make the UNA, it can be readily used to extend DLs by nonmonotonic negation under the WFS, as illustrated along several DLs, including *DL-Lite*$_{\mathcal{R}}$, underlying the important OWL 2 QL profile.

Interesting topics for future research include to investigate the data complexity of guarded normal Datalog$^{\pm}$ and to explore how the approach can be extended by keys and to other ontology languages, including OWL 2 EL and OWL 2 RL.

Acknowledgments. This work was supported by the EPSRC grant EP/H051511/1 "ExODA: Integrating Description Logics and Database Technologies for Expressive Ontology-Based Data Access", by the European Research Council under the EU's 7th Framework Programme (FP7/2007-2013)/ERC grant 246858 (DIADEM), by the European Commission under the EU's FP7 grant 238975 (SEALS), and by a Yahoo! Research Fellowship. Georg Gottlob is a James Martin Senior Fellow, and also gratefully acknowledges a Royal Society Wolfson Research Merit Award. The work was carried out in the context of the James Martin Institute for the Future of Computing.

References

1. Alsaç, G., Baral, C.: Reasoning in description logics using declarative logic programming. Technical report, Dep. of Computer Science and Engineering, Arizona State Univ. (2001)

2. Angele, J., Boley, H., de Bruijn, J., Fensel, D., Hitzler, P., Kifer, M., Krummenacher, R., Lausen, H., Polleres, A., Studer, R.: Web Rule Language (WRL) (Sep 2005), W3C Member Submission. Available at http://www.w3.org/Submission/WRL/.
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathscr{EL}$ envelope. In: Proc. IJCAI-2005.
4. Baral, C., Subrahmanian, V.S.: Dualities between alternative semantics for logic programming and nonmonotonic reasoning. J. Autom. Reasoning 10(3), 399–420 (1993)
5. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP-1981.
6. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR-2008. Revised version: http://www.dbai.tuwien.ac.at/staff/gottlob/CGK.pdf.
7. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: J. Web Sem. (2012)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)
9. Drabent, W., Małuszyński, J.: Well-founded semantics for hybrid rules. In: Proc. RR-2007.
10. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artif. Intell. 172(12/13), 1495–1539 (2008)
11. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Well-founded semantics for description logic programs in the Semantic Web. In: Proc. RuleML-2004.
12. Eiter, T., Simkus, M.: FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. ACM Trans. Comput. Log. 11(2) (2010)
13. Grädel, E., Walukiewicz, I.: Guarded fixed point logic. In: Proc. LICS-1999.
14. Heymans, S., Vermeir, D.: Integrating Semantic Web reasoning and answer set programming. In: Proc. ASP-2003.
15. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web rule language combining OWL and RuleML (May 2004), W3C Member Submission. Available at http://www.w3.org/Submission/SWRL/.
16. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ-description logic to disjunctive Datalog programs. In: Proc. KR-2004.
17. Knorr, M., Alferes, J.J., Hitzler, P.: Local closed world reasoning with description logics under the well-founded semantics. Artif. Intell. 175(9/10), 1528–1554 (2011)
18. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the Semantic Web. In: Proc. ESWC-2007.
19. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the Semantic Web. IEEE Trans. Knowl. Data Eng. 22(11), 1577–1592 (2010)
20. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and logic programming live together happily ever after? In: Proc. ISWC-2006.
21. Motik, B., Rosati, R.: A faithful integration of description logics with logic programming. In: Proc. IJCAI-2007.
22. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
23. Rosati, R.: $\mathscr{DL}$+*log*: Tight integration of description logics and disjunctive Datalog. In: Proc. KR-2006.
24. Swift, T.: Deduction in ontologies via ASP. In: Proc. LPNMR-2004.
25. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM 38(3), 620–650 (1991)