# Well-Founded Semantics for Extended Datalog and Ontological Reasoning

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

[1] UC Santa Cruz, USA & Humboldt-Universität zu Berlin, Germany
hernich@informatik.hu-berlin.de
[2] University of Strathclyde, Glasgow, Scotland, UK
clemens.kupke@strath.ac.uk
[3] University of Oxford, UK
firstname.lastname@cs.ox.ac.uk

**Abstract.** The Datalog$^\pm$ family of expressive extensions of Datalog has recently been introduced as a new paradigm for query answering over ontologies, which captures and extends several common description logics. It extends plain Datalog by features such as existentially quantified rule heads and, at the same time, restricts the rule syntax so as to achieve decidability and tractability. In this paper, we continue the research on Datalog$^\pm$. More precisely, we generalize the well-founded semantics (WFS), as the standard semantics for nonmonotonic normal programs in the database context, to Datalog$^\pm$ programs with negation under the unique name assumption (UNA). We prove that for guarded Datalog$^\pm$ with negation under the standard WFS, answering normal Boolean conjunctive queries is decidable, and we provide precise complexity results for this problem, namely, in particular, completeness for PTIME (resp., 2-EXPTIME) in the data (resp., combined) complexity.

## 1 Introduction

The recent Datalog$^\pm$ family of ontology languages [1] extends plain Datalog by the possibility of existential quantification in rule heads and other features, and simultaneously restricts the rule syntax to achieve decidability and tractability. The following example illustrates how ontological knowledge (encoded in a description logic (DL)) can be 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 is expressible by the two axioms *ConferencePaper* $\sqsubseteq$ *Article* and *Scientist* $\sqsubseteq \exists$*isAuthorOf* in the TBox, respectively, while the knowledge that John is a scientist is expressible by the axiom *Scientist*$(john)$ in the ABox. In Datalog$^\pm$, the former are encoded as the rules *ConferencePaper*$(X) \rightarrow$ *Article*$(X)$ and *Scientist*$(X) \rightarrow \exists Y$ *isAuthorOf*$(X, Y)$, respectively, and the latter is encoded by an identical fact in the database. A simple Boolean conjunctive query (BCQ) asking whether John authors a paper is $\exists X$ *isAuthorOf*$(john, X)$. ■

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 [1], 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, it is natural to ask whether the richer and more expressive well-founded negation could be defined for Datalog$^{\pm}$. The well-founded semantics (WFS) for normal (logic) programs [2] is one of the most widely used semantics for nonmonotonic normal programs. Due to its computational properties (differently from the stable model semantics), it is the standard semantics for such programs for database applications. It is thus especially under a data-oriented perspective of great importance for (dealing with very large amounts of data on) the Web (see [3] for a recent survey of Web-related applications of Datalog$^{\pm}$). 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 [4], we focus 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. But our research there is guided by the goal of defining a WFS for normal Datalog$^{\pm}$ that is close to OWL and its profiles as well as typical DLs, which all have in common that they do not make the unique name assumption (UNA). Thus, the new semantics in [4], called the *equality-friendly WFS (EFWFS)*, also does not make the UNA.

The WFS for normal Datalog$^{\pm}$ without the UNA, however, actually generalizes neither Datalog$^{\pm}$ nor standard normal (logic) programs, which make the UNA, in contrast. The UNA is also rather common in logic programming and databases in general, as well as in some important DLs, such as the *DL-Lite* family of tractable description logics [5,6]. I.e., the results in this paper (which are technically very different from those in [4]) also allow us to interpret extensions of the *DL-Lite*-family with nonmonotonic negation using the standard well-founded semantics. The following example demonstrates this for $\mathit{DL\text{-}Lite}_{\mathcal{R},\sqcap,\mathrm{not}}$ from [4].

*Example 2.* Consider the $\mathit{DL\text{-}Lite}_{\mathcal{R},\sqcap,\mathrm{not}}$-ontology $\mathcal{T}$:

$$\begin{aligned}
\mathit{Person}, \mathit{Employed}, \mathrm{not}\exists \mathit{JobSeekerID} &\sqsubseteq \exists \mathit{EmployeeID},\\
\mathit{Person}, \mathrm{not}\mathit{Employed}, \mathrm{not}\exists \mathit{EmployeeID} &\sqsubseteq \exists \mathit{JobSeekerID},\\
\exists \mathit{EmployeeID}^{-} \sqcap \mathrm{not}\exists \mathit{JobSeekerID}^{-} &\sqsubseteq \mathit{ValidID}.
\end{aligned}$$

The first rule expresses that an employed person who is not registered as a job seeker has an employee's ID, the second rule states that a person who is not employed, and does not have an employee's ID is registered as a job seeker, and the final rule expresses that IDs are only valid, if they are not an employee's and a job seeker's ID simultaneously. As pointed out in [4], it is not difficult to translate this ontology $\mathcal{T}$ into a corresponding guarded normal Datalog$^{\pm}$ program $P_{\mathcal{T}}$. We then have the choice of defining the semantics of $P_{\mathcal{T}}$ as either the equality-friendly WFS of $P_{\mathcal{T}}$ as in [4] or as the standard WFS of $P_{\mathcal{T}}$. The latter is possible thanks to the decidability and complexity results in this paper, and we argue that the WFS is in fact the desirable option here: If we add the set of facts $\{\mathit{Person}(a), \mathit{Person}(b), \mathit{Employed}(a)\}$, it is easy to see that the standard WFS derives both $\mathit{EmployeeID}(a, f(a))$ and $\mathit{JobSeekerID}(b, g(b))$ and, because $f(a) \neq g(b)$, also $\mathit{ValidID}(f(a))$. Here, $f(a)$ and $g(b)$ denote null values that have been generated using Skolem function. In the equality-friendly WFS, however, we do not know whether or not $f(a) \neq g(b)$, and we thus cannot derive that $f(a)$ is a valid ID. ■

Another serious drawback of the equality-friendly WFS is that answering atomic queries is co-NP-hard in the data complexity, and thus does not have the same nice computational properties as in the standard WFS for normal logics programs, namely, a polynomial data complexity. The development of a WFS for normal Datalog$^{\pm}$ under the UNA (with hopefully lower data complexity than the WFS without the UNA) is thus still an important open problem, which we therefore tackle in the present paper.

The central question of this paper is whether the results for guarded positive Datalog$^{\pm}$ from [1] can be extended to guarded normal Datalog$^{\pm}$ under the WFS and UNA, that is, whether there exists a finite part of a chase for normal tuple-generating dependencies (normal TGDs) that can be used to evaluate normal BCQs (NBCQs), which has only constant depth in the data complexity. This then implies that NBCQs to guarded normal Datalog$^{\pm}$ programs can be evaluated in polynomial time in the data complexity. As we will see in this paper, the answer to this central question turns out to be positive. But finding this answer is rather involved technically. Roughly speaking, this is due to the fact that compared to stratified Datalog$^{\pm}$ with negation [1], we now have to make sure that (i) also the derivation of negative atoms in each iteration step, which is done via greatest unfounded sets, can be done on a finite part of an infinite chase, and that (ii) we only need a finite part of the now infinite iteration for the computation of the well-founded model via its fixpoint operator (rather than a finite iteration along the finitely many different levels of a stratification).

As the main contributions of this paper, we thus obtain that answering NBCQs to guarded normal Datalog$^{\pm}$ under the WFS and UNA is decidable and can be done in polynomial time in the data complexity. Furthermore, we show that it is in 2-EXPTIME in the combined complexity in general and in EXPTIME in the combined complexity in the case where the arities of all predicates are bounded by a constant. Hardness for these complexity classes follows from the fact that already answering BCQs to the more restricted guarded Datalog$^{\pm}$ without negation is hard for them. Note that an extended version of this paper is accepted for publication in: *Proceedings PODS-2013* [9].

## 2 Preliminaries

### 2.1 Databases and Queries

We assume (i) an infinite universe of *(data) constants* $\Delta$ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* $\Delta_N$ (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries and dependencies). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in $\Delta_N$ following all symbols in $\Delta$. 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, null, 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. We denote by $pred(\mathbf{a})$ and $dom(\mathbf{a})$ its predicate and the set of all its arguments, respectively. The latter two notations are naturally extended to sets of atoms and conjunctions of atoms. 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 $\Delta$. 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, but without nulls. 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 $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \to \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) $\mu$ 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 $\Delta$ for which there exists a homomorphism

$\mu\colon \mathbf{X}\cup\mathbf{Y}\to\Delta\cup\Delta_N$ 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 $Q(D)\neq\emptyset$.

## 2.2 Normal Logic Programs

We now briefly recall standard normal logic programs. Let $\Xi$ 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 $\Xi$, or of the form $f(t_1,\ldots,t_n)$, where $f$ is a function symbol of arity $n\geqslant 0$ from $\Xi$, and $t_1,\ldots,t_n$ are terms. An *atom* has the form $p(t_1,\ldots,t_n)$, where $p$ is a predicate symbol of arity $n\geqslant 0$ from $\Xi$, and $t_1,\ldots,t_n$ are terms. A *literal* $l$ is an atom $p$ or a negated atom $\neg p$. A *(normal) rule* $r$ is of the form

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

where $\alpha,\beta_1,\ldots,\beta_{n+m}$ are atoms and $m,n\geqslant 0$. We call the atom $\alpha$ 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 called 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 (1) with $m=n=0$ is also called a *fact*. A *normal program* $P$ is a finite set of normal rules (1). We say $P$ is *positive* iff $m=0$ for all normal rules (1) in $P$. For normal programs $P$, we denote by $P^+$ the positive program obtained from $P$ by removing all negative literals from the rule bodies.

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 $\Xi$. 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 in $r$ by a ground term from $HU_P$. We denote by $\mathit{ground}(P)$ 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 $\neg.S=\{\neg.\ell\mid\ell\in S\}$, $S^+=\{a\in S\mid a \text{ is an atom}\}$, and $S^-=\{\neg a\mid\neg a\in S\}$. We denote by $\mathit{Lit}_P=HB_P\cup\neg.HB_P$ the set of all ground literals with predicate symbols from $P$ and ground terms from $HU_P$. A set of ground literals $S\subseteq\mathit{Lit}_P$ is *consistent* iff $S\cap\neg.S=\emptyset$. A *(three-valued) interpretation* w.r.t. $P$ is any consistent set of ground literals $I\subseteq\mathit{Lit}_P$.

## 2.3 Normal BCQs

We 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 ("$\neg$"-free)) atoms of $Q$. In the sequel, w.l.o.g., BCQs contain no constants. An NBCQ $Q$ is satisfied in an interpretation $I\subseteq\mathit{Lit}_P$ if there is a homomorphism $\mu$ such that $\mu(a)\in I$ and $\neg\mu(b)\in I$ for all $a\in Q^+$ and $b\in Q^-$. Answers to an NBCQ over a database are then defined as in the case of BCQs.

## 2.4 Normal TGDs

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* 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 $\mathcal{R}$ (all these atoms without nulls). Note that TGDs can be reduced to TGDs with only single atoms in their heads. Normal TGDs are informally TGDs that may also contain (default-)negated atoms in their bodies. Given a relational schema $\mathcal{R}$, a *normal TGD (NTGD)* $\sigma$ 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 $\mathcal{R}$ (all these atoms without nulls). It is also abbreviated as $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \Psi(\mathbf{X}, \mathbf{Z})$. As in the case of 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 $\sigma$, and by $body^{+}(\sigma)$ and $body^{-}(\sigma)$ the sets of all positive and negative ("$\neg$"-free) atoms in the body of $\sigma$, respectively.

As for the semantics, a normal TGD $\sigma$ 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 $\sigma$ that maps (i) all atoms of $body^{+}(\sigma)$ to atoms of $D$ and (ii) no atom of $body^{-}(\sigma)$ to atoms of $D$ (i.e., atoms not in $D$ are false), then there exists an extension $h'$ of $h$ that maps all atoms of $head(\sigma)$ to atoms of $D$.

A normal TGD $\sigma$ is *guarded* iff it contains a positive atom in its body, denoted $guard(\sigma)$, that contains all universally quantified variables of $\sigma$. W.l.o.g., to simplify such $\sigma$ in formal proofs, constants occur only in the guards of $\sigma$ (as all the other atoms with constants can be abbreviated by fresh atoms without constants, which can be defined via guarded TGDs). A *guarded (normal) Datalog$^{\pm}$ program* is a finite set of guarded (N)TGDs.

Given an NTGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, the *functional transformation* of $\sigma$, denoted $\sigma^{f}$, is the normal rule $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \Psi(\mathbf{X}, \mathbf{f}_{\sigma}(\mathbf{X}, \mathbf{Y}))$, where $\mathbf{f}_{\sigma}$ is a vector of function symbols $f_{\sigma,Z}$ for $\sigma$, one for every variable $Z$ in $\mathbf{Z}$. Given a set $\Sigma$ of NTGDs, the *functional transformation* of $\Sigma$, denoted $\Sigma^{f}$, is obtained from $\Sigma$ by replacing each TGD $\sigma$ in $\Sigma$ by $\sigma^{f}$. Note that the functional transformation of a guarded Datalog$^{\pm}$ program is a positive program.

## 2.5 Guarded Chase Forests

Let $\Sigma$ be a guarded Datalog$^{\pm}$ program (without negation) over a relational schema $\mathcal{R}$, let $D$ be a database for $\mathcal{R}$, and let $P := D \cup \Sigma^{f}$.

The *guarded chase forest* $\mathcal{F}(P)$ of $P$ is the union of the following forests $\mathcal{F}_i(P)$: We start with a forest $\mathcal{F}_0(P)$ that contains, for each fact $a$ in $P$, a unique node labeled $a$; there are no other nodes and no edges. Now let $i \geq 0$. The forest $\mathcal{F}_{i+1}(P)$ is obtained from $\mathcal{F}_i(P)$ by adding new nodes and edges as follows. Let $\mathcal{A}$ be the set of all labels of nodes of $\mathcal{F}_i(P)$. For each node $v$ in $\mathcal{F}_i(P)$ and each rule $r \in$ *ground*$(P)$ such that $\mathrm{guard}(r)$ is the label of $v$ and $B(r) \subseteq \mathcal{A}$, there is a child $w$ of $v$ with label $H(r)$, and the edge from $v$ to $w$ is labeled with $r$.

Given a graph $G$, we denote by $V(G)$ the set of nodes of $G$. We often write $v \in G$ instead of $v \in V(G)$. The label of a node $v$ in $\mathcal{F}(P)$ is denoted by $\mathrm{label}(v)$. We extend this notation to sets $V \subseteq V(\mathcal{F}(P))$ by letting $\mathrm{label}(V) := \bigcup_{v \in V} \mathrm{label}(v)$, and to subforests $\mathcal{F}$ by letting $\mathrm{label}(\mathcal{F}) := \mathrm{label}(V(\mathcal{F}))$. The *derivation level of a node* $v$ in $\mathcal{F}(P)$, denoted $\mathrm{level}_P(v)$, is the smallest integer $i \geq 0$ such that $v \in \mathcal{F}_i(P)$. Observe that $\mathrm{level}_P(v)$ is in general different from the depth of $v$ in $\mathcal{F}(P)$. We define the *derivation level of an atom* $a$ in $\mathcal{F}(P)$, denoted by $\mathrm{level}_P(a)$, as the minimum of $\mathrm{level}_P(v)$ over all nodes $v$ in $\mathcal{F}(P)$ with label $a$, or as $\infty$ if no such node exists.

## 2.6 Well-Founded Semantics for Normal Logic Programs

The *well-founded semantics* [2] is the most widely used semantics for nonmonotonic logic programs, and it is especially under a data-oriented perspective of great importance for the Web. The well-founded semantics of normal programs $P$ has many different equivalent definitions [2,7]. We recall here 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 $r \in ground(P)$ 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 is the greatest unfounded set of $P$ w.r.t. $I$, denoted $U_P(I)$. Intuitively, if $I$ is compatible with $P$, then all atoms in $U_P(I)$ can be safely switched to false and the resulting interpretation is still compatible with $P$. The greatest unfounded set of a partial interpretation $I$ intuitively 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 putting $T_P(I) = \{H(r) \mid r \in ground(P),\ B^+(r) \cup \neg.B^-(r) \subseteq I\}$ and $W_P(I) = T_P(I) \cup \neg.U_P(I)$. The operator $W_P$ is monotonic, and thus has a least fixpoint, denoted $\mathit{lfp}(W_P)$, which is the *well-founded semantics* of $P$, denoted $WFS(P)$ (a three-valued interpretation completable to a two-valued model of $P$). A ground atom $a \in HB_P$ is *well-founded* (resp., *unfounded*) relative to $P$, if $a$ (resp., $\neg a$) is in $\mathit{lfp}(W_P)$. Intuitively, starting with $I = \emptyset$, rules are applied to obtain new positive and negated facts (via $T_P(I)$ and $\neg.U_P(I)$, respectively). This process is repeated until no longer possible. A literal $\ell \in Lit_P$ is a *consequence* of $P$ under the well-founded semantics iff $\ell \in WFS(P)$.

## 3 Well-Founded Semantics for Guarded Normal Datalog$^\pm$

The well-founded semantics for a guarded normal Datalog$^\pm$ program $\Sigma$ relative to a given database $D$ under the UNA is defined using the well-founded semantics of the logic program obtained from taking the union of the functional transformation of $\Sigma$ and $D$.

**Definition 3.** Let $\mathcal{R}$ be a relational schema, and $\Sigma$ be a guarded normal Datalog$^\pm$ program. The *well-founded model* of a database $D$ under $\Sigma$, denoted WFS$(D, \Sigma)$, is defined as WFS$(D \cup \Sigma^f)$. An NBCQ $Q$ is satisfied over $D$ under $\Sigma$ w.r.t. the well-founded semantics if WFS$(D, \Sigma) \models Q$.

*Example 4.* Let $\Sigma$ be a guarded normal Datalog$^\pm$ program such that $\Sigma^f$ has the form:

$$R(X,Y,Z) \rightarrow R(X,Z,f(X,Y,Z)), \qquad R(X,Y,Z) \wedge P(X,Y) \wedge \neg Q(Z) \rightarrow P(X,Z),$$
$$R(X,Y,Z) \wedge \neg P(X,Y) \rightarrow Q(Z), \qquad R(X,Y,Z) \wedge \neg P(X,Z) \rightarrow S(X),$$
$$P(X,Y) \wedge \neg S(X) \rightarrow T(X),$$

and let $D = \{R(0,0,1), P(0,0)\}$. It is easy to see that WFS$(D, \Sigma)$ includes the atom $R(0,1, f(0,0,1))$, as we can apply the first rule in $\Sigma^f$ to derive this atom from the atoms in $D$. Furthermore, WFS$(D, \Sigma)$ includes $P(0,1)$. To see this slightly less obvious fact, note that there is no rule that can derive an atom of the form $R(*,*,1)$, where the $*$'s could be arbitrary constants or nulls. This is due to the fact that any Skolem term of the form $f(t_1,t_2,t_3)$ is by default assumed to be different from 1. Therefore, the only rule instance that could possibly derive $Q(1)$ is $R(0,0,1) \wedge \neg P(0,0) \rightarrow Q(1)$, but as $P(0,0) \in D \subseteq$ WFS$(D, \Sigma)$, we can conclude that $\neg Q(1)$ is in WFS$(D, \Sigma)$. Now $P(0,1) \in$ WFS$(D, \Sigma)$ can be derived using a suitable instance of the second rule. ■

When proving decidability of query answering relative to a Datalog$^\pm$-program, one faces the problem that query answering has to be performed relative to an infinite model that is obtained as a result of the chase algorithm. Nevertheless, as demonstrated in [1], decidability for guarded (positive) Datalog$^\pm$ can be achieved by showing that the guarded chase forest has the following neat "locality" property: for any node $v$ in the guarded chase forest, the tree generated from $v$ is determined (up to isomorphism) by the type of the atom $a$ by which $v$ is labelled. Here, the type of an atom is a pair consisting of $a$ itself together with the collection of atoms $b$ that occur in the chase such that $dom(b) \subseteq dom(a)$. As there are only finitely many non-isomorphic types relative to a given relational schema $\mathcal{R}$, the locality property ensures that any query that can be matched to atoms in the chase can be matched to atoms of bounded depth. In the remainder of this section we demonstrate that a similar locality property holds for guarded normal Datalog$^{\pm}$ programs.

**Characterization via Forward Proofs.** To be able to prove the above-mentioned locality property, we first introduce the notion of a forward proof of an atom. These forward proofs are subforests of the guarded chase forest of the positive part of a guarded normal Datalog$^{\pm}$ program that witness the fact that an atom is potentially contained in the well-founded semantics (an atom without forward proof is certainly false). Forward proofs are instrumental for proving the locality property as they provide a useful link between the guarded chase forest of the positive part of the program on the one hand and the well-founded semantics of the program on the other hand. The characterization of the well-founded semantics for guarded normal Datalog$^{\pm}$ based on the concept of *forward proofs* that we are going to use is essentially the one from [8].

Let $\Sigma$ be a guarded *normal* Datalog$^{\pm}$ program over $\mathcal{R}$, $D$ be a database for $\mathcal{R}$, and $P := D \cup \Sigma^f$. Let $\mathcal{F}^+(P)$ be the forest obtained from $\mathcal{F}(P^+)$ by replacing each edge label, which is a rule $r^+ \in$ *ground*$(P^+)$, by the rule $r \in$ *ground*$(P)$ such that $r^+$ is obtained from $r$ by dropping all negative literals (for simplicity, we assume that $r$ is unique). Let $N(\mathcal{F})$ be the set of all $b \in HB_P$ such that $b \in B^-(r)$ for a rule $r$ that occurs as the label of an edge in $\mathcal{F}$.

The following is a minor modification of the notion of *forward proof* in [8].

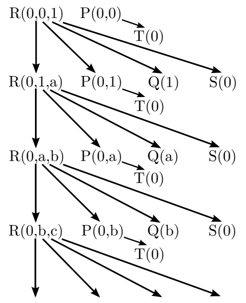
**Definition 5.** A *forward proof of an atom $a \in HB_P$ from $P$ with negative hypotheses* (or just *forward proof* of $a$ from $P$) is a finite subforest $\pi$ of $\mathcal{F}^+(P)$ such that:

1. There is a distinguished node in $\pi$ labeled $a$, called *goal node* of $\pi$.
2. For every node $v \in \pi$ that has a parent $w$ in $\mathcal{F}^+(P)$, we have $w \in \pi$.
3. If $r$ is the label of an edge from a node $v$ to a node $w$ in $\pi$, then for every $b \in B^+(r)$ there is a node $u \in \pi$ with $\text{level}_P(u) < \text{level}_P(w)$ and $\text{label}(u) = b$.

The elements in $\neg.N(\pi)$ are the *negative hypotheses* of $\pi$.

*Example 6.* The program $P^+$ for $P = D \cup \Sigma$ with $D$ and $\Sigma$ chosen as in Example 4 is:

$$\begin{array}{ll} R(0,0,1), P(0,0), & R(X,Y,Z) \to R(X,Z,f(X,Y,Z)), \\ R(X,Y,Z) \wedge P(X,Y) \to P(X,Z), & R(X,Y,Z) \to Q(Z), \\ R(X,Y,Z) \to S(X), & P(X,Y) \to T(X). \end{array}$$



The forest $\mathcal{F}^+(P)$ up to depth three is shown on the right, where $a$, $b$, and $c$ are defined as $f(0,0,1)$, $f(0,1,a)$, and $f(0,a,b)$, respectively. Edge labels are omitted, but they can easily be recovered from the information in Figure 6. For example, the edge from $R(0,0,1)$ to $R(0,1,a)$ is labeled by $R(0,0,1) \to R(0,1,f(0,0,1))$, and the edge from $R(0,1,a)$ to $P(0,a)$ is labeled by $R(0,1,a) \wedge P(0,1) \wedge \neg Q(a) \to P(0,a)$. If $\mathcal{F}$ is the subtree containing $R(0,0,1)$, $P(0,1)$, and $T(0)$ (the child of $P(0,1)$), then $N(\mathcal{F}) = \{Q(1), S(0)\}$. So, $N(\mathcal{F})$ contains all the atoms whose negation is required to fire all the rules in $\mathcal{F}$.

There is exactly one inclusion-minimal forward proof of $R(0,b,c)$ from $P$, namely the subtree $\pi$ of $\mathcal{F}^+(P)$ induced by the nodes labeled $R(0,0,1)$, $R(0,1,a)$, $R(0,a,b)$, and $R(0,b,c)$. Note that $N(\pi) = \emptyset$.

The atom $P(0,a)$ has exactly one inclusion-minimal forward proof from $P$: the subforest $\pi'$ induced by the nodes labeled $R(0,0,1)$, $R(0,1,a)$, $P(0,0)$, $P(0,1)$, and $P(0,a)$. Here, $N(\pi') = \{Q(1), Q(a)\}$. But note that there are infinitely many inclusion-minimal forward proofs of $S(0)$ from $P$, each corresponding to a node in $\mathcal{F}^+(P)$ labeled with $S(0)$. ∎

We need the following operator from [8] (where this operator is denoted by **wf**, and it is defined in terms of minimal forward proofs as in [8]).

**Definition 7** **($\widehat{W}_P$).** We define an operator $\widehat{W}_P$ on $\mathcal{P}(Lit_P)$ as follows. For every $I \subseteq Lit_P$ and every $a \in HB_P$:

- $a \in \widehat{W}_P(I)$ if there is a forward proof $\pi$ of $a$ from $P$ with $\neg.N(\pi) \subseteq I$, and
- $\neg a \in \widehat{W}_P(I)$ if for all forward proofs of $a$ from $P$ there is a $b \in N(\pi)$ with $b \in I$.

It is not hard to see that $\widehat{W}_P$ is monotone, and therefore has a least fixpoint. Recall that this least fixpoint is defined as the union of the fixpoint stages $\widehat{W}_P^0, \widehat{W}_P^1, \ldots$, where $\widehat{W}_{P,0} := \emptyset$, $\widehat{W}_{P,\alpha} := \widehat{W}_P(\widehat{W}_{P,\alpha-1})$ if $\alpha$ is a successor ordinal, and $\widehat{W}_{P,\alpha} := \bigcup_{\beta<\alpha} \widehat{W}_{P,\beta}$ if $\alpha$ is a limit ordinal. Now:

**Theorem 8** **([8]).** WFS$(P)$ *is the least fixpoint of* $\widehat{W}_P$*:* WFS$(P) = \bigcup_\alpha \widehat{W}_{P,\alpha}$.

*Example 9.* Let us revisit Example 6. Define $t_0 := 0$, $t_1 := 1$, and $t_{i+2} := f(0, t_i, t_{i+1})$. Then, we have $\widehat{W}_{P,1} = \{R(0, t_i, t_{i+1}) \mid i \geq 0\} \cup \{P(0,0)\} \cup \{\neg a \mid a \in HB_P,\ a \notin \text{label}(\mathcal{F}^+(P))\}$. It is not hard to see that for every integer $i \geq 1$, $\widehat{W}_{P,2i} = \widehat{W}_{P,1} \cup \{P(0,t_j) \mid 0 \leq j < i\} \cup \{\neg Q(t_j) \mid 0 \leq j \leq i\}$ and $\widehat{W}_{P,2i+1} = \widehat{W}_{P,1} \cup \{P(0,t_j) \mid 0 \leq j \leq i\} \cup \{\neg Q(t_j) \mid 0 \leq j \leq i\}$. Therefore,

$$\widehat{W}_{P,\omega+2} = \widehat{W}_{P,1} \cup \{P(0,t_j) \mid j \geq 0\} \cup \{\neg Q(t_j) \mid j \geq 0\} \cup \{\neg S(0), T(0)\},$$

and $\widehat{W}_{P,\omega+3} = \widehat{W}_{P,\omega+2}$, hence WFS$(P) = \widehat{W}_{P,\omega+2}$.

This demonstrates that the computation of the least fixpoint of $\widehat{W}_P$ does in general not terminate after $\omega$-many steps. The same is true if we compute WFS$(P)$ as least fixpoint of $W_P$: spelling out the definitions, one can check that $W_P^\omega(\emptyset) = \widehat{W}_{P,\omega}$. ∎

**Locality.** After having introduced the characterization of the well-founded semantics in terms of forward proofs, we are now ready to prove the locality property of the well-founded semantics that we discussed at the beginning of this section.

Throughout this section, we fix a finite set $\Sigma$ of guarded NTGDs over a relational schema $\mathcal{R}$, and a database $D$ for $\mathcal{R}$. Let $P := D \cup \Sigma^f$.

The *($P$-)type* of an atom $a \in HB_P$ is the pair type$_P(a) := (a, S)$, where $S$ consists of all literals $\ell \in$ WFS$(P)$ with $dom(\ell) \subseteq dom(a)$. If we speak of a $P$-type without mentioning the atom, we mean a $P$-type of some atom.

Let $I, I' \subseteq Lit_P$ and $X \subseteq dom(I) \cup dom(I')$. An *$X$-isomorphism* from $I$ to $I'$ is a bijective mapping $f$ from $dom(I)$ to $dom(I')$ such that $f(I) = I'$, and for all $x \in X$, (1) $x \in dom(I)$ if and only if $x \in dom(I')$, and (2) if $x \in dom(I)$, then $f(x) = x$. If there is an $X$-isomorphism from $I$ to $I'$, then $I$ and $I'$ are *$X$-isomorphic*, denoted $I \cong_X I'$. Given two $P$-types $(a, S)$ and $(a', S')$, an *$X$-isomorphism* from $(a, S)$ to $(a', S')$ is an $X$-isomorphism $f$

from $\{a\}$ to $\{a'\}$ with $f(S) = S'$. As for sets of literals, $(a, S)$ and $(a', S')$ are $X$*-isomorphic*, denoted $(a, S) \cong_X (a', S')$, if there is an $X$-isomorphism from $(a, S)$ to $(a', S')$.

Let $\mathcal{F}^*(P)$ be the subforest of $\mathcal{F}^+(P)$ induced by all the nodes in $\mathcal{F}^+(P)$ that are goal nodes of forward proofs $\pi$ from $P$ with $\neg.N(\pi) \subseteq \text{WFS}(P)$. Note that $\mathcal{F}^*(P)$ contains exactly the nodes that correspond to the atoms in $\text{WFS}(P)$. The following lemma demonstrates that the truth of any atom below some node $v \in \mathcal{F}^*(P)$ depends only on the type of $\text{label}(v)$, the labels of the tree $T$ generated by $v$, and all negative literals whose arguments occur in the labels of $T$.

**Lemma 10.** *Let $v \in \mathcal{F}^*(P)$ with $\text{type}_P(\text{label}(v)) = (a, S)$, let $T$ be the subtree of $\mathcal{F}^+(P)$ rooted at $v$, and let $I$ be the set of all literals $\ell \in Lit_P$ such that either $\ell \in S$, or $\ell$ is positive and occurs in $T$, or $\ell$ is negative and $dom(\ell) \subseteq dom(\text{label}(T))$. Then, for all atoms $b \in \text{label}(T)$, we have:*

1. *If there is a forward proof $\pi$ of $b$ from $P$ with $\neg.N(\pi) \subseteq X \subseteq \text{WFS}(P)$, then there exists a forward proof $\pi'$ of $b$ from $S^+ \cup \Sigma^f$ with $\neg.N(\pi') \subseteq X \cap I$ and with the property that $\text{label}(\pi') \subseteq S \cup \text{label}(T)$.*
2. *If there exists a forward proof $\pi'$ of $b$ from $S^+ \cup \Sigma^f$, then $\pi'$ can be extended to a forward proof $\pi$ of $b$ from $P$ with $\neg.N(\pi) \setminus \neg.N(\pi') \subseteq \text{WFS}(P)$.*

We are now able to formulate and prove the locality property of the well-founded semantics that is crucial for our decidability result.

**Lemma 11.** *Let $v_1, v_2 \in \mathcal{F}^*(P)$ be two nodes with the types $\text{type}_P(\text{label}(v_i)) = (a_i, S_i)$ for $i \in \{1, 2\}$. Let $T_i$ be the subtree of $\mathcal{F}^+(P)$ rooted at $v_i$, and let $I_i$ be the set of all literals $\ell \in Lit_P$ such that either $\ell \in S_i$, or $\ell$ is positive and occurs in $T_i$, or $\ell$ is negative and $dom(\ell) \subseteq dom(\text{label}(T_i))$. If $f$ is an $X$-isomorphism from $\text{type}_P(a_1)$ to $\text{type}_P(a_2)$, then there is an $X$-isomorphism from $\text{WFS}(P) \cap I_1$ to $\text{WFS}(P) \cap I_2$ that extends $f$.*

The locality property from the previous lemma is the key for obtaining the following bound on the possible matches of a normal Boolean conjunctive query. Its proof is a generalization of the proof of Lemma 4 in [1].

**Proposition 12.** *Let $\mathcal{R}$ be a relational schema $\mathcal{R}$, and $\delta := 2 \cdot |\mathcal{R}| \cdot (2w)^w \cdot 2^{|R| \cdot (2w)^w}$, where $w$ is the maximum arity of a predicate in $\mathcal{R}$. Let $D$ be a database for $\mathcal{R}$, $\Sigma$ be a set of guarded NTGDs over $\mathcal{R}$, and $Q$ be an NBCQ over $\mathcal{R}$ with $n$ literals. If $\text{WFS}(D \cup \Sigma^f) \models Q$, then there is a homomorphism $\mu$ of $Q$ into $\text{WFS}(D \cup \Sigma^f)$ such that:*

1. *For all $a \in Q^+$, $\mu(a)$ has depth at most $n \cdot \delta$ in $\mathcal{F}^*(P)$.*
2. *For all $a \in Q^-$, $\mu(a)$ does not occur in $\mathcal{F}^+(P)$, or $\mu(a)$ has depth $\leq n \cdot \delta$ in $\mathcal{F}^+(P)$.*

## 4 Evaluating NBCQs under the Well-Founded Semantics

This section presents our complexity bounds for evaluating NBCQs in well-founded models under guarded normal Datalog$^\pm$ programs. We start by describing the most important building block—an alternating algorithm, called *WCHECK*, that decides whether a ground atom belongs to the well-founded model of a database under a guarded normal Datalog$^\pm$ program. Due to space limitations, we only give a high-level overview of WCHECK, explaining the underlying ideas as well as the problems that we need to tackle. For a detailed description and proofs, we refer the reader to [9].

WCHECK's goal is to decide whether a ground atom belongs to the well-founded model of some database under a guarded normal Datalog$^{\pm}$ program. More precisely, its input consists of a database $D$, a guarded normal Datalog$^{\pm}$ program $\Sigma$, and a ground atom $a$, and its task is to decide whether $a \in \text{WFS}(P)$, where $P = D \cup \Sigma^f$.

For the case that $\Sigma$ is positive (i.e., $\Sigma$ contains only NTGDs without negated atoms in their bodies), the task of deciding whether $a$ belongs to $\text{WFS}(P)$ is well-understood [10,1]. Indeed, results in [1] imply that, in this case, $a$ is in $\text{WFS}(P)$ if $a$ belongs to a finite "initial segment" of $\text{WFS}(P)$ that is obtained by iteratively applying the $W_P$-operator to the empty interpretation for a finite number of times, where the number of iterations depends only on $\Sigma$. In other words, it suffices to check whether $a$ occurs in this initial segment of $\text{WFS}(P)$. However, as soon as $\Sigma$ contains NTGDs with negated atoms in their bodies, this approach no longer works. Consider $D$ and $\Sigma$ from Example 4. Example 9 shows that the ground atom $T(0)$ occurs in $\text{WFS}(P)$, but "enters" $\text{WFS}(P)$ only after an infinite number of iterations of the $W_P$-operator.

Although the method for checking membership of ground atoms in well-founded models over positive guarded normal Datalog$^{\pm}$ programs described above does not generalize to arbitrary guarded normal Datalog$^{\pm}$ programs, it turns out that a different one, namely the ACHECK algorithm in [10], can be generalized. This is exactly what WCHECK does.

WCHECK is based on the idea that, if a ground atom $a$ belongs to the well-founded model $W$ of a database $D$ and a guarded normal Datalog$^{\pm}$ program $\Sigma$, then the forest $\mathcal{F}^+(D \cup \Sigma^f)$ contains a path from some root node to a node labeled $a$ such that all "side literals" (i.e., non-guard atoms and negated atoms in the bodies of rules applied along the path) belong to $W$ (this is actually true also for non-ground atoms). For example, in the case of $D$ and $\Sigma$ from Example 4, the atom $T(0)$ belongs to $W$, and indeed $\mathcal{F}^+(D \cup \Sigma^f)$ contains a path from the root $R(0,0,1)$ to $T(0)$ whose side literals $P(0,1)$, $\neg Q(f(0,0,1))$, and $\neg S(0)$ belong to $W$. The idea is quite similar to ACHECK's central idea. The only difference is that in the case of ACHECK, $\Sigma$ is a positive guarded normal Datalog$^{\pm}$ program, $W$ is the result of the chase of $D$ and $\Sigma$, and side literals are replaced by "side atoms" [10]. It is straightforward to verify from the definition of the well-founded model that a path as described above is a sufficient and necessary condition for $a$ to belong to $W$.

To decide whether a path from the root of $\mathcal{F}^+(D \cup \Sigma^f)$ to $a$ with the desired properties exists, WCHECK successively guesses atoms $a_0, a_1, a_2, \ldots$, where $a_0$ is the label of a root of $\mathcal{F}^+(D \cup \Sigma^f)$, and each $a_{i+1}$ is the label of a child of (the node labeled) $a_i$. The idea here is to guess a path from $a_0$ to $a$. Of course, we also need to verify that all the side literals on such a path belong to $W$. Therefore, along with each $a_i$, WCHECK guesses a set $S_i$ of literals in $W$ all of whose arguments appear in $a_i$ and which agree with $S_{i-1}$ on the literals whose arguments appear in $a_{i-1}$, and an ordering $\preceq_i$ of the literals in $S_i$ (more precisely, a total preorder that is a linear order on the atoms). The idea is that $S_i$ contains (at least) all the side literals of rules applied on the path from $a_i$ to $a$ that contain only arguments from $a_i$ (this is enforced by checking that the rule generating $a_{i+1}$ has all its side-literals in $S_i$), and $\preceq_i$ is the order of deriving the atoms in $S_i$. For example, if $D$ and $\Sigma$ are as in Example 4, $a$ is the atom $T(0)$, and WCHECK guesses $a_0 = R(0,0,1)$, $a_1 = R(0,1,f(0,0,1))$, $a_2 = P(0,f(0,0,1))$, and $a_3 = T(0)$, then good choices for the sets $S_i$ would be $S_0 = \{P(0,1), \neg S(0)\}$, $S_1 = S_0 \cup \{\neg Q(f(0,0,1))\}$, $S_2 = \{\neg Q(f(0,0,1)), \neg S(0)\}$, and $S_3 = \{\neg S(0)\}$

What remains is to check that all the literals in $S_i$ belong to $W$. To this end, WCHECK launches subcomputations, one for each literal in an $S_i$. The subcomputations for positive literals $b \in S_i$, where $i$ is assumed to be minimal, are similar to the main computation described above. For $i > 0$ (for 0, it is basically the same as above), the basic idea is to find a path from $a_i$ to $b$ such that all side literals along the path belong to $W$ (where literals in $S_{i-1}$ and literals in $S_i$ that are $\preceq_i$-smaller than $b$ may be assumed to belong to $W$). This is justified because $b$ is in $S_i \setminus S_{i-1}$, and as such it contains at least one element that was created in $a_i$ and must therefore occur in the subtree rooted at $a_i$. The case of negative literals $\neg b \in S_i$ is not so clear. The main idea is to check that every path from $a_i$ to a node labeled $b$ contains either a positive side literal $c$ with $\neg c \in W$, or a negative side literal $\neg c$ with $c \in W$ (where literals in $S_{i-1}$ and literals in $S_i$ that are smaller or equal to $\neg b$ with respect to $\preceq_i$ may be assumed to belong to $W$). In our example above, WCHECK would find out that the only path (and thus all paths) from $a_1 = R(0, 1, f(0, 0, 1))$ to $Q(f(0, 0, 1))$ in $\mathcal{F}^+(D \cup \Sigma^f)$ contains the side literal $\neg P(0, 1)$, and since it knows that $P(0, 1) \in W$ (since it belongs to $S_0$), it would conclude that $\neg Q(f(0, 0, 1)) \in S_1$ belongs to $W$. It is not obvious, though, that this test is enough to establish that $\neg b$ belongs to $W$. Its proof requires the machinery and results from the locality part of Section 3.

The above-sketched WCHECK algorithm can be implemented in such a way that it uses space at most $|D|^{|\mathcal{R}| \cdot 2^{O(w)}}$, where $\mathcal{R}$ is the schema and $w$ is the maximum arity of a predicate in $\mathcal{R}$. It can furthermore be shown that WCHECK correctly decides whether $a \in \mathrm{WFS}(D, \Sigma)$. This yields the upper bounds of the following theorem; the lower bounds follow from the corresponding lower bounds for answering BCQs under guarded Datalog$^\pm$ [10].

**Theorem 13.** *Given a database $D$ for a schema $\mathcal{R}$, a guarded normal Datalog$^\pm$ program $\Sigma$ over $\mathcal{R}$, and a ground atom $a$ over $\mathcal{R}$, deciding $a \in \mathrm{WFS}(D, \Sigma)$ is:*

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

The complexity bounds for deciding membership of ground atoms can be extended to deciding membership of arbitrary literals. For positive literals $a$, the idea is to guess a path to $a$, using WCHECK to verify the literals along the path. For negative literals $\neg a$, we have to check that every path to $a$ contains a side-literal whose complement belongs to the well-founded model; WCHECK is again used to verify literals along to the path.

These complexity bounds and Proposition 12 form the key for proving our main result, Theorem 14, on answering NBCQs in well-founded models under guarded normal Datalog$^\pm$.

**Theorem 14.** *Given a database $D$ for a schema $\mathcal{R}$, a guarded normal Datalog$^\pm$ program $\Sigma$ over $\mathcal{R}$, and an NBCQ $Q$ over $\mathcal{R}$, the problem of deciding $\mathrm{WFS}(D, \Sigma) \models Q$ is:*

1. *2-EXPTIME-complete in general.*
2. *EXPTIME-complete in case $w$ is bounded.*
3. *in PTIME in case $|\mathcal{R}|$, the maximum arity $w$ of a predicate symbol in $\mathcal{R}$, and the number of literals in $Q$ are bounded; there are cases where the problem is PTIME-complete.*

## 5 Conclusion

We have introduced the standard well-founded semantics (WFS) for normal Datalog$^\pm$ programs under the unique name assumption (UNA). We have shown that for guarded normal Datalog$^\pm$ under the standard WFS, answering normal Boolean conjunctive queries is decidable. Furthermore, we have shown that this problem is complete for PTIME in the data complexity, and that it is complete for 2-EXPTIME in the combined complexity in general and complete for EXPTIME in the combined complexity in the case where the arities of all predicates are bounded by a constant. A topic of future research is to explore how to add negative constraints and equality-generating dependencies (EGDs), similarly to [1].

*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”, 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. André Hernich was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).

## References

1. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. **14** (2012) 57–83
2. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. J. ACM **38**(3) (1991) 620–650
3. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/–: A family of logical knowledge representation and query languages for new applications. In: Proceedings LICS-2010, IEEE Computer Society (2010) 228–242
4. Gottlob, G., Hernich, A., Kupke, C., Lukasiewicz, T.: Equality-friendly well-founded semantics and applications to description logics. In: Proceedings AAAI-2012, AAAI Press (2012) 757–764
5. 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) (2007) 385–429
6. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics **10** (2008) 133–173
7. Baral, C., Subrahmanian, V.S.: Dualities between alternative semantics for logic programming and nonmonotonic reasoning. J. Autom. Reasoning **10**(3) (1993) 399–420
8. Schlipf, J.S.: The expressive powers of the logic programming semantics. J. Comput. Syst. Sci. **51**(1) (1995) 64–86
9. Hernich, A., Kupke, C., Lukasiewicz, T., Gottlob, G.: Well-founded semantics for extended Datalog and ontological reasoning. In: Proceedings PODS-2013, ACM Press (2013) In press.
10. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proceedings KR-2008, AAAI Press (2008) 70–80 Revised version: `http://www.dbai.tuwien.ac.at/staff/gottlob/CGK.pdf`.
11. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)