Games and Total Datalog$^\neg$ Queries

Jörg Flum [a], Max Kubierschky [a], Bertram Ludäscher [b]

[a] *Institut für mathematische Logik, Universität Freiburg,*
Eckerstr. 1, D-79104 Freiburg, Germany
flum@uni-freiburg.de

[b] *Institut für Informatik, Universität Freiburg,*
Am Flughafen 17, D-79110 Freiburg, Germany
ludaesch@informatik.uni-freiburg.de

Abstract

We show that the expressive power of Datalog$^\neg$ programs under the well-founded semantics does not decrease when restricted to total programs thereby affirmatively answering an open question posed by Abiteboul, Hull, and Vianu [AHV95]. In particular, we show that for every such program there exists an equivalent *total* program whose only recursive rule is of the form

$$win(\bar{X}) \leftarrow move(\bar{X}, \bar{Y}), \neg win(\bar{Y})$$

where *move* is definable by a quantifier-free first-order formula. Also for the noninflationary semantics we derive a new normal form whose only recursive rule simulates a version of the *game of life*.

Key words: Datalog, well-founded semantics, fixpoint logics, games.

1 Introduction

Consider the Datalog program

$$\begin{aligned} \Pi_0 : \qquad & q(X, Y) \leftarrow edge(X, Y) \\ & q(X, Y) \leftarrow edge(X, Y), q(Y, Z). \end{aligned}$$

⋆ Expanded version of [FKL97].

Here, q is an intensional symbol of Π_0 (precise definitions are given in subsequent sections). Given a database instance $\mathcal{D} = (D, edge^{\mathcal{D}})$ with universe D and binary relation $edge^{\mathcal{D}}$, Π_0 defines a sequence

$$q_0, q_1, q_2, \ldots \tag{1}$$

with $q_0 := \emptyset$ and with q_{n+1} the set of pairs $(a, b) \in D \times D$ such that $q(a, b)$ is the head of a ground instance of a rule of Π_0, whose body is true in $\mathcal{D}$, if q_n is taken as the (actual) interpretation of q. (One easily verifies that q_n is the set of pairs (a, b) such that in $\mathcal{D}$ there is a path from a to b of length $\leq n$.)

Since q only occurs positively in Π_0, the sequence in (1) is increasing, so it eventually becomes constant, the constant value $q_{\boldsymbol{t}}$ being the *truth set* of Π_0 in $\mathcal{D}$. (Clearly, $q_{\boldsymbol{t}}$ is the transitive closure of $edge^{\mathcal{D}}$.) The *query* associating with $\mathcal{D}$ the corresponding binary relation $q_{\boldsymbol{t}}$ is denoted by $(\Pi_0, q_{\boldsymbol{t}})$.

Consider the Datalog$^\neg$ (Datalog with negation) program

$$\begin{aligned} \Pi : \qquad & q(X,Y) \leftarrow r(X,Y,Z), \neg q(V,X) \\ & q(X,Y) \leftarrow \neg s(X,Y,Z), r(X,Y,Z), q(X,Y), \neg q(Y,Y). \end{aligned}$$

Now, q occurs negatively and, contrary to the case of Datalog programs, there are various possible semantics leading to different queries. In this paper we consider the *noninflationary semantics* (NI-semantics) (cf. [AV91]) and the *well-founded semantics* (WF-semantics) [VG93]. In both, Π induces a sequence $q_0, q_1, \ldots$ of subsets of $D \times D$ (see below). Set

$$q_{\boldsymbol{t}} := \{(a, b) \mid \text{there is an } n_0 \text{ s.t. } (a, b) \in q_n \text{ for all } n \geq n_0\}$$

($q_{\boldsymbol{t}}$, the *truth set*, consists of those pairs that eventually are in all members of the sequence),

$$q_{\boldsymbol{f}} := \{(a, b) \mid \text{there is an } n_0 \text{ s.t. } (a, b) \notin q_n \text{ for all } n \geq n_0\}$$

($q_{\boldsymbol{f}}$, the *false set*, consists of those pairs that eventually are outside all members of the sequence), and $q_{\boldsymbol{u}} := (D \times D) \setminus (q_{\boldsymbol{t}} \cup q_{\boldsymbol{f}})$ the *undefined set.*

For a fixed semantics, we say that Π is *total*, if $q_{\boldsymbol{u}} = \emptyset$ in all databases; two programs are *equivalent*, if they have the same truth set in all databases.

For the NI-semantics and the WF-semantics, we show that

- every Datalog$^\neg$ program is equivalent to a total one, and
- for every Datalog$^\neg$ program Π there is another program having as truth set the false set of Π.

Moreover, for both semantics we derive normal forms of game-theoretic flavour.

So, in both semantics, from an extensional point of view, we have the same expressive power, if we restrict to queries (Π, q_t), where Π is a total program or, looking at the other extreme, if we admit as queries (Π, q_t), (Π, q_f), and (Π, q_u) with their obvious meanings.

Let us recall the semantics: In the NI-semantics the stages q_n are defined in exactly the same way as for Datalog programs (but now, in general, the sequence q_n is not increasing, since q may occur negatively).

We come to the WF-semantics. Consider the Datalog$^\neg$ program Π above. Replace all *negative* occurrences of q by a new variable q' (keeping the negation symbol), thus obtaining the program

$$\begin{aligned} \Pi': \qquad & q(X,Y) \leftarrow r(X,Y,Z), \neg q'(V,X) \\ & q(X,Y) \leftarrow \neg s(X,Y,Z), r(X,Y,Z), q(X,Y), \neg q'(Y,Y). \end{aligned}$$

In Π' the symbol q' is extensional, hence, Π' is a Datalog program. Now, in a database instance $\mathcal{D} = (D, r^{\mathcal{D}}, s^{\mathcal{D}})$, the stages q_n of the evaluation of the original program Π are defined by induction: $q_0 := \emptyset$ and q_{n+1} is the truth set of q of the Datalog program Π' in $(\mathcal{D}, q_n)$ (i.e., taking q_n as interpretation of q'). Hence, the evaluation of Π in the WF-semantics corresponds to a nested fixpoint.

While for the NI-semantics the results mentioned above (besides the game-theoretic normal form) are consequences of known facts and are more or less explicit in the literature (see [AS91], [Gro92], [EF95]), the corresponding results for the WF-semantics are new; in particular, they solve an open problem stated in [AHV95]. The parts on the NI-semantics (Section 3) and the WF-semantics (Section 4) may be read independently.

For the WF-semantics we use a normal form for least fixpoint logic LFP due to Immerman [Imm86] to show that every program is equivalent to one whose only recursive rule is of the well-known form

$$win(\bar{X}) \leftarrow move(\bar{X}, \bar{Y}), \neg win(\bar{Y}). \qquad (G)$$

For a database instance $(D, move^{\mathcal{D}})$, the elements (more precisely, the tuples of length equal to $length(\bar{X})$) are viewed as the positions in a game between two players $\mathtt{I}$ and $\mathtt{II}$ that move alternately. Read $move(\bar{X}, \bar{Y})$ as *"from position $\bar{X}$ a player can move to position $\bar{Y}$"*. A player *loses* in $\bar{X}$ if she cannot move; she *wins* in $\bar{X}$ if she can move to a position which the opponent loses. Then in the WF-semantics, $win_{\boldsymbol{t}}$ is the set of positions $\bar{X}$ such that $\mathtt{I}$ has a winning strategy for the game starting at $\bar{X}$, while $win_{\boldsymbol{f}}$ are the positions for which $\mathtt{II}$ has a winning strategy. $win_{\boldsymbol{u}}$ are the *drawn* positions for which neither player has a winning strategy.

Consider, for example, a game where $D = \{a, b, c, d\}$ and $move^{\mathcal{D}} = \{(a, b), (b, a), (b, c), (c, d)\}$. Then $win_{\boldsymbol{t}} = \{c\}$, $win_{\boldsymbol{f}} = \{d\}$, and $win_{\boldsymbol{u}} = \{a, b\}$. In fact, a and b are drawn: a player in b can move to a (moving to c would leave the opponent in a won position), thus avoiding to lose by enforcing a game of infinite length. Now our main result concerning the WF-semantics can be rephrased as

Every game is equivalent to a draw-free game.

The achieved normal form retroactively justifies the ubiquity of the *win-move* example in the literature.

Note that the NI-semantics and WF-semantics coincide for Datalog$^{\neg}$ programs having the normal form (G) above. Thus, in terms of expressive power, the nested fixpoint process is superfluous. However in general, the semantics disagree as can be seen from the program

$$q(X) \leftarrow q(X)$$
$$q(X) \leftarrow \neg q(X).$$

2 Preliminaries

A *database schema* (or *signature*) σ consists of finitely many relation symbols $r_1, \ldots, r_k$ with associated arities $arity(r_i) \geq 0$ and of finitely many constants $c_1, \ldots, c_s$. Let *dom* be a fixed and countable underlying domain. A *database instance* (*database*) over σ is a finite structure $\mathcal{D} = (D, r_1^{\mathcal{D}}, \ldots, r_k^{\mathcal{D}}, c_1^{\mathcal{D}}, \ldots, c_s^{\mathcal{D}})$ with finite universe $D \subseteq dom$, relations $r_i^{\mathcal{D}} \subseteq D^{arity(r_i)}$ and elements $c_i^{\mathcal{D}} \in D$.

Let $inst(\sigma)$ denote the set of all database instances over σ. A *k-ary query q* over σ is a computable function on $inst(\sigma)$ such that (i) $q(\mathcal{D})$ is a k-ary relation on D, and (ii) q is preserved under isomorphisms, i.e., for every isomorphism π of $\mathcal{D}$, $q(\pi(\mathcal{D})) = \pi(q(\mathcal{D}))$. Thus, a query defines a k-ary *global relation* on $inst(\sigma)$.

A *query language* $\mathcal{L}$ is a set of expressions together with a semantics which maps every expression $\varphi \in \mathcal{L}$ to a query (over some σ). The *expressive power* of a query language $\mathcal{L}$ is the class of all queries definable in $\mathcal{L}$. $\varphi \in \mathcal{L}_1$ is *equivalent to* $\psi \in \mathcal{L}_2$ if they express the same query. We say that $\mathcal{L}_1$ is *at most as expressive as* $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \leq \mathcal{L}_2$, if for every expression in $\mathcal{L}_1$ there is an equivalent expression in $\mathcal{L}_2$. Both languages have the *same expressive power*, written as $\mathcal{L}_1 \equiv \mathcal{L}_2$, if $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_2 \leq \mathcal{L}_1$.

Notation. Following logic programming notation, we write domain variables in upper case like X, X', Y etc. Relation symbols like $r_1, \ldots, r_k, win, move$ are denoted in lower case.

$\bar{T}$ denotes a vector of n terms $T_1, \ldots, T_n$ (variables or constants). For a term T we denote by $\tilde{T}$ the sequence $T, \ldots, T$; its length will be clear from the context. If r is a relation symbol of arity n and $T_1, \ldots, T_n$ are terms then $r(T_1, \ldots, T_n)$ is an *atom*.

Datalog$^{(\neg)}$ Programs. A *Datalog*$^{\neg}$ program Π is a finite set of rules of the form

$$H \leftarrow B_1, \ldots, B_n$$

where the *head* H is an atom and all B_i in the *body* are literals (i.e., atoms, negated atoms, equalities, or negated equalities). Relational symbols occurring in some head of Π are called *intensional* and form the signature $idb(\Pi)$, all other relations are *extensional*. The extensional symbols together with the constants form the signature $edb(\Pi)$. For notational simplicity, we often assume that Π only contains one intensional relation symbol, usually q.

In a *Datalog* program, only relations from $edb(\Pi)$ may occur negated in bodies of rules. [1]

Let $\mathcal{D}$ be a database over $edb(\Pi)$. A *ground instance* of a rule is obtained by substituting elements from D for all variables. $ground(\Pi, D)$ denotes the set of all such ground instances of rules of Π.

Fixpoint Semantics for Datalog. Fix a Datalog program Π with a unique intensional symbol q of arity m and a database $\mathcal{D}$ over $edb(\Pi)$. The operator $\Gamma\,(= \Gamma(\Pi, \mathcal{D}))$ maps every subset I of D^m to a subset of D^m:

[1] In [AHV95] such programs are called *semipositive*, while (positive) Datalog programs contain no negation at all.

$$\Gamma(I) := \{\bar{a} \mid q(\bar{a}) \leftarrow B_1, \ldots, B_n \in ground(\Pi, D) \text{ and } B_1, \ldots, B_n \text{ are true in } (\mathcal{D}, I)\}.$$

Here, $(\mathcal{D}, I)$ is the database instance over the signature $edb(\Pi) \cup \{q\}$ that extends $\mathcal{D}$ by interpreting q as the set I.

Then Γ induces a sequence $q_0, q_1, \ldots$ of subsets of D^m given by

$$q_0 := \emptyset, \quad \text{and} \quad q_{n+1} := \Gamma(q_n).$$

Clearly,

$$q_0 \subseteq q_1 \subseteq q_2 \subseteq \cdots$$

In Datalog, the truth set $q_{\boldsymbol{t}}$ is given by

$$q_{\boldsymbol{t}} = \bigcup_{n \geq 0} q_n.$$

We associate to Π the query which maps the database instance $\mathcal{D}$ to $q_{\boldsymbol{t}}$. We denote this query by $(\Pi, q_{\boldsymbol{t}})$.

3 NI-Semantics for Datalog$^{\neg}$ Programs [2]

When applied to Datalog$^{\neg}$ programs, the above Γ-operator also induces a sequence $q_0, q_1, \ldots$ of subsets of D^m. However, since negated intensional atoms may occur in the bodies, in general the sequence is not increasing. For the NI-semantics, we define the *truth set* $q_{\boldsymbol{t}}$, the *false set* $q_{\boldsymbol{f}}$, and the *undefined set* $q_{\boldsymbol{u}}$ by

$$\begin{aligned} q_{\boldsymbol{t}} &:= \{\bar{a} \mid \text{there is an } n_0 \text{ s.t. } \bar{a} \in q_n \text{ for all } n \geq n_0\}, \\ q_{\boldsymbol{f}} &:= \{\bar{a} \mid \text{there is an } n_0 \text{ s.t. } \bar{a} \notin q_n \text{ for all } n \geq n_0\}, \\ q_{\boldsymbol{u}} &:= D^m \setminus (q_{\boldsymbol{t}} \cup q_{\boldsymbol{f}}). \end{aligned}$$

Now, in *NI-Datalog* the program Π gives rise to three queries, $(\Pi, q_{\boldsymbol{t}})$, $(\Pi, q_{\boldsymbol{f}})$, and $(\Pi, q_{\boldsymbol{u}})$ with their obvious meanings. Π is called *total*, if for all database instances we have that $q_{\boldsymbol{u}} = \emptyset$ or, equivalently, for some n,

$$q_{\boldsymbol{t}} = q_n = q_{n+1} = \ldots = q_{\boldsymbol{f}}^c,$$

[2] As already remarked in the introduction, the reader only interested in the well-founded semantics may skip this section.

where $q_{\boldsymbol{f}}^c$ denotes the complement of $q_{\boldsymbol{f}}$ with respect to D^m.

NI-Datalog$_2$ is the restriction of NI-Datalog to total programs and to the corresponding queries $(\Pi, q_{\boldsymbol{t}})$.

The following theorem is a straightforward generalization of a result of [AS91] (cf. also [Gro92])).

Theorem 1 (NI-Datalog $\leq$ NI-Datalog$_2$) *For every NI-Datalog program* Π *there is an equivalent total program. Moreover, there is an equivalent total program having as truth set the false set of* Π.

Proof (Sketch) Let Π be a Datalog$^\neg$ program and (for simplicity) q its unique intensional symbol, say of arity m. Given a database instance $\mathcal{D}$ denote, as above, by q_n the n-th stage of the iteration process. Since the universe D of $\mathcal{D}$ is finite, the sequence $q_0, q_1, \ldots$ must become periodic, so there are $n_0 \geq 0$ and $l_0 \geq 1$ such that $q_n = q_{n+l_0}$ holds for all $n \geq n_0$. Choose k s.t. $l := k \cdot l_0 \geq n_0$. Then $q_l = q_{l+k \cdot l_0} = q_{2 \cdot l}$.

Clearly, if $q_{2n} = q_n$ then

- $q_0, q_1, \ldots$ eventually gets constant iff $q_n = q_{n+1}$.
- $q_{\boldsymbol{t}} = q_n \cap \cdots \cap q_{2n-1} = \bigcap_{k \geq 0} q_{n+k}$.
- $q_{\boldsymbol{f}} = q_n^c \cap \cdots \cap q_{2n-1}^c = \bigcap_{k \geq 0} q_{n+k}^c$.

(Here, for $I \subseteq D^m$, we denote by I^c the complement of I with respect to D^m.) These facts can be used to obtain a total program with the same truth set as Π and a total program whose truth set is the false set of Π. $\Box$

Now, we show that for every NI-Datalog program there is an equivalent total one in the form of a game resembling the *game of life* [Gar70] (recall that two programs are *equivalent* if they have the same truth set for all databases).

Theorem 2 (NI-Datalog $\leq$ NI-Datalog$_2^{GL}$) *For every NI-Datalog program there is an equivalent total one whose only recursive*[3] *rule has the form*

$$alive(\bar{X}) \leftarrow r(\bar{X}, \bar{V}), s(\bar{X}, \bar{W}), alive(\bar{V}), \neg alive(\bar{W}).$$

Intuitively, the rule says that cell $\bar{X}$ is alive in the next generation (= stage) if there is an r-neighbour $\bar{V}$ and an s-neighbour $\bar{W}$ of $\bar{X}$ such that in the actual generation $\bar{V}$ is alive and $\bar{W}$ is dead.

[3] A rule r is *recursive* if some literal in the body of r depends—directly or indirectly via other rules—on the atom in the head of r, cf. [AHV95].

To obtain this result we improve a known normal form for *partial fixpoint logic* PFP.

Partial Fixpoint Logic. PFP-formulas are obtained by repeated applications of first-order operations $\{\neg, \wedge, \vee, \forall, \exists\}$ and the fixpoint operator FP starting from atoms and equations; that is, we add to the first-order formation rules the rule

$$\frac{\varphi}{[\mathrm{FP}_{q(\bar{X})}\ \varphi]\bar{T}} \qquad \text{(FP)}$$

where $length(\bar{X}) = length(\bar{T}) = arity(q)$. The semantics $\mathcal{D} \models \psi$ is given as usual (cf. [EF95]). In particular, for $\psi(\bar{Y}, \bar{Z}) = [\mathrm{FP}_{q(\bar{X})}\ \varphi(q, \bar{X}, \bar{Z})]\bar{Y}$ and $\bar{a}, \bar{b} \in D$:

$$\mathcal{D} \models \psi(\bar{a}, \bar{b}) \quad \text{iff} \quad \bar{a} \in q_{\boldsymbol{t}},$$

where the truth set $q_{\boldsymbol{t}}$ is defined by

$$q_{\boldsymbol{t}} := \{\bar{d} \in D \mid \text{ there is some } n_0 \text{ s.t. } \bar{d} \in q_n \text{ for all } n \geq n_0\}$$

and where $q_0 := \emptyset$ and $q_{n+1} := \{\bar{d} \in D \mid \mathcal{D} \models \varphi(q_n, \bar{d}, \bar{b})\}$.

Every PFP-formula $\psi(\bar{Y})$ defines a query q_ψ as follows:

$$q_\psi : \quad \mathcal{D} \mapsto \{\bar{d} \mid \mathcal{D} \models \psi(\bar{d})\}$$

It is well-known that NI-Datalog $\equiv$ PFP (e.g., see [EF95]).

Every PFP-formula is equivalent to a formula which only contains one fixpoint operator FP. Moreover, by increasing the arity of the second-order variable, Grohe [Gro94] has shown that the fixpoint operator can be rewritten in such a way that an element of a new stage is witnessed by two elements, one belonging to the preceding stage, the other one belonging to its complement. More precisely, for every PFP-formula $\psi(\bar{Y})$ there is an equivalent formula of the form

$$\exists U[\mathrm{FP}_{q(\bar{X})}\ (\psi_0(\bar{X}) \vee \exists \bar{V} \exists \bar{W}\ (q(\bar{V}) \wedge \neg q(\bar{W}) \wedge \psi_1(\bar{X}, \bar{V}, \bar{W})))]\bar{Y}\widetilde{U} \qquad (*)$$

where ψ_0, ψ_1 are quantifier-free and do not contain q and where $\widetilde{U} = U, \ldots, U$ for a variable U (thus, $arity(q) = length(\bar{X}) = length(\bar{Y}) + length(\widetilde{U})$). Moreover, one can assume that the formula $(*)$ is *total* (for all databases $\mathcal{D}$ the false set $q_{\boldsymbol{f}}$ is the complement of the truth set $q_{\boldsymbol{t}}$) and *nontrivial* (for all $\mathcal{D}$, we have $\emptyset \neq q_n \neq D^{arity(q)}$ for all $n \geq 1$) [4].

[4] Although the answer to the original query q_ψ may be $\emptyset$ or $D^{arity(q_\psi)}$.

We improve this normal form by replacing the ternary relation between $\bar{X}$, $\bar{V}$, and $\bar{W}$ by two binary relations:

Proposition 3
Every PFP-formula $\psi(\bar{Y})$ is equivalent to a total one of the form

$$\exists U[\mathrm{FP}_{q(\bar{X})}\ (\psi_0(\bar{X}) \\ \vee \exists \bar{V} \exists \bar{W}\ (q(\bar{V}) \wedge \neg q(\bar{W}) \wedge \psi_1(\bar{X}, \bar{V}) \wedge \psi_2(\bar{X}, \bar{W})))]\bar{Y}\widetilde{U}\bar{Y}\widetilde{U} \quad (+)$$

where ψ_0, ψ_1, and ψ_2 are quantifier-free and do not contain q.

We postpone the proof of this proposition and first show Theorem 2:

Proof of Theorem 2. Let Π be a NI-Datalog program. Consider an equivalent PFP-formula which we may assume to be given in the form (+). But the formula (+) is equivalent to a NI-Datalog program Π' of the desired form:

$$\begin{array}{lrl} \Pi': & q(\bar{X}) & \leftarrow \psi_0(\bar{X}) \\ & r(\bar{X}, \bar{V}) & \leftarrow \psi_1(\bar{X}, \bar{V}) \\ & s(\bar{X}, \bar{W}) & \leftarrow \psi_2(\bar{X}, \bar{W}) \\ & q(\bar{X}) & \leftarrow r(\bar{X}, \bar{V}), s(\bar{X}, \bar{W}), q(\bar{V}), \neg q(\bar{W}) \\ & answer(\bar{Y}) & \leftarrow q(\bar{Y}, \widetilde{U}, \bar{Y}, \widetilde{U}). \end{array}$$

More precisely, if $\psi_0(\bar{X})$ has an equivalent disjunctive normal form $\bigvee_{i=1}^{k}(\varphi_{i,1}(\bar{X}) \wedge \cdots \wedge \varphi_{i,m_i}(\bar{X}))$, one has to replace the rule "$q(\bar{X}) \leftarrow \psi_0(\bar{X})$" above by k rules

$$\begin{array}{lcl} q(\bar{X}) & \leftarrow & \varphi_{1,1}(\bar{X}), \ldots, \varphi_{1,m_1}(\bar{X}) \\ \vdots & \vdots & \qquad \vdots \\ q(\bar{X}) & \leftarrow & \varphi_{k,1}(\bar{X}), \ldots, \varphi_{k,m_k}(\bar{X}) \end{array}$$

and similarly for the rules defining r and s. □

Proof of Proposition 3. Assume that the PFP-formula $\psi(\bar{Y})$ has the Grohe normal form

$$\exists U[\mathrm{FP}_{q(\bar{X})}\ (\psi_0(\bar{X}) \vee \exists \bar{V} \exists \bar{W}\ (q(\bar{V}) \wedge \neg q(\bar{W}) \wedge \psi_1(\bar{X}, \bar{V}, \bar{W})))]\bar{Y}\widetilde{U}$$

and that ψ is total and nontrivial. Let $arity(q) = m$. Then $\psi(\bar{Y})$ is equivalent to a formula of the form

$$\exists U[\mathrm{FP}_{r(\widehat{X})}\ ((\psi_0(\bar{X}_1) \wedge \bar{X}_1{=}\bar{X}_2) \\ \vee \exists\widehat{V}\exists\widehat{W}\ (r(\widehat{V}) \wedge \neg r(\widehat{W}) \wedge \chi_1(\widehat{X},\widehat{V}) \wedge \chi_2(\widehat{X},\widehat{W})))]\bar{Y}\widetilde{U}\bar{Y}\widetilde{U}$$

where $arity(r) = 2m$, $\widehat{X} = \bar{X}_1\bar{X}_2$, $\widehat{V} = \bar{V}_1\bar{V}_2$, $\widehat{W} = \bar{W}_1\bar{W}_2$, and $length(\bar{X}_i) = length(\bar{V}_i) = length(\bar{W}_i) = m$ for $i = 1, 2$.

The equivalence holds if χ_1 and χ_2 are arranged in such a way that the following holds for all databases and all $n \geq 1$:

$$\begin{aligned} r_{2n-1} &= \{(\bar{a},\bar{a}) \mid \bar{a} \in q_n\} \cup (q_{n-1} \times q_{n-1}^c) \\ r_{2n} &= \{(\bar{a},\bar{a}) \mid \bar{a} \in q_n\} \cup (q_n \times q_n^c), \end{aligned}$$

(then, $r_{\boldsymbol{t}} = \{(\bar{a},\bar{a}) \mid \bar{a} \in q_{\boldsymbol{t}}\} \cup (q_{\boldsymbol{t}} \times q_{\boldsymbol{t}}^c)$).

This is achieved by setting

$$\begin{aligned} \chi_1 &:= (\bar{X}_1 = \bar{X}_2 \rightarrow (\psi_1(\bar{X}_1,\bar{V}_1,\bar{V}_2) \wedge \bar{V}_1 \neq \bar{V}_2)) \\ &\quad \wedge (\bar{X}_1 \neq \bar{X}_2 \rightarrow (\bar{X}_1 = \bar{V}_1 = \bar{V}_2)) \\ \chi_2 &:= (\bar{X}_1 \neq \bar{X}_2 \rightarrow (\bar{X}_2 = \bar{W}_1 = \bar{W}_2)). \qquad \square \end{aligned}$$

Remark. By passing in the proof to a relation r of higher arity (and a longer sequence $\widetilde{U}$), one can obtain a normal form $\exists U[\ldots]\bar{Y}\widetilde{U}$, where the formula inside the brackets has the same form as in (+) of Proposition 3.

4 WF-Semantics for Datalog$^\neg$ Programs

As mentioned in the introduction, the evaluation of Datalog$^\neg$ programs under the WF-semantics corresponds to a nested fixpoint, also called *alternating fixpoint* [VG93]. It is computed as follows:

Given a Datalog$^\neg$ program Π, replace every negative occurrence of $q \in idb(\Pi)$ by the new relation symbol q' (keeping the negation symbol). Since q' does not occur in the head of any rule, it is extensional, so the resulting program Π' is a Datalog program. The stages q_n of Π for a given database $\mathcal{D}$ are defined using the program Π': $q_0 := \emptyset$, and q_{n+1} is the result of evaluating Π' in $(\mathcal{D}, q_n)$, i.e., where q' is interpreted by q_n. As above, the set $q_{\boldsymbol{t}}$, $q_{\boldsymbol{f}}$, and $q_{\boldsymbol{u}}$ are defined, giving rise to the *WF-Datalog* queries $(\Pi, q_{\boldsymbol{t}})$, $(\Pi, q_{\boldsymbol{f}})$, and $(\Pi, q_{\boldsymbol{u}})$, respectively.

One easily verifies that

$$q_0 \subseteq q_2 \subseteq q_4 \subseteq \ \cdots \ \subseteq q_5 \subseteq q_3 \subseteq q_1 \ ,$$

so

$$q_{\boldsymbol{t}} = \bigcup_{n \geq 0} q_{2n} \ , \quad \text{and} \quad q_{\boldsymbol{f}} = (\bigcap_{n \geq 0} q_{2n+1})^c \ .$$

This was used by van Gelder [VG93] to show [5]

Theorem 4 (WF-Datalog $\leq$ LFP)
For every WF-Datalog query there is an equivalent LFP-formula.

LFP-formulas (for *least fixpoint*) are defined like PFP-formulas except that a proviso is added to the rule (FP) above, namely, the variable q may only occur positively in the formula φ. This implies that all LFP-formulas are total (the truth set always being the least fixpoint of the corresponding operation).

Let *WF-Datalog*$_2$ be the restriction of WF-Datalog to *total programs* and queries of the form $(\Pi, q_{\boldsymbol{t}})$ (recall that Π is total, if $q_{\boldsymbol{u}} = \emptyset$ for all databases $\mathcal{D}$ and all $q \in idb(\Pi)$). In [AHV95], Abiteboul et. al. raised the question whether one can find for each WF-Datalog program an equivalent total one. In other words, is WF-Datalog $\leq$ WF-Datalog$_2$? (WF-Datalog$_2$ $\leq$ WF-Datalog holds trivially.) When restricted to *ordered databases*, this is known to be the case, since stratified Datalog is equivalent to LFP on ordered databases, and $q_{\boldsymbol{u}} = \emptyset$ for stratified Datalog programs evaluated under the WF-semantics (see, e.g., [AHV95]).

As we will show in the sequel, the question can also be answered affirmatively in the absence of order. First, using the above result of van Gelder and a normal form for LFP due to Immerman, we show that every WF-Datalog program can be transformed into a normal form which corresponds to a certain game. Finally, we establish our main tool, the reduction of games to draw-free games.

4.1 Win-Move Games

Definition 5 (Win-Move Games) *A* win-move game *(or* game *for short) is a triple $\mathcal{G} = (V, M, v_0)$ where V is a finite set of* positions *(or* vertices*), $M \subseteq V \times V$ is a set of possible* moves*, and $v_0 \in V$ is the distinguished* start position *of $\mathcal{G}$.*

[5] In [VG93] it was also shown that for every LFP-formula ψ there is an equivalent WF-Datalog query $(\Pi_\psi, q_{\boldsymbol{t}})$, i.e., LFP $\leq$ WF-Datalog.

The game $\mathcal{G}$ is played with a pebble by two players I and II in rounds. *Each round consists of two moves. Initially, I starts the game from the start position v_0. A player can move from x to y iff $(x, y) \in M$. A player loses in x, if she cannot move; she wins in x, if she can move to a position in which the opponent loses.*

A position $x \in V$ is won *for a player if the player can win every game starting at x, no matter how the opponent moves. Conversely, $x \in V$ is* lost *for a player if the opponent can always win the game starting at x, no matter how the player moves. A position x is* drawn *if x is neither lost nor won. $\mathcal{G} = (V, M, v_0)$ is won/lost/drawn if v_0 is won/lost/drawn for I.*

If x is won, the length *of x, denoted $|x|$, is the number of rounds which are necessary for I to win, provided both players play optimal (i.e., each player tries to win as quickly or to lose as slowly as possible). If x is lost or drawn, we let $|x| := \infty$.*

A game is called draw-free *if no position in V is drawn. Note that a game may be* determinate, *i.e., the start position v_0 is either lost or won, yet it may contain positions x which are drawn.*

Observe that the presence of cycles in M is necessary but not sufficient for the existence of drawn positions in $\mathcal{G}$. For example, if $M = \{(a, b), (b, a), (b, c)\}$ then b is won, whereas a and c are lost. If the move (c, d) is added to M then d is lost, c is won, and a and b are drawn.

Games have a very elegant and intuitive representation in WF-Datalog in the form of the famous *win-move* example. Indeed this example has always been used to demonstrate that WF-Datalog handles negation in a nice and intuitive way (but note that the WF-semantics and the NI-semantics coincide for this class of programs).

Definition 6 (WF-Datalog^G) *Let* WF-Datalog^G *be the class of WF-Datalog queries obtained from programs Π which have a single recursive rule of the form*

$$win(\bar{X}) \leftarrow move(\bar{X}, \bar{Y}), \neg win(\bar{Y})$$

where $\bar{X}$ and $\bar{Y}$ have the same arity ≥ 1, and a rule of the form

$$answer(\bar{V}) \leftarrow win(\bar{T})$$

where $\bar{V}$ are variables occurring in $\bar{T}$. All other rules of Π are nonrecursive, contain neither win nor answer, and are semipositive (i.e., negation is allowed only in front of edb relations).

Let WF-Datalog$^G_2$ *be the restriction of WF-Datalog^G to total programs and queries $(\Pi, q_{\boldsymbol{t}})$.*

Remark. Consider the following WF-Datalog^G program

$$\Pi_{Game}: \quad \begin{array}{l} win(X) \leftarrow move(X,Y), \neg win(Y) \\ answer \leftarrow win(v_0). \end{array}$$

Since every game $\mathcal{G} = (V, M, v_0)$ is a finite structure, it can be used as input to Π_{Game}. One easily verifies that Π_{Game} represents such games in the sense that

$$v_0 \in \left\{ \begin{array}{c} q_{\boldsymbol{t}} \\ q_{\boldsymbol{f}} \\ q_{\boldsymbol{u}} \end{array} \right\} \quad \Leftrightarrow \quad \mathcal{G} \text{ is } \left\{ \begin{array}{c} \text{won} \\ \text{lost} \\ \text{drawn} \end{array} \right\}.$$

4.2 Diagrams

As an auxiliary notation for games, we make use of *diagrams* as those depicted in Fig. 1. We assume that with every variable X we have associated a variable X' in a one-to-one fashion.

Definition 7 (Diagrams) *A* diagram *d consists of a finite set of* squares *and a finite set of (possibly labeled)* arrows *between squares. Every diagram contains exactly one distinguished* start square s_0. *In case an arrow is labeled, the label has one of the forms*

$$\text{“}\varphi\text{”}, \quad \text{“}\exists \bar{X}'\ \psi\text{”}, \quad \textit{or} \quad \text{“}\exists \bar{X}'\text{”}\ ,$$

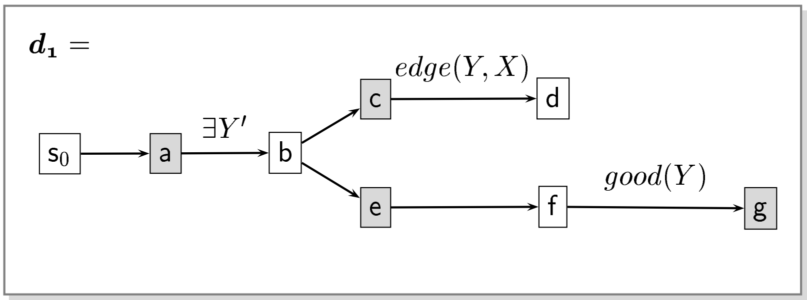
where

(1) φ and ψ are quantifier-free, and
(2) φ only contains unprimed variables.

If “$\exists \bar{X}'$” or “$\exists \bar{X}'\,\psi$” (for some ψ) occurs in d, then $\bar{X}$ are bound *variables of d. All other unprimed variables of d are called* parameters *of d.*

Example 8 *In Fig. 1, the diagram d_1 contains the bound variable Y and the parameter X, while d_2 has no parameters.*

Playing Games with Diagrams. Given a fixed database $\mathcal{D}$, every diagram d with parameters among $\bar{U}$ induces a game $\mathcal{G}_{d,\bar{u}}$ ($\bar{u}$ are the fixed domain values used for the variables $\bar{U}$). $\mathcal{G}_{d,\bar{u}}$ is played as follows:



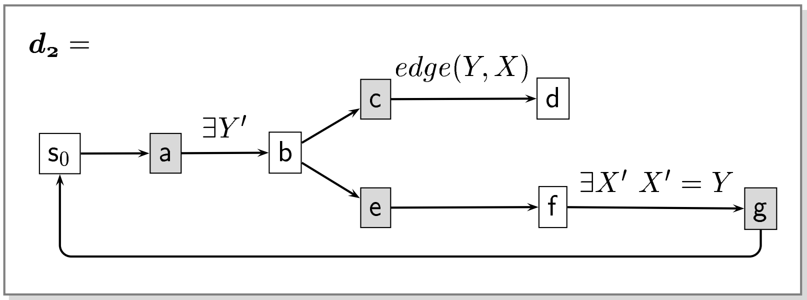


Fig. 1. Two diagrams

Let $Sq = \{\mathsf{s}_0, \ldots, \mathsf{s}_m\}$ be the squares and $\bar{X} = X_1, \ldots, X_n$ the bound variables of d. The game is played with n *domain pebbles* (lying on the current domain values $\bar{x} \in D^n$ assigned to $\bar{X}$) and an additional *square pebble* (lying on the current square $\mathsf{s}_i \in Sq$).

Initially, the square pebble is on the start square s_0 of d and the domain pebbles $\bar{X}$ are on a fixed element $c^{\mathcal{D}}$.[6] The players move alternately with player `I` starting the game. In each move, a player may move the pebbles according to the rules induced by the diagram: the square pebble has to be moved along an arrow of d. Additionally, the domain pebbles have to be moved in accordance with the constraints given by the labels:

More precisely, the positions of $\mathcal{G}_{d,\bar{u}}$ are

$$V = \{(\mathsf{s}, \bar{x}) \mid \mathsf{s} \in Sq, \bar{x} \in D^n\} \ .$$

The start position of $\mathcal{G}_{d,\bar{u}}$ is $(\mathsf{s}_0, \tilde{c})$ where s_0 is the start square of d. The moves between positions are given by the arrows in d: there is a move from $(\mathsf{s}, \bar{x})$ to $(\mathsf{s}', \bar{x}')$ in $\mathcal{G}_{d,\bar{u}}$ if

(1) there is an (unlabeled) arrow $\mathsf{s} \longrightarrow \mathsf{s}'$ in d and $\bar{x}' = \bar{x}$,
(2) there is an arrow $\mathsf{s} \xrightarrow{\varphi} \mathsf{s}'$ with quantifier-free φ such that $\mathcal{D} \models \varphi(\bar{x})$ and $\bar{x}' = \bar{x}$, or
(3) there is an arrow $\mathsf{s} \xrightarrow{\Theta} \mathsf{s}'$ where Θ contains quantifiers and

[6] To simplify the presentation, we assume that there is at least one constant c whose interpretation in $\mathcal{D}$ is $c^{\mathcal{D}}$.

(a) for all X_i such that X_i' is *not* $\exists$-quantified in Θ, we have $x_i' = x_i$, and
(b) if Θ contains a quantifier-free formula $\varphi(\bar{X}, \bar{X}')$, then $\mathcal{D} \models \varphi(\bar{x}, \bar{x}')$.

Theorem 9 *For every diagram d with bound variables $\bar{X}$ there is a WF-Datalog^G program Π_d with move relation $move(S, \bar{X},\ S', \bar{X}')$ such that for Π_d*

$$(\mathsf{s}, \bar{a}) \in \left\{ \begin{matrix} q_{\boldsymbol{t}} \\ q_{\boldsymbol{f}} \\ q_{\boldsymbol{u}} \end{matrix} \right\} \quad \Leftrightarrow \quad (\mathsf{s}, \bar{a}) \ \textit{is} \left\{ \begin{matrix} \textit{won} \\ \textit{lost} \\ \textit{drawn} \end{matrix} \right\} \textit{in } \mathcal{G}_{d,\bar{u}} \ .$$

Proof The translation is straightforward and should be clear from Fig. 2 which shows how d_2 from Fig. 1 is encoded.[7] □

$$move(\mathsf{s}_0, X, Y,\ \mathsf{a}, X', Y') \leftarrow X' = X,\ Y' = Y$$
$$move(\mathsf{a}, X, Y,\ \mathsf{b}, X', Y') \leftarrow X' = X$$
$$move(\mathsf{b}, X, Y,\ \mathsf{c}, X', Y') \leftarrow X' = X,\ Y' = Y$$
$$move(\mathsf{b}, X, Y,\ \mathsf{e}, X', Y') \leftarrow X' = X,\ Y' = Y$$
$$move(\mathsf{c}, X, Y,\ \mathsf{d}, X', Y') \leftarrow edge(Y, X),\ X' = X,\ Y' = Y$$
$$move(\mathsf{e}, X, Y,\ \mathsf{f}, X', Y') \leftarrow X' = X,\ Y' = Y$$
$$move(\mathsf{f}, X, Y,\ \mathsf{g}, X', Y') \leftarrow X' = Y,\ Y' = Y$$
$$move(\mathsf{g}, X, Y,\ \mathsf{s}_0, X', Y') \leftarrow X' = X,\ Y' = Y$$
$$win(S, X, Y) \leftarrow move(S, X, Y,\ S', X', Y'),\ \neg win(S', X', Y')$$
$$answer(V) \leftarrow win(\mathsf{s}_0, V, Y).$$

Fig. 2. Encoding of diagram d_2 in WF-Datalog^G

In the following, we show how one can find for every WF-Datalog query an equivalent query from WF-Datalog^G, i.e., in the form of a game. As a first step, we show how to encode first-order formulas as games:

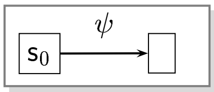
Theorem 10
For every first-order formula $\psi(\bar{U})$ there is a diagram d_ψ such that

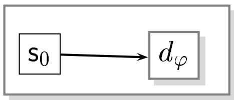
(1) I wins $\mathcal{G}_{d_\psi,\bar{u}} \ \Leftrightarrow\ \mathcal{D} \models \psi(\bar{u})$.

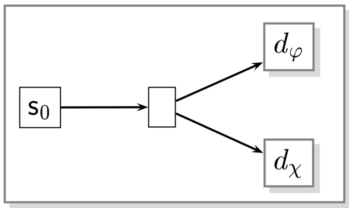
[7] Note that the constants $\mathsf{s}_0, \mathsf{a}, \mathsf{b}, \ldots$ denoting squares of a diagram do not belong to the database schema in question. Here and in the following, one can avoid such new constants $\mathsf{c}_1, \ldots, \mathsf{c}_r$ using new variables $Z_1, \ldots, Z_r$ and letting all tuples $\bar{z} \in D^r$ with $z_i \neq z_1 = \ldots z_{i-1} = z_{i+1} = \cdots = z_r$ take over the role of c_i.

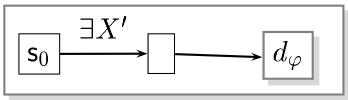
(2) $\mathcal{G}_{d_\psi,\bar{u}}$ is draw-free.

Proof We define d_ψ by induction on the structure of ψ. Note that s_0 denotes the start square of the corresponding diagram. In the inductive definition of diagrams, arrows pointing to a subdiagram are connected to the start square of this subdiagram.

- if ψ is atomic then $d_\psi :=$ 

  Clearly, I wins $\mathcal{G}_{d_\psi,\bar{u}}$ iff $\mathcal{D} \models \psi(\bar{u})$.
- if $\psi = \neg\varphi$ then $d_\psi :=$ 

  Note that I wins (loses) $\mathcal{G}_{d_\psi,\bar{u}}$ iff II loses (wins) $\mathcal{G}_{d_\varphi,\bar{u}}$. Here, we use that $\mathcal{G}_{d_\varphi,\bar{u}}$ is draw-free by induction hypothesis.
- if $\psi = \varphi \wedge \chi$ then $d_\psi :=$ 

  I wins $\mathcal{G}_{d_\psi,\bar{u}}$ iff she can win both the game for φ and the game for χ (if one of them is lost for I, then II could choose the corresponding arrow and win).
- if $\psi = \exists X\ \varphi$ then $d_\psi :=$ 

  In this case, I wins $\mathcal{G}_{d_\psi,\bar{u}}$ iff there is a value $a \in D$ such that I wins $\mathcal{G}_{d_\varphi,\bar{u}\,a}$.

The game is draw-free since the move graph of $\mathcal{G}_{d_\psi,\bar{u}}$ is acyclic. □

Note that the squares of the d_ψ can be 2-colored such that player I may only move along arrows from white to black squares, while II may only use arrows from black to white squares (cf. Fig. 1). Moreover, by the above construction, it is clear that if an atomic formula φ occurs only positively (negatively) in ψ, then arrows marked with "φ" can only be used by player I (II).

We want to extend part (1) of Theorem 10 to LFP-formulas. For this purpose, we need the following theorem which is due to Immerman [Imm86]:

Theorem 11 *Every LFP-formula is equivalent to a formula of the form*

$$[\mathrm{LFP}_{q(\bar{X})}\ \varphi]\tilde{c}$$

where φ is first-order.

We will use Theorem 11 to prove

Theorem 12 *For every LFP-formula $\psi(\bar{U})$ there is a diagram d_ψ such that*

$$\mathtt{I} \textit{ wins } \mathcal{G}_{d_\psi,\bar{u}} \;\Leftrightarrow\; \mathcal{D} \models \psi(\bar{u}).$$

Proof Let $\psi \in$ LFP. By Theorem 11 we may assume that

$$\psi(\bar{U}) = [\mathrm{LFP}_{q(\bar{X})}\ \varphi(\bar{U}, \bar{X})]\tilde{c}$$

where φ is first-order and positive in q. By Theorem 10 there is a diagram d_φ such that $\mathtt{I}$ wins $\mathcal{G}_{d_\varphi,\bar{u}\,\bar{x}} \Leftrightarrow \mathcal{D} \models \varphi(\bar{u}, \bar{x})$. We modify d_φ as follows to obtain the diagram d_ψ:

Let A be an arrow of d_φ which is marked with "$q(\bar{T})$" (q being the relation symbol bounded by the LFP-operator). Since $\psi \in$ LFP, q occurs only positively in ψ and by the construction of d_φ, A can only be used by player $\mathtt{I}$ (i.e., points from a white to a black square). In other words, $\mathtt{I}$ has the obligation to proof that $q(\bar{T})$ holds. This is achieved in the new diagram d_ψ as follows:

- The label of A is replaced by "$\exists \bar{X}'\ \bar{X}' = \bar{T}$", and
- there is a new unlabeled arrow from the black square in which A ends to the start of d_φ.

This construction is illustrated by Example 13 below and d_2 in Fig. 1.

Let q_k be the k-th iteration of q in ψ. By induction on k one easily verifies that:

- If $\bar{a} \in q_k$ then player $\mathtt{I}$ has a winning strategy for $\mathcal{G}_{d_\psi,\bar{a}}$ where she uses the start square at most k times. In the induction step, $\mathtt{I}$ uses a winning strategy for the game $\mathcal{G}_{d_\varphi,\bar{a}}$ in $(\mathcal{D}, q_{k-1})$, i.e., taking q_{k-1} as the interpretation of q in φ.

Conversely, if $\bar{a} \notin q_{\boldsymbol{t}}$ then player $\mathtt{II}$ can prevent $\mathtt{I}$ from winning $\mathcal{G}_{d_\psi,\bar{a}}$ by repeatedly using a winning strategy for $\mathcal{G}_{d_\varphi,\bar{a}}$ in $(\mathcal{D}, q_{\boldsymbol{t}})$.

Finally, let $\bar{a} = \tilde{c}^{\mathcal{D}}$ and the claim follows. $\square$

Note that in general, the game d_ψ constructed for a LFP-formula ψ contains drawn positions, i.e., if $\mathcal{D} \not\models \psi(\bar{u})$ then $\mathtt{II}$ can prevent $\mathtt{I}$ from winning but may not be able to win $\mathcal{G}_{d_\psi,\bar{u}}$ either.

Example 13 (Good Nodes [AHV95]) *Consider the LFP-formula*

$$\psi(U) = [\mathrm{LFP}_{good(X)}\ \underbrace{\forall Y\ (edge(Y,X) \to good(Y)}_{\varphi(X)}]U$$

It computes the "good nodes" of a directed graph, i.e., those that cannot be reached from a cycle. The diagram d_1 in Fig. 1 corresponds to the subformula $\varphi(X)$ of ψ (according to the proof of Theorem 10) and d_2 in Fig. 1 corresponds to ψ (according to the proof of Theorem 12).

By applying Theorem 12 and Theorem 9 we directly obtain

Corollary 14
Every LFP-formula is equivalent to a WF-Datalog^G query of the form $(\Pi, q_{\boldsymbol{t}})$.

Using Theorem 4 this implies

Corollary 15 (WF-Datalog $\leq$ WF-Datalog^G) *Every WF-Datalog query (of any of the forms $(\Pi, q_{\boldsymbol{t}})$, $(\Pi, q_{\boldsymbol{f}})$, or $(\Pi, q_{\boldsymbol{u}})$) is equivalent to a WF-Datalog^G query of the form $(\Pi, q_{\boldsymbol{t}})$. In particular, WF-Datalog $\leq$ WF-Datalog^G.*

Remark. In [FKL97] an alternative proof was given using a normal form for LFP which is due to Grohe [Gro94] and which allows a very simple translation into a game. In contrast, the proof presented above uses the normal form of Immerman [Imm86] which is more known and easier to obtain.

4.3 Reduction from Games to Draw-Free Games

In this section, we show that WF-Datalog^G $\leq$ WF-Datalog$^G_2$, i.e., for every WF-Datalog query, there is an equivalent query in WF-Datalog$^G_2$. By Corollary 15 every WF-Datalog program corresponds to a game. It remains to show that for each such game, an equivalent draw-free game can be constructed and represented by a WF-Datalog^G program.

Theorem 16 (WF-Datalog^G $\leq$ WF-Datalog$^G_2$)
For every WF-Datalog^G query there is an equivalent query in WF-Datalog$^G_2$.

Proof First, we present an informal proof emphasizing the idea of the construction. [8] Technical details are given afterwards.

[8] The reduction presented is due to [Kub95] which also contains the details of a proof of a normal form for LFP implying Theorem 16.

The main problem consists in detecting and avoiding drawn positions. In the absence of an order on the domain it seems particularly difficult to limit the length of the game in order to eliminate drawn positions, e.g. we cannot use a counter for that purpose.

The basic idea is to limit the length of a game by comparing it to a game of maximal length. Two games are compared by playing them independently but synchronously. Thus, we construct a new game $2\mathcal{G}$ which simulates these two games on the original structure $\mathcal{G}$. To do so, we need two pebbles—one for each game in $\mathcal{G}$. Call these the *clock pebble* $\bar{Y}$ (on position $\bar{y}$ in $\mathcal{G}$) and the *verify pebble* $\bar{X}$ (on position $\bar{x}$ in $\mathcal{G}$).[9] The game played with the clock pebble is used to limit the length of the game played with the verify pebble. The latter plays the role of the pebble in the original game $\mathcal{G}$.

Initially, player I claims that the verify pebble is on a won position, i.e., $|\bar{x}| < \infty$ (cf. Definition 5). II places the clock pebble on $\bar{y}$ and claims that $|\bar{y}|$ is the maximal length of a won position in the game. If this is true, I and II can compare $|\bar{x}|$ and $|\bar{y}|$ and thus verify the original claim of I. The difficulty remains that both players have to agree upon the choice of $\bar{y}$. To solve this, one has to design $2\mathcal{G}$ in such a way, that II can be disproved if she "cheats" by choosing a $\bar{y}$ which is not maximal.

The new game $2\mathcal{G}$ is constructed as follows (cf. Fig. 4): We use two macros $1\,\mathsf{round}(\bar{X})$ and $1\,\mathsf{round}(\bar{Y})$ to denote a round of moves of the pebbles on $\bar{x}$ and $\bar{y}$ in $\mathcal{G}$, respectively (Fig. 3). Note that in the simulated game $\mathcal{G}$, I moves first in $1\,\mathsf{round}(\bar{X})$ while II moves first in $1\,\mathsf{round}(\bar{Y})$.

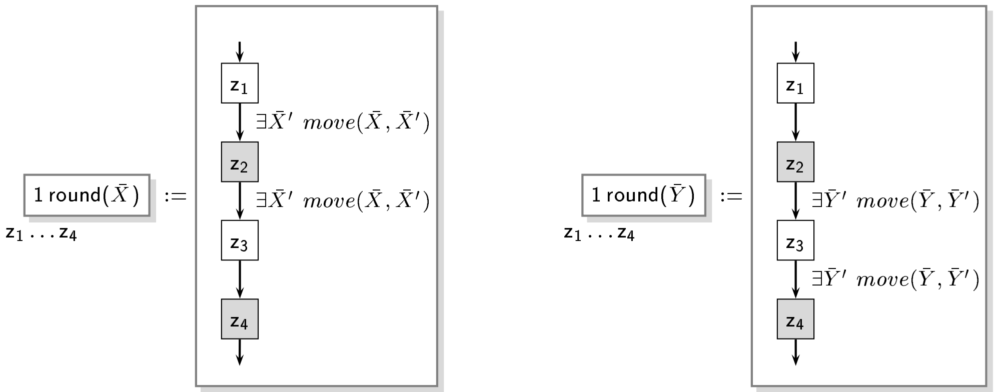
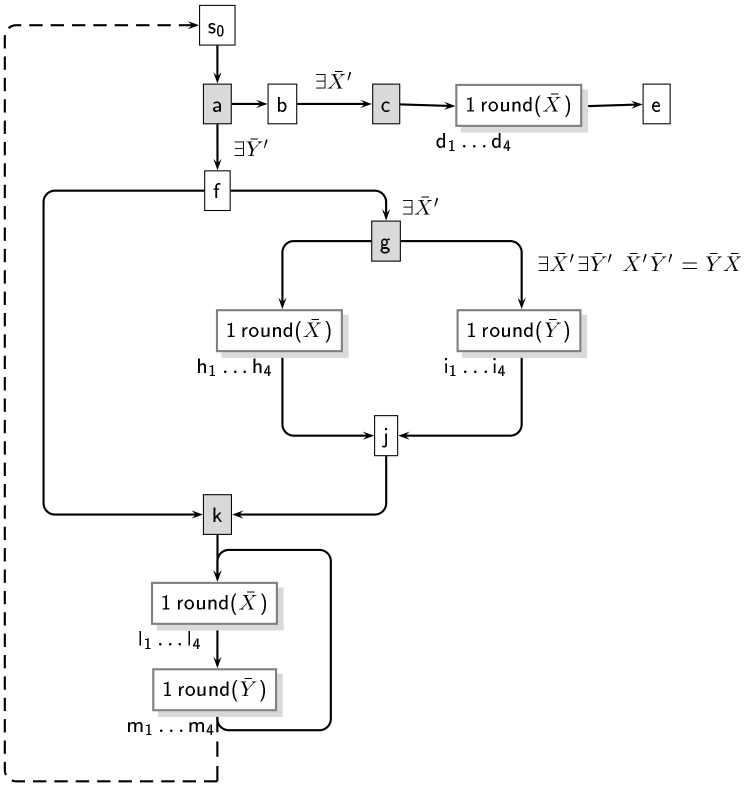


Fig. 3. Macro definitions

Like above, the diagram in Fig. 4 defines a set of semipositive nonrecursive rules for the new relation $move(S, \bar{X}, \bar{Y},\ S', \bar{X}', \bar{Y}')$. Thus, if the move relation of the original game $\mathcal{G}$ (used in the macros of Fig. 3) is n-ary, the new move

[9] Thus, we abstract from the fact that $\bar{X}$ is a *tuple* of pebbles, and simply call $\bar{X}$ a pebble in $2\mathcal{G}$; analogously for $\bar{Y}$.



Player I	
$s_0 \to a$	$\lvert\bar{x}\rvert < \infty$, "the length of $\bar{x}$ is finite, i.e., $\bar{x}$ is won"
$b \to c$	$\exists \bar{x} : \lvert\bar{x}\rvert = 1$, "I show you a new $\bar{x}$ which is won in 1 round"
$f \to k$	$\lvert\bar{x}\rvert \leq \lvert\bar{y}\rvert$, "$\bar{y}$ is not shorter than $\bar{x}$"
$f \to g$	$\lvert\bar{y}\rvert < \infty, \exists \bar{x} : \lvert\bar{x}\rvert = \lvert\bar{y}\rvert + 1$, "$\bar{y}$ is finite, but not of maximal length: I show you a new $\bar{x}$ which is 1 round longer"
$j \to k$	$\lvert\bar{x}\rvert \leq \lvert\bar{y}\rvert$, "$\bar{y}$ is not shorter than $\bar{x}$"
$l_1 \to l_2$	$\lvert\bar{x}\rvert \leq \lvert\bar{y}\rvert$, "I can win in time"

Player II	
$a \to b$	$\neg \exists \bar{x} : \lvert\bar{x}\rvert < \infty$, "there is no $\bar{x}$ in $\mathcal{G}$ which is won"
$a \to f$	$\exists \bar{y} : \lvert\bar{y}\rvert < \infty, \lvert\bar{y}\rvert$ maximal, $\lvert\bar{x}\rvert > \lvert\bar{y}\rvert$, "$\bar{y}$ is finite, of maximal length and shorter than $\bar{x}$"
$k \to l_1$	$\lvert\bar{x}\rvert > \lvert\bar{y}\rvert$, "you can't win on $\bar{x}$ in time"
$g \to h_1$	$\lvert\bar{x}\rvert > \lvert\bar{y}\rvert + 1$, "$\bar{x}$ is more than 1 round longer than $\bar{y}$: I give you a lead of 1 round $\bar{x}$ and you lose"
$g \to i_1$	$\lvert\bar{x}\rvert \leq \lvert\bar{y}\rvert$, "your chosen $\bar{x}$ is not longer than my $\bar{y}$: let us swap the pebbles on $\bar{x}$ and $\bar{y}$ and give me a lead of 1 round in $\bar{y}'$: then you lose on $\bar{x}$ against the clock"

Fig. 4. Draw-free game $2\mathcal{G}$ and implicit claims of I and II.

relation of $2\mathcal{G}$ is $2(n+1)$-ary. For a given answer relation $answer(\bar{V}) \leftarrow win(\bar{X})$ in $\Pi_{\mathcal{G}}$, the new answer relation of $\Pi_{2\mathcal{G}}$ is defined as

$$answer(\bar{V}) \leftarrow win(\mathsf{s}_0, \bar{X}, \bar{Y}).$$

It is easy to see that $\mathtt{I}$ wins in $1\,\mathsf{round}(\bar{X})$ if $|\bar{x}| = 1$, and $\mathtt{II}$ wins $1\,\mathsf{round}(\bar{Y})$ if $|\bar{y}| = 1$.

Assume for the moment that the dashed edge from m_4 to s_0 in Fig. 4 is absent. The loop $\mathsf{l}_1 \to \mathsf{m}_4 \to \mathsf{l}_1$ compares the lengths of $\bar{x}$ and $\bar{y}$: $\mathtt{I}$ wins this comparison if $|\bar{x}| < \infty$ and $|\bar{x}| \leq |\bar{y}|$, while $\mathtt{II}$ wins if $|\bar{y}| < \infty$ and $|\bar{y}| < |\bar{x}|$.

To get a better understanding of the construction of $2\mathcal{G}$, we explain the diagram in Fig. 4 as a dialog between $\mathtt{I}$ and $\mathtt{II}$, where each move corresponds to a claim of the moving player. Observe that each claim of a player contradicts the previous claim of the opponent, and that each false claim can indeed be disproved using the corresponding moves in the diagram.

Using the diagram and the implicit claims of the players, it should be clear that $\mathtt{I}$ wins $(\mathsf{s}_0, \bar{x}, \bar{y})$ in $2\mathcal{G}$ (for arbitrary $\bar{y}$) if $\mathtt{I}$ wins $\bar{x}$ in $\mathcal{G}$, and $\mathtt{II}$ wins $(\mathsf{s}_0, \bar{x}, \bar{y})$ in $2\mathcal{G}$ if $\bar{x}$ is lost or drawn (for $\mathtt{I}$) in $\mathcal{G}$. Thus the new game $2\mathcal{G}$ is *determinate* for positions $(\mathsf{s}_0, \bar{x}, \bar{y})$.

However $2\mathcal{G}$ may still contain positions which are drawn: Consider, for example, $(\mathsf{l}_1, \bar{x}, \bar{y})$ where $\bar{x}$ and $\bar{y}$ are drawn in $\mathcal{G}$. Then $\mathtt{II}$ gets no chance of refuting the claim that $\bar{x}$ is won in $\mathcal{G}$, hence $(\mathsf{l}_1, \bar{x}, \bar{y})$ is also drawn in $2\mathcal{G}$. In order to allow $\mathtt{II}$ to defeat such false claims, the dashed edge is needed. By moving along $\mathsf{m}_4 \to \mathsf{s}_0$, $\mathtt{II}$ can win and refute $\mathtt{I}$ by choosing the maximal $\bar{y}$ in the move $\mathsf{a} \to \mathsf{f}$.

The final obstacle is that one has to verify that if $\bar{x}$ is won in $\mathcal{G}$, then $\mathtt{II}$ cannot delay the game infinitely using the edge $\mathsf{m}_4 \to \mathsf{s}_0$. Indeed $|\bar{x}|$ decreases each time the game reaches m_4: If $\mathtt{II}$ chooses in a some $\bar{y}$ with $|\bar{y}| \geq |\bar{x}|$, then $\mathtt{I}$ has to move along $\mathsf{f} \to \mathsf{k}$ thereby enforcing that at least $1\,\mathsf{round}(\bar{X})$ is played. Otherwise, if $\mathtt{II}$ chooses $|\bar{y}| < |\bar{x}|$, then $\mathtt{I}$ chooses a new $\bar{x}$ with $|\bar{x}| = |\bar{y}| + 1$. Independent of the choice of $\mathtt{II}$ ($\mathsf{g} \to \mathsf{h}_1$ or $\mathsf{g} \to \mathsf{i}_1$), the new $\bar{x}$ will be at least one smaller, when m_4 is reached.

We turn to the formal proof. In the sequel, if we say that *"player $\mathtt{I}$ ($\mathtt{II}$) achieves ..."*, we mean that there is a strategy such that either player $\mathtt{I}$ ($\mathtt{II}$) wins, or situation "..." occurs.

The following lemmas are immediate:

L1: Let $|\bar{x}| \leq k$ and $\mathtt{I}$ starts to move (from the first square) in a subdiagram $1\,\mathsf{round}(\bar{X})$. Then $\mathtt{I}$ achieves that $|\bar{x}| \leq k-1$ on the exit square (of $1\,\mathsf{round}(\bar{X})$). In particular, $\mathtt{I}$ wins if $k=1$.

L2: Let $|\bar{x}| \geq k > 1$ and $\mathtt{I}$ starts in a subdiagram $1\,\mathsf{round}(\bar{X})$. Then $\mathtt{II}$ achieves that $|\bar{x}| \geq k-1$ on the exit square. If $|\bar{x}| = \infty$, $\mathtt{II}$ achieves that $|\bar{x}| = \infty$ on the exit square.

L3: Let $|\bar{y}| \leq k$ and $\mathtt{I}$ starts in a subdiagram $1\,\mathsf{round}(\bar{Y})$. Then $\mathtt{II}$ achieves that $|\bar{y}| \leq k-1$ on the exit square. In particular, $\mathtt{II}$ wins if $k=1$.

L4: Let $|\bar{y}| \geq k > 1$ and $\mathtt{I}$ starts in a subdiagram $1\,\mathsf{round}(\bar{Y})$. Then $\mathtt{I}$ achieves that $|\bar{y}| \geq k-1$ on the exit square. If $|\bar{y}| = \infty$, $\mathtt{I}$ achieves that $|\bar{y}| = \infty$ on the exit square.

L5: $(\mathsf{d}_1, \bar{x}, \bar{y})$ is won $\Leftrightarrow |\bar{x}| = 1$.

L6: $(\mathsf{b}, \bar{x}, \bar{y})$ is won $\Leftrightarrow \exists \bar{x}\ |\bar{x}| < \infty$. (Note that $\exists \bar{x}\ |\bar{x}| < \infty \ \Leftrightarrow\ \exists \bar{x}\ |\bar{x}| = 1$.)

The following lemma is shown by induction on k:

L7: Let $|\bar{x}| \leq k$. We simultaneously show that $(\mathsf{s}, \bar{x}, \bar{y})$ is won ...

(i) ... if $\mathsf{s} = \mathsf{l}_1$ and $|\bar{y}| \geq k$: $\mathtt{I}$ achieves that if m_4 is reached, then $|\bar{x}| \leq k-1$ and $|\bar{y}| \geq k-1$. If $\mathtt{II}$ moves back to l_1, then $\mathtt{I}$ wins by the induction hypothesis; if $\mathtt{II}$ moves to s_0, then $\mathtt{I}$ wins using (v) for $k-1$.

(ii) ... if $\mathsf{s} = \mathsf{h}_1$ and $|\bar{y}| \geq k-1$: $\mathtt{I}$ achieves that if h_4 is reached, then $|\bar{x}| \leq k-1$ and wins by (i) for $k-1$.

(iii) ... if $\mathsf{s} = \mathsf{i}_1$ and $|\bar{y}| > k$: $\mathtt{I}$ achieves that if i_4 is reached, then $|\bar{y}| \geq k$ and wins using (i).

(iv) ... if $\mathsf{s} = \mathsf{f}$: If $|\bar{y}| \geq k$ then $\mathtt{I}$ moves to k and wins by (i). Otherwise, if $|\bar{y}| < k$ then $\mathtt{I}$ moves to g and chooses some $\bar{x}$ such that $|\bar{x}| = |\bar{y}|+1$. This is possible since by assumption $|\bar{x}| = k > |\bar{y}|$, hence $\bar{y}$ is not maximal. Now if $\mathtt{II}$ moves to h_1 then $\mathtt{I}$ wins by (ii) for some smaller k. If $\mathtt{II}$ moves to i_1 instead, then after the move $|\bar{y}| = |\bar{x}| + 1$ (since the variables have been swapped!) and $\mathtt{I}$ wins by (iii).

(v) ... if $\mathsf{s} = \mathsf{s}_0$: $\mathtt{I}$ moves to a. If $\mathtt{II}$ moves to b, $\mathtt{I}$ wins by L6 above; otherwise, if $\mathtt{II}$ moves to f then $\mathtt{I}$ wins by (iv).

We need two final lemmas:

L8: If $|\bar{y}| = k$, $|\bar{x}| \geq k+1$ and $\mathtt{I}$ starts to move from $(\mathsf{l}_1, \bar{x}, \bar{y})$, then $\mathtt{II}$ wins:

By induction on k, after passing l and m we have: $|\bar{y}| \leq k-1$, $|\bar{x}| \geq k$, so $\mathtt{II}$ moves back to l and wins by the induction hypothesis.

L9: If $|\bar{x}| = \infty$, then $\mathtt{II}$ wins from $(\mathsf{a}, \bar{x}, \bar{y})$:

If no position is won, i.e. $|\bar{x}| = \infty$ for all $|\bar{x}|$ then $\mathtt{II}$ moves to b and wins by L6. Otherwise, $\mathtt{II}$ moves to f choosing some $\bar{y}$ such that $|\bar{y}|$ is maximal. There are three cases:

(a) $\mathtt{I}$ moves to k: then $\mathtt{II}$ wins by L8.

(b) $\mathtt{I}$ moves to g choosing $\bar{x}$ s.t. $|\bar{x}| = \infty$: then $\mathtt{II}$ moves to h. Since

$|\bar{x}| = \infty$, $\mathtt{II}$ achieves that after h, still $|\bar{x}| = \infty$, hence wins using L8.
(c) $\mathtt{I}$ moves to g choosing $\bar{x}$ s.t. $|\bar{x}| = k < \infty$: since $\bar{y}$ is maximal, $|\bar{x}| \leq |\bar{y}|$. $\mathtt{II}$ moves to i after which $|\bar{y}| \leq |\bar{x}|$ (recall that the variables are swapped!). Thus after i, $\mathtt{II}$ achieves $|\bar{y}| \leq k-1$ and wins by L8.

Summarizing, this shows that (for arbitrary $\bar{y}$)

- $\mathtt{I}$ wins $(\mathsf{s}_0, \bar{x}, \bar{y})$ in $2\mathcal{G}$ iff $\bar{x}$ is won in $\mathcal{G}$ (use L7-(v)), and
- $\mathtt{II}$ wins $(\mathsf{a}, \bar{x}, \bar{y})$ in $2\mathcal{G}$ iff $\bar{x}$ is lost or drawn in $\mathcal{G}$ (use L9), and
- no positions $(\mathsf{s}, \bar{x}, \bar{y})$ in $2\mathcal{G}$ are drawn (use L7-(i) and L8). □

Putting everything together, we have

$$\text{WF-Datalog} \overset{\text{Corollary 15}}{\leq} \text{WF-Datalog}^G \overset{\text{Theorem 16}}{\leq} \text{WF-Datalog}_2^G \leq \text{WF-Datalog}$$

which proves

Corollary 17 (WF-Datalog $\equiv$ WF-Datalog$_2$)
For every WF-Datalog query, there is an equivalent query in WF-Datalog$_2$.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley, 1995.

[AS91] S. Abiteboul and E. Simon. Fundamental Properties of Deterministic and Nondeterministic Extensions of Datalog. *Theoretical Computer Science*, 78(1):137–158, 1991.

[AV91] S. Abiteboul and V. Vianu. Datalog Extensions for Database Queries and Updates. *Journal of Computer and System Sciences*, 43(1):62–124, 1991.

[EF95] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Perspectives in Mathematical Logic. Springer, 1995.

[FKL97] J. Flum, M. Kubierschky, and B. Ludäscher. Total and Partial Well-Founded Datalog Coincide. In F. Afrati and P. Kolaitis, editors, *Proc. 6th Intl. Conference on Database Theory (ICDT)*, number 1186 in LNCS, pp. 113–124, Delphi, Greece, 1997. Springer.

[Gar70] M. Gardner. The Game of Life. *Scientific American*, 223, October 1970.

[Gro92] M. Grohe. Fixpunktlogiken in der endlichen Modelltheorie. Master's thesis, Universität Freiburg, 1992.

[Gro94] M. Grohe. *The Structure of Fixed-Point Logics*. PhD thesis, Universität Freiburg, 1994.

[Imm86] N. Immerman. Relational Queries Computable in Polynomial Time. *Information and Control*, 68:86–104, 1986.

[Kub95] M. Kubierschky. Remisfreie Spiele, Fixpunktlogiken und Normalformen. Master's thesis, Universität Freiburg, 1995.

[VG93] A. Van Gelder. The Alternating Fixpoint of Logic Programs with Negation. *Journal of Computer and System Sciences*, 47(1):185–221, 1993.