6th Intl. Conference on Database Theory (ICDT)
Delphi, Greece, January 8–10, 1997
LNCS 1186, Springer

Total and Partial Well-Founded Datalog Coincide

Jörg Flum[I] Max Kubierschky[I] Bertram Ludäscher[II]

[I] Mathematische Fakultät, Universität Freiburg,
Eckerstr. 1, 79104 Freiburg, Germany
{flum,maku}@ruf.uni-freiburg.de

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

Abstract. We show that the expressive power of well-founded Datalog does not decrease when restricted to total programs (it is known to decrease from Π^1_1 to Δ^1_1 on infinite Herbrand structures) thereby affirmatively answering an open question posed by Abiteboul, Hull, and Vianu [AHV95]. In particular, we show that for every well-founded Datalog 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. This yields a nice new normal form for well-founded Datalog and implies that it is sufficient to consider draw-free games in order to evaluate arbitrary Datalog programs under the well-founded semantics.

1 Introduction

The well-founded semantics (WFS) [VGRS88, VG93] has become popular as an intuitive and "well-behaved"[1] semantics for the language of logic programs containing negative cyclic dependencies, like the famous program P_{game}:

$$win(X) \leftarrow move(X, Y), \neg\, win(Y).$$

A position X in a game is won, if there is a move to some position Y which is not won (since then the opponent has to move). WFS assigns a partial (3-valued) model $\mathrm{WFS}(P, D)$ to every logic program P and database D. The third truth-value *undefined* is assigned if the truth of an atom A depends negatively on itself and there is no other "well-founded" derivation for A leading to true. Consider for example a move graph for P_{game} consisting of the edges $move(a, b), move(b, a), move(b, c)$ and $move(c, d)$. Under the well-founded

[1] Dix [Dix95] formally defines this notion using certain abstract properties, and shows that WFS is the weakest well-behaved extension of the generally accepted stratified semantics [ABW88].

semantics, $win(d)$ is false, since there are no moves from d. Consequently, $win(c)$ is true, since it is possible to move from c to d. On the other hand, $win(a)$ and $win(b)$ are undefined since a is won iff b is not won, and b is won iff a is not won. This corresponds nicely to the fact, that the positions a and b in the game are *drawn*: the player moving from b has no winning strategy (moving to c would leave the opponent in a won position), but she can enforce a game of infinite length by moving from b to a and thus avoid losing.

Another aspect of languages which has always played an important role in database theory is expressive power, i.e., the class of queries definable in a language. The query associated with a logic program P and database D is defined in terms of the *true* atoms of $\mathrm{WFS}(P, D)$ (hence *undefined* and *false* atoms belong to the complement of the query). Van Gelder showed that Datalog evaluated under WFS is equivalent to (least) fixpoint logic [VG89, VG93].

A natural question arising is: What is the expressive power of programs which never yield the truth value *undefined*, i.e. which are *total* for all databases D? For logic programs over infinite Herbrand structures, the restriction to total programs results in loss of expressive power from Π^1_1 to Δ^1_1 [Sch95]. For Datalog programs, the question has been posed by Abiteboul et. al. [AHV95] and remained open so far. We give the somewhat surprising[2] answer that on finite structures, i.e. for well-founded Datalog, there is no loss of expressive power.[3]

As it turns out, games play a crucial role in our solution: using a normal form for fixpoint logic, we first show that every Datalog program can be viewed as a game between two players. Thus, the ubiquitous *win-move* example is raised retroactively to an elegant normal form for well-founded Datalog. The drawn positions of the game are exactly the undefined atoms of the well-founded model. The second result is that for every game one can find an equivalent game which is *draw-free*, i.e. all positions in the game are either won or lost. This implies, that total well-founded Datalog and well-founded Datalog have the same expressive power.

The paper is structured as follows. In Section 2 the required concepts and terminology are briefly introduced. They are based on [AHV95], [VG93, AB94] (for WFS) and [EF95] (for fixpoint logic). In Section 3 we first introduce games and then show our main result, the reduction of games to draw-free games.

---

[2] Indeed, in [VG93] van Gelder writes: *"This suggests that the alternating fixpoint on normal programs captures the negation of positive existential closures (such as transitive closure), but not the negation of positive universal closures (such as well-foundedness)."* The generalization of WFS to *general logic programs*, i.e., with first-order rule bodies avoids – at least for some examples – undefined atoms in the well-founded model (cf. [Che95]).

[3] In [Kub95], the second author has obtained a normal form theorem for LFP. Rewriting programs as logic formulas, the first author realized that Kubierschky's result can be used to solve the problem by Abiteboul et. al. The present exposition is due to the third author.

2 Preliminaries

A *database schema* (or *relational schema*) σ is a finite set of relation symbols $r_1, \ldots, r_k$ with associated arities $\alpha(r_i) \geq 0$. Let *dom* be a fixed and countable underlying domain. A *database instance* (*database*) over σ is a finite structure $D = (U, r_1^D, \ldots, r_k^D)$ with finite universe $U \subseteq dom$ and relations $r_i^D \subseteq U^{\alpha(r_i)}$.

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) $q(D)$ is a k-ary relation on U, and (ii) q is preserved under isomorphisms, i.e. for every isomorphism π of D, $q(\pi(D)) = \pi(q(D))$. 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 $\mathcal{L}_2 \leq \mathcal{L}_1$. $\mathcal{L}$ may denote both the language and the class of queries definable in it.

Notation. Following logic programming notation, we write domain variables in upper case like X, X', Y. Constants like x, y, a and relation symbols like $win, move$ are denoted in lower case.

$\bar{T}$ denotes a vector of n terms $T_1, \ldots, T_n$ (variables or constants).

$\widetilde{T}$ denotes n-ary repetition of T, i.e. a vector $T, T, \ldots, T$.

We write $\varphi(\bar{X})$ to emphasize that all free variables of φ are among $\bar{X}$; if we write only φ, nothing is said about the free variables of φ.

Well-Founded Datalog. A *Datalog*$^\neg$ program P is a finite set of rules of the form

$$H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m$$

where the head H is an atom, all B_i, C_j are atoms or equalities $T_1 = T_2$ where T_1, T_2 are terms. A rule where $n = m = 0$ is called a *fact*.

The signature σ_P of P is partitioned into a set $idb(P)$ of relation symbols of P occurring in some head of P and $edb(P)$ of relation symbols occurring only in the bodies of rules.

Fix a program P and a database D over $edb(P)$. A *ground instance* of a rule is obtained by substituting constants from D for all variables; $ground(P, D)$ denotes the set of all such ground instances of rules of P, and $\mathcal{B}_{P,D}$ denotes the set of all ground instances of atomic formulas of P.

Let $Y \subseteq \mathcal{B}_{P,D}$. For $X \subseteq \mathcal{B}_{P,D}$ let

$$\begin{aligned} \mathrm{T}_P^Y(X) := \{H \mid\ & (H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m) \in ground(P, D) \\ & \text{with } (B_i \in D \text{ or } B_i \in X) \text{ for all } 1 \leq i \leq n \\ & \text{and } C_j \notin Y \text{ for all } 1 \leq j \leq m\} \end{aligned}$$

Then T_P^Y is a monotone operator. Let $\Gamma_P(Y) := \mathrm{lfp}(\mathrm{T}_P^Y)$ be its least fixpoint. The operator Γ_P is antimonotone (observe how Y is used in T_P^Y), i.e., $Y_1 \subseteq Y_2$ implies $\Gamma_P(Y_2) \subseteq \Gamma_P(Y_1)$. It follows that Γ_P^2 ($:= \Gamma_P \circ \Gamma_P$) is a monotone operator; thus it has a least and a greatest fixpoint $\mathrm{lfp}(\Gamma_P^2)$ and $\mathrm{gfp}(\Gamma_P^2)$. These are used to define the truth value of a ground atom A under the *well-founded semantics* $\mathrm{WFS}(P, D)$ for a given program P and database D:

$$\mathrm{WFS}(P,D)(A) := \begin{cases} \mathit{true} & \text{if } A \in \mathrm{lfp}(\Gamma_P^2) \\ \mathit{false} & \text{if } A \notin \mathrm{gfp}(\Gamma_P^2) \\ \mathit{undef} & \text{if } A \in \mathrm{gfp}(\Gamma_P^2) \setminus \mathrm{lfp}(\Gamma_P^2) \end{cases} \qquad (\star)$$

Definition 2.1 (W-Datalog, W-Datalog$_2$)
A program P is called *total* (or *2-valued*) if *for all databases* D there is no ground atom A with $\mathrm{WFS}(P, D)(A) = \mathit{undef}$.

Let W-Datalog denote the set of Datalog$^\neg$ programs evaluated under the well-founded semantics, W-Datalog$_2$ is the set of *total* W-Datalog programs. □

Using the true atoms of the well-founded semantics, P defines for every relation $r \in \mathit{idb}(P)$ a query $q_{P,r}$ over $\mathit{edb}(P)$:

$$q_{P,r} : \ D \mapsto \{\bar{x} \mid \mathrm{WFS}(P,D)(r(\bar{x})) = \mathit{true}\} \qquad (\star\star)$$

We may assume w.l.o.g. that P contains one distinguished relation symbol $\mathit{answer} \in \mathit{idb}(P)$. This uniquely associates a query with every Datalog program P.

Least Fixpoint Logic. Let FO be the set of first-order formulas. By closing FO under least fixpoints of positive formulas we obtain *least fixpoint logic* LFP. The set of LFP-formulas is given by the following rules:[4]

$$\frac{}{\varphi} \text{ if } \varphi \text{ is an atom} ; \quad \frac{\varphi}{\neg\varphi} ; \quad \frac{\varphi, \psi}{\varphi \wedge \psi} ; \quad \frac{\varphi}{\exists X \varphi} ; \quad \frac{\varphi}{[\mathrm{LFP}_{R(\bar{X})}\ \varphi]\bar{V}} \text{ if } (+)$$

where $(+)$ is the proviso that the inductive relational variable R occurs only *positively* (i.e., under an even number of negations) in φ, and $\bar{X}, \bar{V}$ are $\alpha(R)$-ary tuples of variables.[5] The semantics of LFP-formulas is given by a relation $D \models \psi$ as usual. In particular, for $\psi(\bar{U}, \bar{V}) = [\mathrm{LFP}_{R(\bar{X})}\ \varphi(R, \bar{X}, \bar{U})]\bar{V}$ and $\bar{u}, \bar{v}$ in D:

$$D \models \psi(\bar{u}, \bar{v}) \ :\Leftrightarrow \ \bar{v} \in R_{\bar{u}}^\infty$$

where $R_{\bar{u}}^0 := \emptyset$, $R_{\bar{u}}^{i+1} := \{\bar{x} \mid D \models \varphi(R_{\bar{u}}^i, \bar{x}, \bar{u})\}$ and $R_{\bar{u}}^\infty := \bigcup_{i \in \mathbb{N}} R_{\bar{u}}^i$.

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

$$q_\psi : \ D \mapsto \{\bar{x} \mid D \models \psi(\bar{x})\}$$

Using $(\star)$ and $(\star\star)$ one can easily show W-Datalog $\leq$ LFP.

[4] In fixpoint formulas, relational variables are denoted in upper case.

[5] $\forall, \vee$ and $\rightarrow$ are viewed as abbreviations. For notational convenience, we only consider variables $\bar{V}$ instead of terms $\bar{T}$ in the last rule.

3 A Normal Form for Well-Founded Datalog

Abiteboul et. al. raised the question whether one can find for each W-Datalog program an equivalent *total program* [AHV95, pp. 397,401,403]. In other words, is W-Datalog $\leq$ W-Datalog$_2$? (W-Datalog$_2$ $\leq$ W-Datalog holds trivially.) When restricted to *ordered databases*, this is obviously the case, since S-Datalog (stratified Datalog) is equivalent to LFP (and thus captures PTIME) on ordered databases, and WFS is 2-valued for S-Datalog (see e.g. [AHV95]).

As we will show, the question can also be answered affirmatively in the absence of order. First, using results of van Gelder [VG89] and Grohe [Gro94], we show that every W-Datalog program can be transformed into a normal form which corresponds to a certain game. The main result is that one can reduce such games to draw-free games, which is equivalent to the fact that the corresponding W-Datalog program is total.

3.1 Games

A *game* is a finite structure $G = (V, move^G)$ with signature $\sigma = \{move\}$ and universe V. V are the *positions* (or *vertices*), $move^G \subseteq V \times V$ the set of possible *moves*.

The game 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 some position x_0. A player can move from x to y iff $(x, y) \in move^G$. 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 I) if I can always win the game starting at x, no matter how II moves. Conversely, $x \in V$ is *lost* (for I) if II can always win the game, no matter how I moves. A position x is *drawn* if x is neither lost nor won. Observe that the presence of cycles in $move^G$ is necessary but not sufficient for the existence of drawn positions in G.

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, let $|x| = \infty$. A game is called *draw-free* if no position in V is drawn.

Games have a very elegant and intuitive representation in W-Datalog in the form of the famous *win-move* example. Indeed this example has always been used to demonstrate that WFS handles negation in a nice and intuitive way.

Definition 3.1 (W-Datalog^G) Let W-Datalog^G be the class of W-Datalog programs P which have a single recursive rule of the form

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

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

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

where $\bar{U}$ are variables occurring in $\bar{T}$. All other rules of P are nonrecursive, contain neither *win* nor *answer*, and are *semipositive*, i.e., negation is allowed only in front of *edb* relations.[6]

Let W-Datalog_2^G be the set of *total* programs in W-Datalog^G. □

The simplest program in W-Datalog^G is P_G:

$$win(X) \leftarrow move(X, X'), \neg\, win(X').$$
$$answer \leftarrow win(x_0).$$

One easily verifies that P_G represents games, i.e., for every game $G = (V, move^G)$, x_0 is won/lost/drawn in G iff $\text{WFS}(P_G, G)(answer) = true/false/undef$.

We use the following theorems to show that an arbitrary W-Datalog program can be transformed into a W-Datalog^G program:

Theorem 3.2 (W-Datalog ≡ LFP, [VG89])
For every W-Datalog program there is an equivalent LFP-formula and vice versa.

Theorem 3.3 (Bounded Skolem Normal Form, [Gro94])
Every LFP-formula with free variables $\bar{U}$ is equivalent to a formula $\psi(\bar{U})$ of the form

$$\exists V[\text{LFP}_{W(\bar{X})}\ \varphi_0(\bar{X}, \bar{U}) \vee \exists \bar{Y} \forall \bar{Z}(\varphi(\bar{X}, \bar{Y}, \bar{Z}, \bar{U}) \rightarrow W(\bar{Z}))]\widetilde{V}$$

where φ_0, φ are quantifier-free first-order formulas not containing W.

Theorem 3.2 and Theorem 3.3 imply that for every W-Datalog program P, there is an equivalent LFP-formula ψ in bounded Skolem normal form. In the sequel, we show how to obtain an equivalent program $P_\psi \in \text{W-Datalog}^G$ by viewing ψ as a game.

Diagrams. As an auxiliary notation for games, we make use of *diagrams* as the one depicted in Fig. 1 (we do not need a formal definition). With every diagram d and structure D we associate a game $G_{d,D} = (V, move^{G_{d,D}})$ as follows:

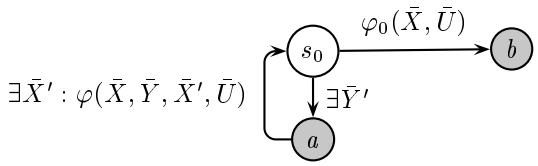
Let $\bar{X} = X_1, \ldots, X_n$ be the unprimed variables of the diagram d, and let $Sq = \{s_1, \ldots, s_m\}$ be the (white and black) "squares" of d. Then the positions of $G_{d,D}$ are

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

The possibles moves between positions are given by the edges in d:
$((s, \bar{x}), (s', \bar{x}')) \in move^{G_{d,D}}$ iff there is an edge $s \xrightarrow{\Theta} s'$ in d, and (i), (ii) hold:

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

[6] A rule r is *nonrecursive* if no literal in the body of r is depending – directly or indirectly via other rules – on the atom in the head of r, see e.g. [AHV95].



$move(s_0, \bar{X}, \bar{Y}, \bar{U},\ a, \bar{X}', \bar{Y}', \bar{U}') \leftarrow \bar{X}' = \bar{X}, \bar{U}' = \bar{U}.$
$move(s_0, \bar{X}, \bar{Y}, \bar{U},\ b, \bar{X}', \bar{Y}', \bar{U}') \leftarrow R_{\varphi_0}(\bar{X}, \bar{U}), \bar{X}' = \bar{X}, \bar{Y}' = \bar{Y}, \bar{U}' = \bar{U}.$
$move(a, \bar{X}, \bar{Y}, \bar{U},\ s_0, \bar{X}', \bar{Y}', \bar{U}') \leftarrow R_{\varphi}(\bar{X}, \bar{Y}, \bar{X}', \bar{U}), \bar{Y}' = \bar{Y}, \bar{U}' = \bar{U}.$
$R_{\varphi_0}(\bar{X}, \bar{U}) \leftarrow \ldots$
$R_{\varphi}(\bar{X}, \bar{Y}, \bar{Z}, \bar{U}) \leftarrow \ldots$
$win(S, \bar{X}, \bar{Y}, \bar{U}) \leftarrow move(S, \bar{X}, \bar{Y}, \bar{U},\ S', \bar{X}', \bar{Y}', \bar{U}'), \neg\, win(S', \bar{X}', \bar{Y}', \bar{U}').$
$answer(\bar{U}) \leftarrow win(s_0, \widetilde{V}, \bar{Y}, \bar{U}).$

Fig. 1. Reduction from Bounded Skolem Normal Form to a Game

$G_{d,D}$ can be viewed as a game played with a *sequence of pebbles* $(S, X_1, \ldots, X_n)$ whose actual value $(s, x_1, \ldots, x_n)$ is a position in $G_{d,D}$: the pebble S is on some square s in d, the pebbles X_i are on elements x_i of D. The players of $G_{d,D}$ move alternately between white and black squares (player I) or vice versa (player II). The pebbles can be moved from $(s, \bar{x})$ to $(s', \bar{x}')$ only if there is an edge $s \xrightarrow{\Theta} s'$ in d, and if additionally the old positions $\bar{x}$ of the X-pebbles and their new positions $\bar{x}'$ satisfy the conditions (i) and (ii) above. In particular, all pebbles X_i have to remain on their positions during a move unless X_i' is $\exists$-quantified in Θ.

Every diagram d with unprimed variables $\bar{X}$ can be directly translated into the definition of the move relation $move(S, \bar{X},\ S', \bar{X}')$ of the game $G_{d,D}$ such that for the resulting W-Datalog^G program P_d we have $\mathrm{WFS}(P_d, D)(win(s, \bar{x})) =$ *true*/*false*/*undef* iff $(s, \bar{x})$ is won/lost/drawn in $G_{d,D}$. This translation is straightforward and should be clear from Fig. 1. Now we are in position to prove

Theorem 3.4 (W-Datalog $\leq$ W-Datalog^G)
For every W-Datalog program P there is an equivalent program $P_G \in$ W-Datalog^G.

Proof. Let P be a W-Datalog program. By Theorems 3.2 and 3.3 there is an equivalent LFP-formula $\psi(\bar{U})$ in bounded Skolem normal form. We view $\psi(\bar{U})$ as the game depicted by the diagram in Fig. 1. As explained before, the program P_ψ in Fig. 1 can be directly obtained from the diagram and represents this game.

Since φ_0 and φ in the diagram are quantifier-free FO-formulas, the rules defining the equivalent *idb* relations $R_{\varphi_0}(\bar{X}, \bar{U})$, $R_\varphi(\bar{X}, \bar{Y}, \bar{Z}, \bar{U})$ of P_ψ can be chosen semipositive and nonrecursive.

The idea behind the game is as follows: Player I wants to prove that some

$\bar{X}$ are in the least fixpoint $W^{\infty}_{\bar{u}}$ of Theorem 3.3. Player II wants to prove the contrary. $\bar{U}$ are fixed parameters and passed around unchanged.

If $\varphi_0(\bar{X}, \bar{U})$ holds, I can move from s_0 to b and win, since there are no moves from b. The other possibility for I to win is the move to a. I can win by moving to a if she chooses some $\bar{Y}$ such that for all $\bar{Z}$ for which $\varphi(\bar{X}, \bar{Y}, \bar{Z}, \bar{U})$ holds, $W(\bar{Z})$ also holds. In terms of the game, this means that $\bar{Z}$ has to be established as a won position for I *in the next round*. This is achieved by substituting $\bar{X}'$ for $\bar{Z}$ in $\varphi(\bar{X}, \bar{Y}, \bar{Z}, \bar{U})$ as in Fig. 1, which "feeds back" the new $\bar{X}$ in place of $\bar{Z}$ in the fixpoint process. By induction one can verify that (for all $\bar{y}$):

$$win(s_0, \bar{x}, \bar{y}, \bar{u}) \in \Gamma^{2k}_{P_\psi} \Leftrightarrow \bar{x} \in W^k_{\bar{u}} \quad ,$$

which is the case iff I wins the game in s_0 in k rounds. Therefore,

$$win(s_0, \bar{x}, \bar{y}, \bar{u}) \in \mathrm{lfp}(\Gamma^2_{P_\psi}) \Leftrightarrow \bar{x} \in W^{\infty}_{\bar{u}} \quad ,$$

which implies

$$\begin{aligned} answer(\bar{u}) \in \mathrm{lfp}(\Gamma^2_{P_\psi}) &\Leftrightarrow \text{ for some } v: \ \widetilde{v} \in W^{\infty}_{\bar{u}} \\ &\Leftrightarrow D \models \psi(\bar{u}) \quad , \end{aligned}$$

where D is the structure which is implicit in the definition of $W^k_{\bar{u}}$ and Γ_{P_ψ}. ∎

Remark. We have chosen as Theorem 3.3 the normal form of [Gro94] since it allows a particularly short translation into a game. Theorem 3.4 can also be proven from the following more familiar normal form theorem:

Theorem 3.5 ([Imm86])
Every LFP-formula is equivalent to a formula of the form $[\mathrm{LFP}_{R(\bar{X})}\varphi]\bar{T}$ *where* φ *is a FO-formula.*

Sketch of a proof of Theorem 3.4 using Theorem 3.5: By induction on FO-formulas define a diagram d_φ such that $G_{d_\varphi, D}$ reflects the evaluation of φ in D. In order to convert d_φ to a diagram d_ψ for $\psi = [\mathrm{LFP}_{R(\bar{X})}\varphi]\bar{T}$, substitute all arrows with label containing R into appropriate loops back to the start of d_φ (use that φ is positive in R). Convert d_ψ into a W-Datalog^G program as described above.

3.2 Reduction from Games to Draw-Free Games

It remains to show that for each game, there is an equivalent draw-free game. We present an informal proof emphasizing the idea of the construction.[7]

Theorem 3.6 (W-Datalog$^G \leq$ W-Datalog$^G_2$)
For every W-Datalog^G program there is an equivalent program in W-Datalog$^G_2$.

[7] The presented reduction is due to [Kub95] which also contains the details of a proof of a normal form for LFP implying Theorem 3.6.

Proof. The main problem consists in 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 $2G$ which simulates these two games on the original structure G. To do so, we need two pebbles – one for each game in G. Call these the *clock pebble* $\bar{Y}$ (on position $\bar{y}$ in G) and the *verify pebble* $\bar{X}$ (on position $\bar{x}$ in G).[8] 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 G.

Initially, player I claims that the verify pebble is on a won position, i.e. $|\bar{x}| < \infty$. 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 $2G$ in such a way, that II can be disproved if she "cheats" by choosing a $\bar{y}$ which is not maximal.

The new game $2G$ is constructed as follows (cf. Fig. 3): 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 G, respectively (Fig. 2). Note that in the simulated game G, I moves first in $1\,\mathsf{round}(\bar{X})$ while II moves first in $1\,\mathsf{round}(\bar{Y})$.

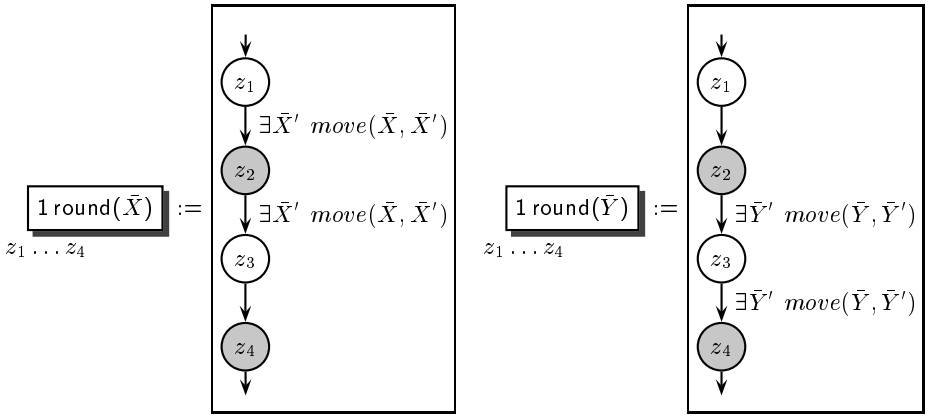
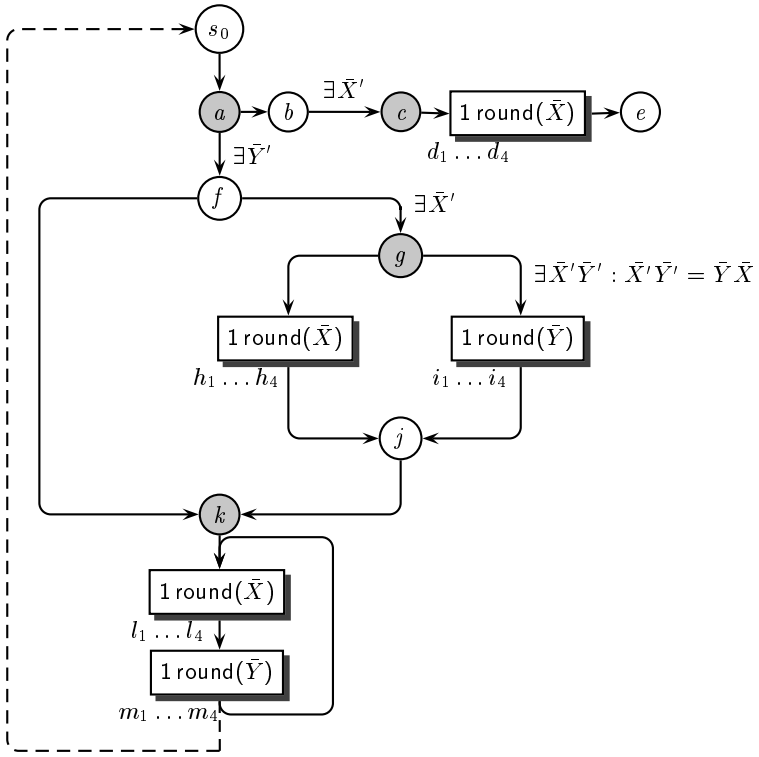


Fig. 2. Macro Definitions

Like above, the diagram in Fig. 3 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

[8] As noted before, $\bar{X} = X_1, \ldots, X_n$ is a sequence of pebbles on positions $\bar{x}$; analogously for $\bar{Y}$.



Player I	
$s_0 \to a$	$\lvert\bar{x}\rvert < \infty$, "the length of $\bar{x}$ is finite, ie $\bar{x}$ 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 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. 3. Draw-Free Game $2G$ and Implicit Claims of I and II.

of the original G (used in the macros of Fig. 2) is n-ary, the new move relation of $2G$ is $2(n+1)$-ary. For a given answer relation $answer(\bar{U}) \leftarrow win(\bar{X})$ in P_G, the new answer relation of P_{2G} is defined as

$$answer(\bar{U}) \leftarrow win(s_0, \bar{X}, \bar{Y}).$$

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

Assume for the moment that the dashed edge $m_4 \to s_0$ in Fig. 3 is absent. The loop $l_1 \to m_4 \to l_1$ compares the lengths of $\bar{x}$ and $\bar{y}$: I wins this comparison if $|\bar{x}| < \infty$ and $|\bar{x}| \leq |\bar{y}|$, while II wins if $|\bar{y}| < \infty$ and $|\bar{y}| < |\bar{x}|$.

To get a better understanding of the construction of $2G$, we explain the diagram in Fig. 3 as a dialog between I and 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 I wins $(s_0, \bar{x}, \bar{y})$ in $2G$ (for arbitrary $\bar{y}$) if I wins $\bar{x}$ in G, and II wins $(s_0, \bar{x}, \bar{y})$ in $2G$ if $\bar{x}$ is lost or drawn (for I) in G. Thus the new game $2G$ is *determinate* for positions $(s_0, \bar{x}, \bar{y})$.

However $2G$ may still contain positions which are drawn: Consider e.g. $(l_1, \bar{x}, \bar{y})$ where $\bar{x}$ and $\bar{y}$ are drawn in G. Then II gets no chance of refuting the claim that $\bar{x}$ is won in G, hence $(l_1, \bar{x}, \bar{y})$ is also drawn in $2G$. In order to allow II to defeat such false claims, the dashed edge is needed. By moving along $m_4 \to s_0$, II can win and refute I by choosing the maximal $\bar{y}$ in the move $a \to f$.

The final obstacle is that one has to verify that if $\bar{x}$ is won in G, then II cannot delay the game infinitely using the edge $m_4 \to s_0$. Indeed $|\bar{x}|$ decreases each time the game reaches m_4:

(a) if II chooses in a some $\bar{y}$ with $|\bar{y}| \geq |\bar{x}|$, then I has to move along $f \to k$ thereby enforcing that at least $1\,\mathsf{round}(\bar{X})$ is played.
(b) if II chooses $|\bar{y}| < |\bar{x}|$, then I chooses a new $\bar{x}$ with $|\bar{x}| = |\bar{y}|+1$. Independent of the choice of II ($g \to h_1$ or $g \to i_1$), the new $\bar{x}$ will be at least one smaller, when m_4 is reached.

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

- I wins $(s_0, \bar{x}, \bar{y})$ in $2G$ iff $\bar{x}$ is won in G, and
- II wins $(a, \bar{x}, \bar{y})$ in $2G$ iff $\bar{x}$ is lost or drawn in G.
- No positions $(s, \bar{x}, \bar{y})$ in $2G$ are drawn.

■

Putting everything together, we have

$$\mathsf{W\text{-}Datalog} \overset{\text{Theorem 3.4}}{\leq} \mathsf{W\text{-}Datalog}^G \overset{\text{Theorem 3.6}}{\leq} \mathsf{W\text{-}Datalog}_2^G \leq \mathsf{W\text{-}Datalog}$$

which proves

Corollary 3.7 ($\mathsf{W\text{-}Datalog} \equiv \mathsf{W\text{-}Datalog}_2$)
For every well-founded Datalog program, there is an equivalent total program.

Acknowledgements. The third author would like to thank JÜRGEN DIX, WOLFGANG MAY and CHRISTIAN SCHLEPPHORST for many fruitful discussions, and his co-authors for illuminating insights into the realm of finite model theory and games.

References

[AB94] K. R. Apt and R. N. Bol. Logic Programming and Negation: A Survey. *Journal of Logic Programming*, 19/20:9–71, 1994.

[ABW88] K. R. Apt, H. Blair, and A. Walker. Towards a Theory of Declarative Knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89 – 148. Morgan Kaufmann, 1988.

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

[Che95] W. Chen. Query Evaluation in Deductive Databases with Alternating Fixpoint Semantics. *ACM Transactions on Database Systems*, 20(3):239–287, 1995.

[Dix95] J. Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. In A. Fuhrmann and H. Rott, editors, *Logic, Action and Information.* de Gruyter, 1995.

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

[Gro94] M. Grohe. *The Structure of Fixed-Point Logics.* PhD thesis, Universität Freiburg, 1994. http://logimac.mathematik.uni-freiburg.de/preprints/groh12-94-{1,2,3}.ps.

[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. http://logimac.mathematik.uni-freiburg.de/preprints/kub95.ps.

[Sch95] J. S. Schlipf. Complexity and Undecidability Results in Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 15(III-IV), 1995.

[VG89] A. Van Gelder. The Alternating Fixpoint of Logic Programs with Negation. In *Proc. ACM Symposium on Principles of Database Systems*, pages 1–10, 1989.

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

[VGRS88] A. Van Gelder, K. Ross, and J. Schlipf. Unfounded Sets and Well-Founded Sematics for General Logic Programs. In *Proc. ACM Symposium on Principles of Database Systems*, pages 221–230, 1988.