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

Georg Gottlob[1,2] **André Hernich**[3] **Clemens Kupke**[1] **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 for Datalog rules with existentially quantified variables in their heads and negations in their bodies. In particular, we provide a well-founded semantics (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 $\mathcal{EL}$ and of members of the *DL-Lite* family, as well as corresponding complexity results for query answering.

Introduction

The recent Datalog$^{\pm}$ family of ontology languages (Calì, Gottlob, and Lukasiewicz 2012) 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) A DL knowledge base consists of a TBox and an ABox. For example, the knowledge that every conference paper is an article and that every scientist is the author of at least one paper can be expressed by the two axioms $\mathit{ConferencePaper} \sqsubseteq \mathit{Article}$ and $\mathit{Scientist} \sqsubseteq \exists \mathit{isAuthorOf}$ in the TBox, respectively, while the knowledge that John is a scientist can be expressed by the axiom $\mathit{Scientist}(\mathit{john})$ in the ABox. In Datalog$^{\pm}$, the former are encoded as the rules $\mathit{ConferencePaper}(X) \rightarrow \mathit{Article}(X)$ and $\mathit{Scientist}(X) \rightarrow \exists Y\ \mathit{isAuthorOf}(X, Y)$, respectively, and the latter is encoded by an identical fact in the database. Furthermore, the TBox axiom $\mathit{ConferencePaper} \sqsubseteq \neg \mathit{JournalPaper}$, encoding 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 Datalog 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 (Calì, Gottlob, and Lukasiewicz 2012), 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 (van Gelder, Ross, and Schlipf 1991) 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 $\mathcal{EL}$ and *DL-Lite*$_{\mathcal{R}}$ (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 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 $\mathrm{not}\, Beach$ and $\mathrm{not}\exists BeachBus$ hold. This can be achieved by the following two rules:

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

$$Dest \sqcap Swimming \sqcap \mathrm{not}\exists BeachBus \sqsubseteq Beach. \quad (2)$$

Observe also that $\mathrm{not}\, Swimming(d)$ would immediately imply $\mathrm{not}\, Beach(d)$ and $\mathrm{not}\exists BeachBus(d)$, 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: $Employee(John)$, $Employee(Sam)$. To these axioms, we add a concept inclusion axiom that maintains that every employee must have an employer: $Employee \sqsubseteq \exists.hasEmployer$. Finally, we would like to test whether or not *John* and *Sam* work in the same company, which is expressed by the query $\exists x(hasEmployer(John, x) \sqcap \mathrm{not}\, hasEmployer(Sam, x))$. 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. ■

Preliminaries

In this section, we briefly recall some basics on Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012), namely, on relational databases, conjunctive queries (CQs), Boolean conjunctive queries (BCQs), tuple-generating dependencies (TGDs), negative constraints, and normal TGDs and BCQs.

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

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

Tuple-Generating Dependencies (TGDs). Tuple-generating dependencies (TGDs) describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads; their syntax and semantics are as follows. Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there is a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, 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 $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable (Beeri and Vardi 1981), even with fixed schema and TGDs (Calì, Gottlob, and Kifer 2008).

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 $\mathcal{R}$, a *normal TGD (NTGD)* has the form $\forall\mathbf{X}\forall\mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,\Psi(\mathbf{X},\mathbf{Z})$, where $\Phi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathcal{R}$, and $\Psi(\mathbf{X},\mathbf{Z})$ is a conjunction of atoms over $\mathcal{R}$. It is also abbreviated as $\Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,\Psi(\mathbf{X},\mathbf{Z})$. As for standard TGDs, w.l.o.g., $\Psi(\mathbf{X},\mathbf{Z})$ is a singleton atom. We denote by $head(\sigma)$ the atom in the head of σ, and by $body^+(\sigma)$ and $body^-(\sigma)$ the sets of all positive and negative ("¬"-free) atoms in the body of σ, respectively. We say σ is *guarded* iff it contains a positive body atom, denoted $guard(\sigma)$, that contains all universally quantified variables of σ. W.l.o.g., constants in the body of guarded σ occur only in guards. We say σ is *linear* iff σ is guarded and has exactly one positive atom in its body.

As for the semantics, a normal TGDs σ is *satisfied* in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h for all the variables and constants in the body of σ that maps (i) all atoms of $body^+(\sigma)$ to atoms of D and (ii) no atom of $body^-(\sigma)$ to atoms of D, then there exists an extension h' of h that maps all atoms of $head(\sigma)$ to atoms of D.

We next add negation to BCQs as follows. 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 $\mathbf{X}$. We denote by Q^+ (resp., Q^-) the set of all positive (resp., negative ("¬"-free)) atoms of Q. We say Q is *safe* iff every variable in a negative atom in Q also occurs in a positive atom in Q. We say Q is *covered* iff for every negative atom α in Q, there exists a positive atom β in Q such that every argument in α occurs in β. Observe that the coveredness of Q implies also the safeness of Q, but not vice versa. In the sequel, w.l.o.g., BCQs contain no constants.

Note that normal non-Boolean CQs over finite databases can be reduced to constant-free NBCQs, by first reducing them to NBCQs with constants, and then moving their constants into the TGDs by introducing new predicate symbols.

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

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

Syntax of Normal Programs. Let Φ be a first-order vocabulary with nonempty finite sets of constant, function, and predicate symbols. Let $\mathcal{V}$ be a set of variables. A *term* is either a variable from $\mathcal{V}$, a constant symbol from Φ, or of the form $f(t_1,\ldots,t_n)$, where f is a function symbol of arity $n \geqslant 0$ from Φ, and $t_1,\ldots,t_n$ are terms. An *atom* is of the form $p(t_1,\ldots,t_n)$, where p is a predicate symbol of arity $n \geqslant 0$ from Φ, and $t_1,\ldots,t_n$ are terms. A *literal* l is an atom p or a negated atom $\neg p$. A *normal rule* (or simply *rule*) r is of the form

$$\beta_1,\ldots,\beta_n,\neg\beta_{n+1},\ldots,\neg\beta_{n+m} \rightarrow \alpha\,, \tag{3}$$

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 $\beta_1,\ldots,\beta_n,\neg\beta_{n+1},\ldots,\neg\beta_{n+m}$ is its *body*. We define $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{\beta_1,\ldots,\beta_n\}$ and $B^-(r) = \{\beta_{n+1},\ldots,\beta_{n+m}\}$. A rule of the form (3) with $m = n = 0$ is also called a *fact*. A *normal program* P is a finite set of normal rules (3).

WFS of Normal Programs. The WFS (van Gelder, Ross, and Schlipf 1991) is 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. The WFS of normal programs P has many different equivalent definitions (see also (Baral and Subrahmanian 1993)). We recall here the one based on unfounded sets, via the operators U_P, T_P, and W_P.

We first give some preliminary definitions. The *Herbrand universe* of a normal program P, denoted HU_P, is the set of all terms constructed from constant and function symbols appearing in P. If there is no such constant symbol, then we take an arbitrary constant symbol from Φ. As usual, terms, atoms, literals, rules, programs, etc. are *ground* iff they do not contain any variables. The *Herbrand base* of a normal program P, denoted HB_P, is the set of all ground atoms that can be constructed from the predicate symbols appearing in P and the ground terms in HU_P. A *ground instance* of a rule $r \in P$ is obtained from r by uniformly replacing every variable that occurs in r by a ground term from HU_P. We denote by $ground(P)$ the set of all ground instances of rules in P. For literals $\ell = a$ (resp., $\ell = \neg a$), we use $\neg.\ell$ to denote $\neg a$ (resp., a), and for sets of literals S, we define $\neg.S = \{\neg.\ell \mid \ell \in S\}$ and $S^+ = \{a \in S \mid a \text{ is an atom}\}$. We denote by $Lit_P = HB_P \cup \neg.HB_P$ the set of all ground literals with predicate symbols from P and ground terms from HU_P. A set of ground literals $S \subseteq Lit_P$ is *consistent* iff $S \cap \neg.S = \emptyset$. A *(three-valued) interpretation* relative to P is any consistent set of ground literals $I \subseteq Lit_P$.

We next define the notion of an unfounded set. 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 $r \in ground(P)$ 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 consistent $I \subseteq Lit_P$ by:

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

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 normal programs, and define their equality-friendly WFS (EFWFS) as the set of WFS 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* $\mathcal{I}$ 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 $\mathcal{I}$. Let $\mathcal{I}(P)$ 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 $\{\text{WFS}(P'') \mid \mu\colon dom(P) \rightarrow \Delta, P'' \in \mathcal{I}(P_\mu)\}$. If there are additionally negative constraints in P, then the equality-friendly well-founded semantics comprises all elements in the above set that satisfy the constraints.

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.

Translation into Guarded Fixed Point Logic

The definition of the EFWFS refers to an infinite set of programs, which is rather inconvenient for reasoning tasks. We now characterize the EFWFS in terms of *guarded fixed point logic*, which is more suitable.

A Briefing on Guarded Fixed Point Logic. *Guarded fixed point logic (GFP)*, introduced by Grädel and Walukiewicz (1999), 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 $\mathcal{R}$ be a relational schema. The set of formulas of GFP over $\mathcal{R}$ is built from atomic formulas over $\mathcal{R}$ (including equality atoms) using Boolean combinations, and the following two additional formula formation rules:

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

II. Let R be a k-ary predicate, and $\mathbf{X}$ a k-tuple of variables. Let $\psi(R,\mathbf{X})$ be a GFP formula over $\mathcal{R} \cup \{R\}$ 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 $\mathcal{R}$ with free variables $\mathbf{X}$.

As for the semantics of the formulas in II, given a database D for $\mathcal{R}$, ψ defines an operator $F\colon 2^{dom(D)^k} \rightarrow 2^{dom(D)^k}$ 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 *(Grädel and Walukiewicz 1999)* The following GFP formula says that binary relation E is well-founded, i.e., no element is the endpoint of an infinite E-path: $\forall x,y\,\big(E(x,y) \rightarrow \big[\mathbf{lfp}_{W,x}\,\forall y\big(E(y,x) \rightarrow W(y)\big)\big]\,(x)\big)$.

The Translation. We are now ready to specify the translation of guarded normal Datalog$^{\pm}$ into GFP. Throughout this section, 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;[1] 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" the instances of NTGDs that we use to compute the WFS, and to mimic the fixed-point definition of the WFS using those instances. To encode all the instances, we use $2k$-ary predicates S_σ, for each $\sigma \in \Sigma$; here, $S_\sigma(\mathbf{a},\mathbf{b})$ encodes the instance of σ with guard atom $R(\mathbf{a})$ and head $R(\mathbf{b})$. We also use k-ary predicates T^*, C,

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

T, and F, where T^* is intended to encode a superset of all true atoms, C is intended to contain all tuples *covered* by a tuple in T^* (i.e., all tuples $\mathbf{a}$ such that $\mathbf{a} \subseteq \mathbf{b}$ for some tuple $\mathbf{b}$ in T^*), and T, F are intended to encode the set of all true atoms, and all false atoms, respectively.

To enforce the desired interpretations of the predicates S_σ, $\mathsf{efwfs}(P)$ contains the formula $\psi_S := \bigwedge_{\sigma\in\Sigma} \psi_{S,\sigma}$. If $\sigma = \forall\mathbf{X}\mathbf{Y}(\phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\, R(\mathbf{V}))$, and $R(\mathbf{U})$ is its guard, then $\psi_{S,\sigma}$ is the formula $\forall\mathbf{X}\mathbf{Y}(T^*(\mathbf{U}) \rightarrow \exists\mathbf{Z}\, S_\sigma(\mathbf{U},\mathbf{V})) \wedge \forall\mathbf{X}\mathbf{Y}\mathbf{Z}(S_\sigma(\mathbf{U},\mathbf{V}) \rightarrow T^*(\mathbf{V}))$.[2] To ensure that T^* contains at least all atoms of the database D, we add the formula $\psi_D := \bigwedge_{R(\mathbf{a})} T^*(\mathbf{a})$. To enforce the desired interpretation of C, we add a formula ψ_C which states that C contains all tuples of T^*, and for every tuple $(a_1,\ldots,a_k)$ in C also the tuples $(a_2,\ldots,a_k,a_1)$ and $(a_2,a_1,a_3,\ldots,a_k)$. Finally, we add formulas $\psi_T := \forall\mathbf{W}(T^*(\mathbf{W}) \rightarrow (T(\mathbf{W}) \leftrightarrow \xi_T(\mathbf{W})))$ and $\psi_F := \forall\mathbf{W}(C(\mathbf{W}) \rightarrow (F(\mathbf{W}) \leftrightarrow \xi_F(\mathbf{W})))$, where $\mathbf{W}$ is a k-tuple of distinct variables that do not appear in Σ, to enforce that T and F are interpreted by all true and false atoms, respectively. The formulas $\xi_T(\mathbf{W})$ and $\xi_F(\mathbf{W})$ define the set of true and false atoms, respectively, and will be specified below. Note that we include in F only the "false" tuples from C (tuples that are not in C are false anyway). Furthermore, to enforce that T is a "subset" of T^*, we include the formula $\forall\mathbf{W}(T(\mathbf{W}) \rightarrow T^*(\mathbf{W}))$.

Our definition of $\xi_T(\mathbf{W})$ and $\xi_F(\mathbf{W})$ uses the following formula $\xi'_F(\mathbf{W})$. Informally, $\xi'_F(\mathbf{a})$ means $\mathbf{a} \in U_{P'}(I)$, where P' is a program compatible with our choice of instances of NTGDs, and I is the set of all atoms $R(\mathbf{a})$ with $\mathbf{a} \in T'$ (T' is the k-ary predicate used in ψ_T; think of I as a set of atoms already derived as true). [3] We let $\xi'_F(\mathbf{W}) := [\mathbf{gfp}_{F',\mathbf{W}}\, \delta_F](\mathbf{W})$, where $\delta_F(F',\mathbf{W}) := \bigwedge_{\sigma\in\Sigma} \delta_{F,\sigma}$. Here, if $\sigma = \forall\mathbf{X}\mathbf{Y}(\phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\, R(\mathbf{V}))$, and $R(\mathbf{U})$ is its guard, we let $\delta_{F,\sigma}(F',\mathbf{W})$ be[4]

$$\bigwedge_{\substack{1\le i\le k\\ V_i\in\Delta}} W_i{=}V_i \wedge \bigwedge_{\substack{1\le i<j\le k\\ V_i=V_j}} W_i{=}W_j \rightarrow \forall\mathbf{Y}\big(S_\sigma(\mathbf{U},\mathbf{V}) \rightarrow \chi\big),$$

where χ is obtained from the body of σ by replacing conjunctions with disjunctions, by replacing every positive literal $R(\mathbf{X}')$ with $F'(\mathbf{X}')$, and every negative literal $\neg R(\mathbf{X}')$ with $T'(\mathbf{X}')$. To simplify the presentation, we assume above that the variables in σ have been renamed so that if V_i is a variable, then $V_i = W_i$.

We obtain $\xi_F(\mathbf{W})$ from $\xi'_F(\mathbf{W})$ by replacing the predicate T' with T. Furthermore, $\xi_T(\mathbf{W}) := [\mathbf{lfp}_{T',\mathbf{W}}\, \delta_T](\mathbf{W})$, where $\delta_T(\mathbf{W})$ mimics the operator $T_{P'}$ with P' as above. Using the formula $\xi'_F(\mathbf{W})$ to check whether a tuple is false in a certain stage of the fixed-point iteration, it is now an easy exercise to define $\delta_T(\mathbf{W})$ as desired. This concludes the description of $\mathsf{efwfs}(P)$.

[2]Note that both $\mathbf{U}$ and $\mathbf{V}$ may contain constants; therefore we quantify over $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ instead. Note also that $\mathbf{U}$ contains all variables from $\mathbf{X}, \mathbf{Y}$, and that $\mathbf{V}$ contains all variables from $\mathbf{X}, \mathbf{Z}$.

[3]$U_{P'}(I)$ can be characterized by the greatest fixed point of a certain operator $F_{P',I}$; see (van Gelder, Ross, and Schlipf 1991). $\xi'_F(\mathbf{a})$ just checks whether $R(\mathbf{a})$ occurs in this greatest fixed point.

[4]$\mathbf{V}_i$ denotes the i-th component of $\mathbf{V}$; similarly with $\mathbf{W}$.

Lemma 6 *For all sets T and F of atoms over $\{R\}$, the following are equivalent:*

- $T \cup \neg.F$ *is in* EFWFS(P).
- *There is a model M of* $\mathsf{efwfs}(P)$ *such that* $T = \{R(\mathbf{a}) \mid T(\mathbf{a}){\in}M\}$ *and* $F{=}\{R(\mathbf{a}) \mid F(\mathbf{a}) \in M$ *or* $C(\mathbf{a}) \notin M\}$.

Proof (sketch). The direction from top to bottom is easy. For the other direction, let M be a model of $\psi_D \wedge \psi_S \wedge \psi_C$. We define an instance $P(M)$ of Σ as follows. Let $\sigma = \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\, R(\mathbf{V})$ be a TGD in Σ with guard $R(\mathbf{U})$, and $h\colon \mathbf{U} \rightarrow \Delta$. For all extensions h' of h with $h'(S_\sigma(\mathbf{U},\mathbf{V})) \in M$, add the instance $\Phi(h'(\mathbf{X}), h'(\mathbf{Y})) \rightarrow R(h'(\mathbf{V}))$ to $P(M)$. If there is no such extension, choose an arbitrary extension h' of h, and add $\Phi(h'(\mathbf{X}), h'(\mathbf{Y})) \rightarrow R(h'(\mathbf{V}))$ to $P(M)$. It is now not hard to verify that, for $T := \{R(\mathbf{a}) \mid T'(\mathbf{a}) \in M\}$ and $F := \{R(\mathbf{a}) \mid M \models \xi'_F(\mathbf{a})\}$, we have $F = U_{P(M)}(T)$. Using this, it it straightforward to prove that, with $T' := \{R(\mathbf{a}) \mid M \models \xi_T(\mathbf{a})\}$ and $F' := \{R(\mathbf{a}) \mid M \models \xi_F(\mathbf{a})\}$, it holds that $T' \cup \neg.F' =$ WFS$(P(M))$. This is enough to establish the lemma. □

For an NBCQ Q over $\{R\}$, let Q^* be the BCQ over $\{T, F\}$ 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 yields:

Corollary 7 *For all covered NBCQs Q over the schema of P, we have* EFWFS$(P) \models Q$ *if and only if* $\mathsf{efwfs}(P) \models Q^*$.

A similar result also holds for *unions of covered NBCQs*, i.e., queries $\bigvee_{i=1}^n Q_i$, where each Q_i is a covered NBCQ. In that case, we would define Q^* to be $\bigvee_{i=1}^n Q_i^*$.

Complexity Results

Let us call an NBCQ Q *acyclic* if the BCQ obtained from Q by dropping all negative literals is acyclic.

Theorem 8 *Deciding* EFWFS$(P) \models Q$*, where $P = D \cup \Sigma$ is a guarded normal Datalog$^\pm$ program and Q an acyclic covered NBCQ, is 2-EXPTIME-complete in the general case, and EXPTIME-complete in case of bounded arities. Hardness holds even with respect to atomic queries.*

Proof (sketch). The upper bounds follow from Corollary 7 and (Grädel and Walukiewicz 1999). First, we transform Q into a BCQ Q^*, as described immediately before Corollary 7. Since Q is acyclic and covered, it follows that Q^* is acyclic. Hence, it is possible to turn Q^* into a GFP formula ψ_{Q^*}. Moreover, by Corollary 7, we have EFWFS$(P) \models Q$ iff $\mathsf{efwfs}(P) \wedge \neg\psi_{Q^*}$ is satisfiable. Since by (Grädel and Walukiewicz 1999), satisfiability for GFP sentences is in 2-EXPTIME, and in EXPTIME in case of bounded arities, the result follows. We remark that the result in (Grädel and Walukiewicz 1999) holds only for GFP sentences without constants, but using the techniques in (Grädel 1999, Lemma 2.3), for the general case, and in (ten Cate and Franceschet 2005, Proposition 5), for the case of bounded arities, we can reduce the case of GFP sentences with constants to the constant-free case.

Hardness follows from the corresponding hardness results for guarded Datalog$^\pm$ (Calì, Gottlob, and Lukasiewicz

2012). Just note that, if P does not contain negation, we have EFWFS$(P) \models Q$ iff $P \models Q$. □

For covered NBCQs which are not necessarily acyclic, we can prove the following:

Theorem 9 *Deciding* EFWFS$(P) \models Q$*, where* $P = D \cup \Sigma$ *is a guarded normal Datalog*$^{\pm}$ *program and* Q *is a covered NBCQ, is 2-EXPTIME-complete.*

Proof (sketch). Let Q be a covered NBCQ. As before, we transform Q into a BCQ Q^*. Now let χ be the disjunction of all acyclic BCQs T such that $T \models Q^*$, and T has at most three times as many atoms than Q^*. In (Bárány, Gottlob, and Otto 2010), it is shown that for all GFP sentences ψ without fixed point operators, we have $\psi \models Q^*$ iff $\psi \models \chi$. Their proof readily extends to GFP sentences with fixed point operators, so that, if we take ψ to be $\mathsf{efwfs}(P)$, we obtain $\mathsf{efwfs}(P) \models Q^*$ iff $\mathsf{efwfs}(P) \models \chi$. Since all disjuncts in χ are acyclic, χ can easily be turned into a GFP sentence χ'. It follows that we can check EFWFS$(P) \models Q$ by checking whether the GFP sentence $\mathsf{efwfs}(P) \wedge \neg\chi'$ is unsatisfiable, which, by (Grädel and Walukiewicz 1999), is exponential in the size of $\mathsf{efwfs}(P) \wedge \neg\chi'$, and doubly exponential in the maximum arity of a predicate symbol. Since the length of χ' is only exponential in the length of Q (Bárány, Gottlob, and Otto 2010), the claim follows. □

We remark that the results of this section carry over to unions of covered NBCQs (cf. the remark below Corollary 7 for a definition of unions of covered NBCQs, and how to define the query Q^* in that case).

WFS for OWL 2 QL

All the DLs of the *DL-Lite* family in (Calvanese et al. 2007; Poggi et al. 2008) and the DL $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) can be embedded into Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2012). In particular, this holds for *DL-Lite*$_{\mathcal{R}}$, 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*$_{\mathcal{R}}$[5] (which is underlying the OWL 2 QL profile) and *DL-Lite*$_{\mathcal{R},\sqcap}$ (Calvanese et al. 2007; Poggi et al. 2008), and $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) with nonmonotonic negation under the EFWFS.

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

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

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

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

The semantics of *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) is defined by translating a given *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) knowledge base KB into a normal Datalog$^{\pm}$ program P_{KB} and by computing the well-founded semantics of P_{KB}. The details of the translation of *DL-Lite*$_{\mathcal{R},\mathrm{not}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap,\mathrm{not}}$) into Datalog$^{\pm}$ are an extension of the translation of *DL-Lite*$_{\mathcal{R}}$ (resp., *DL-Lite*$_{\mathcal{R},\sqcap}$) given in (Calì, Gottlob, and Lukasiewicz 2012). Similarly, it is possible to translate $\mathcal{EL}_{\mathrm{not}}$ into our formalism. Note that the $\mathrm{sameAs}(a, b)$ and $\mathrm{differentFrom}(a, b)$ 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 $\mathsf{efwfs}(P_{KB})$.

Example 11 (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 $\mathrm{differentFrom}(d_1, d_2)$, $\mathrm{differentFrom}(d_2, d_3)$, and $\mathrm{differentFrom}(d_1, d_3)$. 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

$$Dest \sqcap \mathrm{not}\, Beach \sqcap \mathrm{not} \exists BeachBus \sqsubseteq WalkingOnly. \quad (4)$$

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

$$Dest \sqcap \mathrm{not}\, WalkingOnly \sqsubseteq Swimming. \quad (5)$$

[5] Note that although *DL-Lite*$_{\mathcal{R}}$ 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*$_{\mathcal{R}}$ ontology.

Consider the following further facts about the destinations: $WalkingOnly(d_1)$; $BeachBus(d_2, d_3)$. Clearly, $WalkingOnly(d_1)$ implies that $Swimming(d_1)$ cannot be derived — because rule (5), the only one that can be used to derive $Swimming(d_1)$, requires the negation of $WalkingOnly(d_1)$ to be true. Thus, the WFS of the knowledge base includes $\mathrm{not}\, Swimming(d_1)$. 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 (5) would trivially be satisfied for d_1. Other facts that are derived for d_1 are $\mathrm{not}\, Beach(d_1)$ and $\mathrm{not}\, \exists BeachBus(d_1)$ (= $\mathrm{not}\, R(x, y)$ for any y), because the fact $\mathrm{not}\, Swimming(d_1)$ 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 destination d_2, the WFS contains the following atoms: $\mathrm{not}\, Beach(d_2); \mathrm{not}\, WalkingOnly(d_2); Swimming(d_2)$. ■

The complexity of answering covered NBCQs for any of our *DL-Lite*$_{\mathrm{not}}$ logics and for $\mathcal{EL}_{\mathrm{not}}$ can now be determined using Theorem 8. Note that Theorem 12 also yields immediate bounds for the complexity of standard DL problems such as instance and satisfiability checking.

Theorem 12 *Let $\mathcal{L}$ be any of DL-Lite$_{\mathcal{R},\sqcap,\mathrm{not}}$, DL-Lite$_{\mathcal{R},\mathrm{not}}$ or $\mathcal{EL}_{\mathrm{not}}$. Then, given a knowledge base $KB = (\mathcal{T}, \mathcal{A})$ and an acyclic (resp., general) covered NBCQ Q, deciding whether Q is true under the EFWFS of KB is in EXPTIME (resp., 2EXPTIME) for combined complexity.*

Related Work

To our knowledge, there exists no closely related approach to the WFS in DLs to date. But 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 loose or tight fashion. Representatives of the former are in particular the dl-programs in (Eiter et al. 2008) and their extension to HEX-programs (Eiter et al. 2005). The combination of defeasible reasoning with DLs in (Antoniou 2002), the calls to DL reasoners in TRIPLE (Sintek and Decker 2002), and the hybrid MKNF KBs in (Motik et al. 2006; Motik and Rosati 2007) are also close in spirit. Some representatives of tight integrations of rules and ontologies are in particular the works due to Donini et al. (1998), Levy and Rousset (1998), Grosof et al. (2003), Motik, Sattler, and Studer (2005), Heymans, Nieuwenborgh, and Vermeir (2005), and Rosati (2005; 2006). SWRL (Horrocks et al. 2004) and WRL (Angele et al. 2005) also belong to this category. For several of the above combinations of rules and ontologies, a well-founded semantics has been defined; more specifically, Eiter et al. (2004), Knorr, Alferes, and Hitzler (2011), and Drabent and Małuszyński (2007) define a well-founded semantics for the loosely integrated dl-programs in (Eiter et al. 2008), for the hybrid MKNF knowledge bases in (Motik et al. 2006; Motik and Rosati 2007), and for an integration of rules and ontologies that is close in spirit to Rosati's approach (2005; 2006), 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 the works by Alsaç and Baral (2001), Swift (2004), Heymans and Vermeir (2003), and Hustadt, Motik, and Sattler (2004). Probably the closest relationship to our work has the paper on FDNC rules by Eiter and Simkus (2010), where the stable (rather than the well-founded) semantics is used in order to obtain a rule-based formalism with negation-as failure that allows for the formulation of ontological knowledge. FDNC rules are syntactically restricted to ensure a forest-shaped model property.

Note that approaches to defining nonmonotonic DLs, such as the one in (Bonatti, Lutz, and Wolter 2009), which is based on circumscription, are less closely related.

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 several 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, which we have illustrated along several DLs, including *DL-Lite*$_{\mathcal{R}}$, which is 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 under the 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, 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. We thank the reviewers of this paper for their useful comments, which have helped to improve this work.

References

Alsaç, G., and Baral, C. 2001. Reasoning in description logics using declarative logic programming. Technical report, Department of Computer Science and Engineering, Arizona State University.

Angele, J.; Boley, H.; de Bruijn, J.; Fensel, D.; Hitzler, P.; Kifer, M.; Krummenacher, R.; Lausen, H.; Polleres, A.; and Studer, R. 2005. Web Rule Language (WRL).

W3C Member Submission. Available at http://www.w3.org/Submission/WRL/.

Antoniou, G. 2002. Nonmonotonic rule systems on top of ontology layers. In *Proc. ISWC-2002*, volume 2342 of *LNCS*, 394–398. Springer.

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

Baral, C., and Subrahmanian, V. S. 1993. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. Autom. Reasoning* 10(3):399–420.

Bárány, V.; Gottlob, G.; and Otto, M. 2010. Querying the guarded fragment. In *Proc. LICS-2010*, 1–10. IEEE Computer Society.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Proc. ICALP-1981*, volume 115 of *LNCS*, 73–85. Springer.

Bonatti, P. A.; Lutz, C.; and Wolter, F. 2009. The complexity of circumscription in DLs. *J. Artif. Intell. Res.* 35:717–773.

Calì, A.; Gottlob, G.; and Kifer, M. 2008. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. KR-2008*, 70–80. AAAI Press. Revised version: http://www.dbai.tuwien.ac.at/staff/gottlob/CGK.pdf.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. In *J. Web Sem.* In press.

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

Donini, F. M.; Lenzerini, M.; Nardi, D.; and Schaerf, A. 1998. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. Intell. Inf. Syst.* 10(3):227–252.

Drabent, W., and Małuszyński, J. 2007. Well-founded semantics for hybrid rules. In *Proc. RR-2007*, volume 4524 of *LNCS*, 1–15. Springer.

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

Eiter, T.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2004. Well-founded semantics for description logic programs in the Semantic Web. In *Proc. RuleML-2004*, volume 3323 of *LNCS*, 81–97. Springer.

Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2005. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proc. IJCAI-2005*, 90–96. Professional Book Center.

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

Grädel, E., and Walukiewicz, I. 1999. Guarded fixed point logic. In *Proc. LICS-1999*, 45–54. IEEE Computer Society.

Grädel, E. 1999. On the restraining power of guards. *J. Symb. Log.* 64(4):1719–1742.

Grosof, B. N.; Horrocks, I.; Volz, R.; and Decker, S. 2003. Description logic programs: Combining logic programs with description logics. In *Proc. WWW-2003*, 48–57. ACM Press.

Heymans, S., and Vermeir, D. 2003. Integrating semantic web reasoning and answer set programming. In *Proc. ASP-2003*, volume 78 of *CEUR Workshop Proceedings*, 194–208. CEUR-WS.org.

Heymans, S.; Nieuwenborgh, D. V.; and Vermeir, D. 2005. Nonmonotonic ontological and rule-based reasoning with extended conceptual logic programs. In *Proc. ESWC-2005*, volume 3532 of *LNCS*, 392–407. Springer.

Horrocks, I.; Patel-Schneider, P. F.; Boley, H.; Tabet, S.; Grosof, B.; and Dean, M. 2004. SWRL: A Semantic Web rule language combining OWL and RuleML. W3C Member Submission. Available at http://www.w3.org/Submission/SWRL/.

Hustadt, U.; Motik, B.; and Sattler, U. 2004. Reducing $\mathcal{SHIQ}^-$-description logic to disjunctive Datalog programs. In *Proc. KR-2004*, 152–162. AAAI Press.

Knorr, M.; Alferes, J. J.; and Hitzler, P. 2011. Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.* 175(9/10):1528–1554.

Levy, A. Y., and Rousset, M.-C. 1998. Combining Horn rules and description logics in CARIN. *Artif. Intell.* 104(1/2):165–209.

Motik, B., and Rosati, R. 2007. A faithful integration of description logics with logic programming. In *Proc. IJCAI-2007*, 477–482. AAAI Press/IJCAI.

Motik, B.; Horrocks, I.; Rosati, R.; and Sattler, U. 2006. Can OWL and logic programming live together happily ever after? In *Proc. ISWC-2006*, volume 4273 of *LNCS*, 501–514. Springer.

Motik, B.; Sattler, U.; and Studer, R. 2005. Query answering for OWL-DL with rules. *J. Web Sem.* 3(1):41–60.

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

Rosati, R. 2005. On the decidability and complexity of integrating ontologies and rules. *J. Web Sem.* 3(1):61–73.

Rosati, R. 2006. $\mathcal{DL}+log$: Tight integration of description logics and disjunctive Datalog. In *Proc. KR-2006*, 68–78. AAAI Press.

Sintek, M., and Decker, S. 2002. TRIPLE - A query, inference, and transformation language for the Semantic Web. In *Proc. ISWC-2002*, volume 2342 of *LNCS*, 364–378. Springer.

Swift, T. 2004. Deduction in ontologies via ASP. In *Proc. LPNMR-2004*, volume 2923 of *LNCS*, 275–288. Springer.

ten Cate, B., and Franceschet, M. 2005. Guarded fragments with constants. *J Logic Lang. Inform.* 14(3):281–288.

van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *J. ACM* 38(3):620–650.