Inductive Definability with Counting on Finite Structures

Erich Grädel[*] Martin Otto[*]

1 Introduction

We study the properties and the expressive power of logical languages that include both a mechanism for inductive definitions and the ability to count. The most important of these languages is fixpoint logic with counting terms, denoted (FP + C).

To motivate the study of these logics, we recall that the expressive power of first-order logic (FO) is limited by two main reasons: It lacks the power to express anything that requires recursion (the most notable example is the transitive closure query) and it cannot count (the best-known example here is that no first-order formula is true precisely on the structures with even cardinality). There are several well-studied logics and database query languages that add recursion in one way or another to FO (or part of it), notably the various forms of fixpoint logics, the query language Datalog and its extensions.

On ordered finite structures, some of these languages express precisely the queries that are computable in Ptime or Pspace. However, on arbitrary finite structures they do not, and almost all known examples showing this involve counting. While in the presence of an ordering, the ability to count is inherent e.g. in fixpoint logic, hardly any of it is retained in its absence. Thus, Immerman [15] proposed to add counting quantifiers to fixpoint logic and asked whether this would suffice to capture Ptime. Cai, Fürer and Immerman [5] answered this question negatively; in fact (FP + C) does not even express all Logspace-computable queries. Nevertheless we argue that fixpoint logic with counting is an important language that deserves more attention.

We will show that in the presence of counting terms (or counting quantifiers) inductive definability on arbitrary finite structures has nice properties that it retains without counting only in the case of ordered structures. The organization of the paper is summed up in the following:

Equivalent characterizations of inductive definability with counting. In section 2, we define *(inflationary) fixpoint logic with counting* (FP + C) and *partial fixpoint logic with counting* (PFP + C) by closing a two-sorted version of first-order logic under counting terms and the usual rules for building inflationary or partial fixpoints. We also investigate several other logical and algorithmical versions of inductive definability with counting and show that they all have the same expressive power. This supports our belief that (FP + C) is a natural and robust class. In particular we consider:

[*]Lehrgebiet Mathematische Grundlagen der Informatik, RWTH Aachen, Ahornstr. 55, D-5100 Aachen, email: {graedel,otto}@mjoli.informatik.rwth-aachen.de

Datalog with counting. Counting terms can be added also to Datalog. We show that (Datalog + C) = (FP + C). In particular, (Datalog + C) is closed under negation and therefore, in the presence of counting, all the many extensions of Datalog, notably Stratified Datalog are equivalent to Datalog. This is not at all the case without counting.

A functional logic. Gurevich has proved that the usual schemes, that define — over $\mathbb{N}$ — the recursive functions, characterize, when interpreted over ordered finite structures, precisely the functions computable in polynomial time. We define a class of global functions, defined by a similar scheme, which has precisely the power of (FP + C).

A machine theoretic characterization. An algorithmic definition of (FP + C) is provided by a "relational machine" inspired by [3]. This machine operates like a Turing machine but interacts with a relational store in a "generic" way so that the machine treats algebraically indistinguishable structures and tuples in the same way. The queries computed in polynomial-time by these machines are precisely the queries in (FP + C).

Infinitary logic with counting. Let $C^k_{\infty\omega}$ be infinitary logic with k variables, $L^k_{\infty\omega}$, extended by the quantifiers $\exists^{\geq m}$ ("there exist at least m") for all $m \in \mathbb{N}$. It is easy to see that (FP + C) $\subseteq \bigcup_k C^k_{\infty\omega}$. We investigate the relationship of fixpoint logic and infinitary logic in the presence of counting. In particular, we show that the $C^k_{\infty\omega}$-types can be uniformly ordered by a formula in (FP + C). These results parallel those in [3, 8] on FP and $L^\omega_{\infty\omega}$. As a consequence, a functor can be introduced which associates with every structure an arithmetical invariant (i.e. essentially a collection of numerical predicates), which characterizes the original structure up to $C^k_{\infty\omega}$-equivalence. Furthermore the (FC + C) definable properties of the original structures exactly correspond to the FP definable (or PTIME) properties of the associated invariants.

We also obtain a characterization of (FP + C) as a sublogic of $\bigcup_k C^k_{\infty\omega}$ in terms of families $(\varphi_n)_{n\in\omega}$ of finitary formulae in $C^k_{\infty\omega}$ (in which φ_n applies to structures of size n). The uniformity condition which exactly restricts the expressive power of these to (FP + C) is just PTIME constructibility of the sequence of the φ_n. This is quite unlike the situation for FP and the $L^k_{\infty\omega}$.

Inflationary vs. partial fixpoints. Abiteboul and Vianu [3] recently proved that partial fixpoint logic collapses to fixpoint logic if and only if PTIME = PSPACE. The analogous result in the presence of counting is also true: PTIME = PSPACE $\Longleftrightarrow$ (FP + C) = (PFP + C). On the other side, Abiteboul and Vianu also proved that already a collapse of PFP$|_P$ to fixpoint logic is equivalent to PTIME = PSPACE. (The logic PFP$|_P$ is partial fixpoint logic restricted to PFP operators that close after at most polynomially many iterations on every structure.) In the presence of counting we can prove instead that (FP + C) = (PFP$|_P$ + C). In fact it turns out that the PSPACE restriction of the above machine model corresponds to (PFP + C), or While with counting.

The algorithmic characterization of (FP + C), which is fully expanded in [19], inspired some of the results concerning the relation with infinitary logic and that with partial fixpoints. We here present these results in a self-contained way, which does not rely on the machine model in an essential way. While the present approach might be methodically more concise, we also would like to refer the interested reader to the treatment fully exploiting the algorithmic characterization in [19]. For some arguments it remains the intuitively more appealing.

2 Characterizations of inductive definability with counting

2.1 Fixpoint logics with counting

Logics with counting are two-sorted. With any one-sorted finite structure $\mathfrak{A}$ with universe A, we associate the two-sorted structure $\mathfrak{A}^* := \mathfrak{A} \,\dot\cup\, \langle n; \leq, 0, e\rangle$ with $n = |A| + 1$, where $\leq$ is the canonical ordering on $n = \{0, \dots, n-1\}$ and 0 and e stand for the first and the last element of n. Thus, we have taken the disjoint union of $\mathfrak{A}$ with a canonical ordered structure of size n. We take $n = |A| + 1$ rather than $n = |A|$ to be able to represent the cardinalities of all subsets of $|A|$ within n.

We start with first-order logic over two-sorted vocabularies $\sigma \cup \{\leq, 0, e\}$, with semantics over structures $\mathfrak{A}^*$ defined in the obvious way. We will use latin letters $x, y, z, \dots$ for the variables over the first sort, and greek letters $\lambda, \mu, \nu, \dots$ for variables over the second sort.

The two sorts are related by *counting terms*, defined by the following rule: Let $\varphi(x)$ be a formula with variable x (over the first sort) among its free variables. Then $\#_x[\varphi]$ is a term in the second sort, with the set of free variables $\mathrm{free}(\#_x[\varphi]) = \mathrm{free}(\varphi) - \{x\}$. The value of $\#_x[\varphi]$ is the number of elements a that satisfy $\varphi(a)$.

First-order logic with counting, denoted (FO + C), is the closure of two-sorted first-order logic under counting terms.

Example. To illustrates the use of counting terms we present a formula $\psi(E_1, E_2) \in$ (FO + C) which expresses that two equivalence relations E_1 and E_2 are isomorphic; of course a necessary and sufficient condition for this is that for every i, they have the same number of equivalence classes of size i:

$$\psi(E_1, E_2) \equiv (\forall \mu)(\#_x[\#_y[E_1xy] = \mu] = \#_x[\#_y[E_2xy] = \mu]).$$

We obtain *(inflationary) fixpoint logic with counting* (FP + C) and *partial fixpoint logic with counting* (PFP + C) by closing first-order logic under counting terms and the usual rule for building inflationary or partial fixpoints:

Let $R \notin \sigma$ be a relational variable of mixed arity (k, ℓ), i.e. with arity k over the first and arity ℓ over the second sort. A formula $\psi(R, \bar{x}, \bar{\mu})$ of vocabulary $\sigma \cup \{R\}$, defines for every σ-structure $\mathfrak{A}$ an operator $\psi^{\mathfrak{A}}$ on the class of (k, ℓ)-ary predicates over $\mathfrak{A}^*$, namely

$$\psi^{\mathfrak{A}} : R \longmapsto R \cup \{(\bar{u}, \bar{\nu}) \mid \mathfrak{A}^* \models \psi(R, \bar{u}, \bar{\nu})\}.$$

Since this operator is *inflationary* — i.e. $\psi^{\mathfrak{A}}(R) \supseteq R$ — it has a fixed point that is constructed inductively, starting with the empty relation: Set $\Psi^0 = \varnothing$ and $\Psi^{i+1} = \psi^{\mathfrak{A}}(\Psi^i)$. At some stage i, this process reaches a stable predicate $\Psi^i = \Psi^{i+1}$, which is called the *inflationary fixed point* of ψ on $\mathfrak{A}$, and denoted by Ψ^∞. The *stage* of a tuple $(\bar{u}, \bar{\nu})$ in Ψ^∞ is the minimal i such that $(\bar{u}, \bar{\nu}) \in \Psi^i$ (if $(\bar{u}, \bar{\nu}) \notin \Psi^\infty$ then $\mathrm{stage}_R(\bar{u}, \bar{\nu}) = \infty$).

Definition 2.1 *Fixpoint logic with counting*, denoted (FP + C), is the closure of (two-sorted) first-order logic under

(i) the rule for building counting terms;

(ii) the usual rules of first-order logic for building terms and formulae;

(iii) the fixpoint formation rule: Suppose that $\psi(R, \bar{x}, \bar{\mu})$ is a formula of vocabulary $\sigma \cup \{R\}$ where $\bar{x} = x_1, \dots, x_k$, $\bar{\mu} = \mu_1, \dots, \mu_\ell$, and R has mixed arity (k, ℓ), that $\bar{u}$ is a k-tuple of terms over the first sort and $\bar{\nu}$ an ℓ-tuple of terms over the second sort. Then
$$[\mathrm{IFP}_{R,\bar{x},\bar{\mu}}\ \psi(R, \bar{x}, \bar{\mu})](\bar{u}, \bar{\nu})$$
is a formula of vocabulary σ.

The interpretation of the fixpoint formula in *(iii)* in a structure $\mathfrak{A}$ is that $(\bar{u}, \bar{\nu})$ is included in the fixpoint Ψ^∞.

An interesting example for a (FP + C)-query is the method of *stable colourings* for graph-canonization. Given a graph G with a colouring $f : V \to 0, \dots, r$ of its vertices, we define a refinement f' of f, where vertex x has the new colour $f'x = (fx, n_1, \dots, n_r)$ where $n_i = \#y[Exy \wedge (fy = i)]$. The new colours can be sorted lexicographically so that they form again an initial subset of $\mathbb{N}$. Then the process can be iterated until a fixpoint, the *stable colouring* of G is reached. It is known that almost all graphs have the property that no two vertices have the same stable colour. Thus stable colourings provide a polynomial-time graph-canonization algorithm for a dense class of graphs. It should be clear that the stable colouring of a graph is definable in (FP + C) (see [16] for more details).

Remarks. A slightly different definition of fixpoint logic with counting has been proposed by Immerman [15], who related the two sorts by *counting quantifiers* rather than counting terms. Counting quantifiers have the form $(\exists i\ x)$ for "there exist at least i x", where i is a second-sort variable. It is obvious, that the two definitions are equivalent. In fact, (FP + C) is a very robust logic. For instance, it would not change its expressive power if we would allow counting over tuples, even of mixed type, i.e. terms of the form $\#_{\bar{x},\bar{\mu}}\varphi$. We will illustrate this for the case of (Datalog + C), a language that is equivalent to (FP + C) (see section 2.2). However, this is not true for (FO + C) which is very sensitive to the precise definition that is chosen.

Different variants of counting logics have been investigated by Grumbach and Tollu [10]; their languages are weaker since they do not allow nested counting (i.e. counting over formulae, that do themselves contain counting terms). For further results on fixed point logic with counting we refer to [9].

Partial fixpoints. Besides the inflationary operator defined above, a formula $\psi(R, \bar{x}, \bar{\mu})$ gives us also the operator
$$\tilde{\psi}^{\mathfrak{A}} : R \longmapsto \{(\bar{u}, \bar{\nu}) \mid \mathfrak{A}^* \models \psi(R, \bar{u}, \bar{\nu})\}.$$
Starting with $\tilde{\Psi}^0 = \varnothing$, we can also define the iterative stages $\tilde{\Psi}^m$ of this operator, but since it is in general neither inflationary nor monotone, this process is not guaranteed to reach a fixed point. (As an example consider the formula $\psi(R, x) \equiv \neg Rx$, whose stages oscillate between the empty and the full relation.) Define the *partial fixpoint* of ψ on $\mathfrak{A}$ to be $\tilde{\Psi}^m$ if there is an m with $\tilde{\Psi}^{m+1} = \tilde{\Psi}^m$, and to be empty otherwise.

Partial fixpoint logic with counting, denoted (PFP + C), is defined in the same way as (FP + C), but with the inflationary fixpoint operation replaced by a partial fixpoint operation.

Fixpoint logics without counting. There is an extensive literature on various forms of fixpoint logics without counting terms (see e.g. [2, 4, 6, 7, 13, 14, 17, 18, 21]). We just recall a few important facts about fixpoint logic:

First, on every class of finite structures, the expressive power of fixpoint logic is between first-order definability and PTIME-computability. There are two extreme cases: On sets, fixpoint logic collapses to first-order logic, and on ordered structures, fixpoint logic coincides with PTIME [14, 21]. On most other classes, and in particular on the class of all finite structures, fixpoint logic is more powerful than first-order logic, but does not capture all PTIME-queries.

Abiteboul and Vianu have shown, that in the absence of counting, partial fixpoint logic has the same expressive power as the query language *while*, which is first-order logic extended by while-loops as an iteration construct. Together with a result of Vardi [21], this proves that on ordered structures, partial fixpoint logic captures PSPACE.

2.2 Datalog with counting

Let us first recall the basic definitions of Datalog:

Definition 2.2 A *Datalog program* Π consists of a finite set of rules of the form

$$H \leftarrow B_1 \wedge \cdots \wedge B_m$$

where H is an atomic formula $S(x_1, \ldots, x_r)$, called the *head* of the rule, and $B_1 \wedge \cdots \wedge B_m$ is a conjunction of atomic formulae $R(x_1, \ldots, x_m)$ and equalities $x_i = x_j$. Every x_i is either a variable or a constant. A predicate that occurs in the head of some rule is called an *intensional database predicate*, abbreviated IDB predicate; a predicate occuring only in the bodies of the rules is called an *extensional database predicate*, or EDB predicate. One of the IDB predicates is the *goal predicate* of the program. The extensional vocabulary of Π is formed by all EDB predicates and by all constants occurring in Π; a finite structure of this vocabulary is called an *extensional database* EDB for Π. Given any extensional database $\mathfrak{B}$, the program computes intensional relations, by the usual fixpoint semantics (or, equivalently, minimum model semantics). The result of Π on $\mathfrak{B}$ is the value of the goal predicate after the computation has terminated. If Π is a Datalog program with goal predicate Q we denote by (Π, Q) the query computed by this program.

On unordered databases, Datalog, in fact even Stratified Datalog, is a proper subset of fixpoint logic [7, 17]. On ordered structures, Datalog is also weaker than fixpoint logic. However, Datalog($\neg$), i.e. Datalog with negations over the EDB predicates is equivalent to fixpoint logic over ordered structures and therefore captures polynomial time [4]. Here we show that in the presence of counting, the situation is different: Datalog with counting terms has the same expressive power as fixpoint logic with counting.

We can increase the power of Datalog by a counting mechanism in a very similar way as we did with fixpoint logic:

Definition 2.3 *Datalog with counting*, denoted (Datalog + C), extends Datalog by allowing two-sorted IDB predicates and counting terms. The two-sorted IDB predicates have the form $R(\bar{x}, \bar{\mu})$, where $\bar{x}$ range over the first sort, i.e. over elements of the extensional database $\mathfrak{A}$, and $\bar{\mu}$ range over the second sort, i.e. over n. For any atom $R(x, \bar{y}, \bar{\mu})$ we have a counting term $\#_x[R(x, \bar{y}, \bar{\mu})]$. A term over the second sort is called an *arithmetical term*. The arithmetical terms are either 0, e, counting terms or $t + 1$, where t is also an arithmetical term. Thus, a program in (Datalog + C) is a finite set of clauses of the form

$$H \leftarrow B_1 \wedge \cdots \wedge B_m$$

where the head H is an atomic formula $R(\bar{x}, \bar{\mu})$, and $B_1, \ldots, B_m$ are atomic formulae $R(\bar{x}, \bar{\mu})$ or equalities of terms (over the first or the second sort).

On every extensional database, the program computes intensional relations by inflationary fixpoint semantics. Note that for classical Datalog programs, it makes no difference whether the fixpoint semantics is defined to be inflationary or not, since the underlying operator is monotone anyway. However, for programs in (Datalog + C), the semantic has to be inflationary, since otherwise, the equalities of arithmetical terms give rise to non-monotone operators. For the same reason, minimum model semantics will no longer be defined. Since inflationary fixpoint semantics is one of the various equivalent ways to define the semantics of Datalog, we feel that both syntax and semantics of (Datalog + C) generalize Datalog in a natural way.

We could also introduce counting in a (at first sight) more general form, namely by allowing counting terms of the form $\#_{\bar{x},\bar{\mu}}[R(\bar{x}, \bar{\mu}, \bar{y}, \bar{\nu})]$. While this may be convenient to write a program in shorter and more understandable form, it does not affect the power of (Datalog + C):

Lemma 2.4 *Counting over tuples, even of mixed type, does not increase the expressive power of* (Datalog + C).

PROOF. To illustrate the argument we show that counting over pairs can be reduced to counting over single elements. Note that

$$\#_{x,y}[Rxy] = \sum_x \#_y[Rxy] = \sum_\mu \mu \cdot \#_x[\#_y[Rxy] = \mu].$$

Using the two clauses

$$\begin{aligned} Ax\mu &\leftarrow (\#_y[Rxy] = \mu) \\ B\mu\nu &\leftarrow (\#_x[Ax\mu] = \nu) \end{aligned}$$

we can build up a purely arithmetical predicate B, which is the graph of the function $f(\mu) = \#_x[\#_y[Rxy] = \mu]$. We then have to build a program for the expression $\sum_\mu \mu f(\mu)$, which is no problem since, over the second, ordered sort, arithmetical expressions of this kind can be handled in Datalog. Counting over elements and tuples of the second sort can also be easily simulated using similar arithmetical expressions. ■

Hence cardinalities of arbitrary predicates can be equated in a Datalog program: we take the liberty to write equalities like $|Q| = |R|$ in the body of a rule, for simplicity.

Lemma 2.5 *Let* Π *be a* (Datalog + C)*-program with IDB predicates* $Q_1, \ldots, Q_r$. *There exists another* (Datalog + C)*-program* Π', *whose IDB predicates include* $Q_1, \ldots, Q_r$ *and a Boolean control predicate* C^* *such that*

- $(\Pi', Q_i) = (\Pi, Q_i)$ *for all* i;
- (Π', C^*) *is true on all databases and* C^* *becomes true only at the last stage of the evaluation of* Π'.

PROOF. Besides C^*, we add a unary IDB predicate C^0 and, for every IDB predicate Q_i of Π a new IDB predicate Q_i' of the same arity. Then, Π' is obtained by adding the following clauses to Π:

$$\begin{array}{rcl} & & C^0(x) \\ Q'_i(\bar{x},\bar{\mu}) & \leftarrow & Q_i(\bar{x},\bar{\mu}) \quad \text{for } 1 \le i \le r \\ C^* & \leftarrow & C^0(x) \wedge (|Q_1| = |Q'_1|) \wedge \cdots \wedge (|Q_r| = |Q'_r|) \end{array}$$

Observe that Q'_i just lags one step behind Q_i up to the point of saturation. The atom $C^0(x)$ is necessary to avoid that C^* is set to true right at the first stage. ■

Lemma 2.5 essentially says, that we can attach to any program a Boolean control predicate which becomes true when the evaluation of the program is terminated. We can then compose two Datalog programs while making sure that the evaluation of the second program starts only after the first has been terminated. As a first application, we show that (Datalog + C) is closed under negation.

Lemma 2.6 *The complement of a* (Datalog + C)*-query is also a* (Datalog + C)*-query.*

Proof. Let (Π, Q) be a (Datalog + C)-query, and let Π' be the program as specified in Lemma 2.5. Take a new variable z and new IDB predicates $\tilde{Q}$ and R with arity$(R) =$ arity(Q) and arity$(\tilde{Q}) =$ arity$(Q) + 1$. Construct Π'' by adding to Π' the rules:

$$\begin{array}{rcl} \tilde{Q}(\bar{x},\bar{\mu},z) & \leftarrow & Q(\bar{x},\bar{\mu}) \\ R(\bar{x},\bar{\mu}) & \leftarrow & C^* \wedge (\#_z[\tilde{Q}(\bar{x},\bar{\mu},z)] = 0) \end{array}$$

The query (Π'', R) is the complement of (Π, Q). ■

Difficulties to express negation are the reason why, in the absence of counting (or of an ordering), Datalog is weaker than fixpoint logic. Also the limited form of negation that is available in Stratified Datalog (which does not allow for "recursion through negation") does not suffice to express all fixpoint queries [7, 17]. We now prove that (Datalog + C) does not have these limitations and is equally expressive as (FP + C):

Theorem 2.7 (Datalog + C) = (FP + C).

Proof. It is obvious that (Datalog + C) $\subseteq$ (FP + C). Conversely, we show that for every formula $\psi \in$ (FP + C), there exists a (Datalog + C)-program Π_ψ with goal predicate Q_ψ such that (Π_ψ, Q_ψ) is equivalent to ψ.

If ψ is atomic, this is trivial. Also, both Datalog and (Datalog + C) are trivially closed under disjunction and existential quantification. Closure under negation has already been proved.

We next consider counting terms: Suppose that we have a program Π_φ describing the formula $\varphi(x,\bar{y},\bar{\mu})$ in the way just specified. By Lemma 2.5, we can assume that Π_φ contains a control predicate C^*_φ becoming true at the last stage of the evaluation of Π_φ. We need a program Π_ψ for the formula $\psi(\bar{y},\bar{\mu},\nu) \equiv \#_x[\varphi(\bar{x},\bar{y},\bar{\mu})] = \nu$. This can be obtained by adding to Π_φ the clause

$$Q_\psi(\bar{y},\bar{\mu},\nu) \leftarrow C^*_\varphi \wedge \#_x[Q_\varphi(\bar{x},\bar{y},\bar{\mu})] = \nu.$$

Finally, we explain how to simulate fixpoint formulae $\psi \equiv [\mathrm{IFP}_{R,\bar{x},\bar{\mu}}\, \varphi(R,\bar{x},\bar{\mu})](\bar{y},\bar{\nu})$. By induction hypothesis, there exists a program Π_φ with goal predicate Q_φ and control predicate C^*_φ which defines φ. Of course R is an EDB predicate of Π_φ.

Construct Π_ψ as follows: For every (EDB or IDB) predicate Q in Π_φ (including the control predicate), take a predicate Q' whose arity over the second sort is increased by a sufficiently large number ℓ; intuitively $Q'(\bar{x}, \bar{\mu}, \bar{\lambda})$ shall mean that $Q(\bar{x}, \bar{\mu})$ is true at stage $\bar{\lambda}$ of the evaluation of the fixpoint operator.

Start with the clauses

$$\begin{array}{lcll} C^*_\varphi(\bar{0}) & & & \\ Q'(\bar{x}, \bar{\mu}, 0) & \leftarrow & Q(\bar{x}, \bar{\mu}) & \quad \text{for all predicates } Q \end{array}$$

Add to this the program obtained from Π_φ as follows

1. Replace all EDB atoms $Q(\bar{x}, \bar{\mu})$ by $Q'(\bar{x}, \bar{\mu}, \bar{\alpha})$ (this includes the case that $Q = R$).

2. Replace all IDB atoms $Q_\varphi(\bar{x}, \bar{\mu})$ by $R'(\bar{x}, \bar{\mu}, \bar{\alpha}+1)$ and all other IDB atoms $P(\bar{x}, \bar{\mu})$ by $P'(\bar{x}, \bar{\mu}, \bar{\alpha}+1)$.

3. Add to the body of every clause the predicate $C^*_\varphi(\bar{\alpha})$.

4. Add the clauses

$$\begin{array}{rcl} R'(\bar{x}, \bar{\mu}, \bar{\alpha}+1) & \leftarrow & R'(\bar{x}, \bar{\mu}, \bar{\alpha}) \\ Q_\psi(\bar{x}, \bar{\mu}) & \leftarrow & R'(\bar{x}, \bar{\mu}, \bar{\alpha}) \end{array}$$

This programs simulates the evalution of the fixpoint operator stage per stage. ∎

To illustrate the expressive power of (Datalog + C) we exhibit a simple program for the GAME query. The GAME query is complete for fixpoint logic with respect to quantifier-free interpretations and is the canonical example that separates fixpoint logic from Stratified Datalog [7, 17]. Given a structure $\mathfrak{A} = (A, S, M)$ with universe A to be interpreted as a set of positions in a two-player game, with a monadic predicate S defining the positions where Player I has won, and binary predicate M describing the possible moves. The GAME query asks for the set Z of positions from which Player I has a winning strategy when he is to make the next move. GAME is definable in fixpoint logic, since

$$Z = [\mathrm{IFP}_{W,x}\ Sx \vee \exists y \forall z (Mxy \wedge (Myz \rightarrow Wz))].$$

Here is a (Datalog + C)–program with goal predicate Z defining GAME:

$$\begin{array}{rcl} Wx0 & \leftarrow & Sx \\ Uyz\mu & \leftarrow & Myz \wedge Wz\mu \\ Vy\mu & \leftarrow & \#_z[Myz] = \#_z[Uyz\mu] \\ Wx\lambda & \leftarrow & Mxy \wedge Vy\mu \wedge \lambda = \mu + 1 \\ Zx & \leftarrow & Wx\mu \end{array}$$

The evaluation of this program on a game structure $\mathfrak{A}$ assigns to W (respectively V) the set of pairs $(x, \mu) \in A \times |A|$, such that Player I has a winning strategy from position x in μ moves when he (respectively Player II) begins the game.

2.3 A functional logic

Gurevich showed that the usual schemes for defining recursive functions on the natural numbers define, when interpreted over finite ordered structures, precisely the polynomial-time computable global functions [11, 12]. In particular, both fixpoint logic and recursive functions coincide over ordered finite structures with polynomial-time computability.

In this section we define a similar functional scheme that works over the two-sorted structures $\mathfrak{A}^*$. We obtain a class of global functions which corresponds to (FP + C).

Definition 2.8 A global function f of arity (j,k), coarity (m,ℓ) and vocabulary σ, is a functor that defines for every σ-structure $\mathfrak{A}$ a function

$$f^{\mathfrak{A}} : A^j \times n^k \longrightarrow A^m \times n^\ell$$

where, as above, $n = |A| + 1$.

It should be kept in mind that there is a natural bijection of numbers up to $n^m - 1$ with m-tuples of numbers up to $n-1$. So actually we consider the values in the second sort of these functions to be in an initial subset of $\mathbb{N}$, and we will take the liberty to perform the usual arithmetical operations on these values.

Definition 2.9 The *basic σ-functions* are

(i) the functions in σ and the characteristic functions χ_R of the predicates $R \in \sigma$ (the coarity of χ_R is of course $(0,1)$);

(ii) the characteristic functions of equality (over either sort)

$$\begin{aligned} \mathbf{eq}(x,y) &= (\mathbf{if}\ x = y\ \mathbf{then}\ 1\ \mathbf{else}\ 0) \\ \mathbf{eq}(\mu,\nu) &= (\mathbf{if}\ \mu = \nu\ \mathbf{then}\ 1\ \mathbf{else}\ 0) \end{aligned}$$

(iii) the constants 0, e (considered as functions of arity $(0,0)$ and coarity $(0,1)$);

(iv) the successor function S, with the convention that $Se = e$;

(v) the function $1 \dot{-} \mu = (\mathbf{if}\ \mu = 0\ \mathbf{then}\ 1\ \mathbf{else}\ 0)$ (of arity and coarity $(0,1)$);

(vi) the projections.

Composition is defined in the obvious way: if $H_1, \dots H_m$ are global functions of the same arity, with coarities summing up to the arity of the global function F, then we can build the global function $F(\bar{x},\bar{\mu}) = G(H_1(\bar{x},\bar{\mu}), \dots, H_m(\bar{x},\bar{\mu}))$.

The summation operator allows to define from a given global function $G(x,\bar{y},\bar{\mu})$ of arity $(1+j,k)$ and coarity $(0,\ell)$ a new global function

$$F(\bar{y},\bar{\mu}) = \sum_x G(x,\bar{y},\bar{\mu})$$

of arity (j,k) and coarity $(0,\ell+1)$, with the obvious meaning. (Note that coarity $(0,\ell+1)$ suffices: the values of $G(x,\bar{y},\bar{\mu})$ are smaller than n^ℓ, so the value of $F(\bar{x},\bar{\mu})$ does not exceed $(n-1)(n^\ell - 1) < n^{\ell+1}$.)

Recursion. Let G be a global function of arity (j,k), coarity (m,ℓ) and vocabulary σ; let H be a global function of the same arity and coarity, but over the vocabulary $\sigma \cup \{h\}$ where h is a function symbol that has the same arity and coarity as G. Then a new global function of vocabulary σ, arity $(j,k+1)$ and coarity (m,ℓ) is defined by the scheme

$$\begin{aligned} F(\bar{x},\bar{\mu},0) &= G(\bar{x},\bar{\mu}) \\ F(\bar{x},\bar{\mu},\nu+1) &= [h(\bar{y},\bar{\lambda}) \equiv F(\bar{y},\bar{\lambda},\nu)]H(\bar{x},\bar{\mu}). \end{aligned}$$

The notation $[h(\bar{y},\bar{\lambda}) \equiv F(\bar{y},\bar{\lambda},\nu)]$ is to be understood as an explicit definition of h: to compute $F(\bar{x},\bar{\mu},\nu+1)$, one has to evaluate H, replacing every occurrence of a term $h(\bar{y},\bar{\lambda})$ by $F(\bar{y},\bar{\lambda},\nu)$.

Definition 2.10 Let $\mathcal{S}$ be the closure of the basic functions under composition, the summation operator and the recursion scheme given above.

Theorem 2.11 *$\mathcal{S}$ has the same expressive power as* (FP + C).

PROOF. It is easy to see that the graph of every function in $\mathcal{S}$ can be expressed by a formula in (FP + C). For the converse, we show that for every formula φ of arity (k,ℓ) in (FP + C), there exists a function $F_\varphi : A^k \times n^\ell \to \{0,1\}$ in $\mathcal{S}$, equivalent to the characteristic function of φ, and that every arithmetical term ν of arity (k,ℓ) is equivalent to a function $G_\nu : A^k \times n^\ell \to n$ in $\mathcal{S}$. These simulations are presented in the following table:

$x = y$	$\mathbf{eq}(x,y)$	$(\exists x)\psi$	$1 \dot{-} (1 \dot{-} \sum_x F_\psi)$
$R\bar{x}$	$\chi_R(\bar{x})$	$\#_x[\psi]$	$\sum_x F_\psi$
$\psi \wedge \varphi$	$F_\psi \cdot F_\varphi$	$\mu \leq \nu$	$1 \dot{-} (G_\mu \dot{-} G_\nu)$
$\neg\psi$	$1 \dot{-} F_\psi$		

Finally, a fixpoint formula $\psi \equiv [\text{IFP}_{R,\bar{y},\bar{\nu}}\ \varphi](\bar{x},\bar{\mu})$ is described by $F_\psi = F(\bar{x},\bar{\mu},\bar{e})$ where F is defined from F_φ by the scheme

$$\begin{aligned} F(\bar{x},\bar{\mu},0) &= [\chi_R \equiv 0] F_\varphi(\bar{x},\bar{\mu}) \\ F(\bar{x},\bar{\mu},\lambda+1) &= [\chi_R(\bar{y},\bar{\nu}) \equiv F(\bar{y},\bar{\nu},\lambda)] F_\varphi(\bar{x},\bar{\mu}). \end{aligned}$$

■

2.4 An algorithmic characterization

We introduce a machine model which leads to yet another characterization of (FP + C) and (PFP + C), respectively. The model itself is inspired by the corresponding approach to FP developed in [3, 1]. It is a "generic" or, as we prefer to say, isomorphism-preserving model of computation in the sense that the performance of the machine only depends on the isomorphism type of the input structure (and not on a particular choice of representation which adds extra structure, and hence reduces symmetry).

The intended algorithms can be performed by the following type of machines which consist of two components:

- A relational store: a sequence of relational registers, $(R_i)_{i\in\omega}$, all of some fixed arity r. Every register R_i can hold a collection of r-tuples of elements from the universe of the input structure. The register contents should be thought of as current interpretations of auxiliary r-ary relations over the input domain.

- A Turing component, with a work tape with read/write head as usual, and also extra tapes (with write heads only), which are used for communication from the Turing component to the relational store.

The interaction of the components is twofold:

(i) the Turing component can trigger the execution of certain algebraic operations on the contents of the relational registers, and

(ii) the performance of the Turing control (apart from the usual dependency) also depends on the information whether or not one particular relational register, R_0 say, is empty.

The finite set of operations which can be applied to the relational store in a transition, $\mathcal{O}$ say, depends on the relational type of the input structures, and on the arity r of the relational registers. $\mathcal{O}$ contains the boolean operations, load operations for the input relations and for equality, and the crucial "counting projections". The latter correspond to updates $R_{i_1} := \{(x_1, \dots, x_r) \mid \exists^{\geq i_0} x_j \; R_{i_2} x_1 \dots x_r\}$ in which the relational indices i_1 and i_2 as well as the counting threshold i_0 are all to be regarded as parameters provided from the comunication tapes whenever the operation is called.

Definition 2.12 Let $\mathcal{M}^r[\tau]$ be the class of machines as characterized, with arity r for the registers in the relational store, and formatted for loading finite τ-structures. Let $\mathcal{M}[\tau] := \bigcup_r \mathcal{M}^r[\tau]$. The time or space complexity of an $\mathcal{M}$-computation is defined in terms of the corresponding resource of the Turing component.

The proof of the following theorem is to be found in [19].

Theorem 2.13 *The* PTIME *and* PSPACE *restrictions of the computational model $\mathcal{M}$ correspond to* (FP + C) *and* (PFP + C) *resp., in the sense that e.g. exactly those classes of finite τ-structures are definable by a sentence of* (FP + C)$[\tau]$*, which are accepted by a machine in $\mathcal{M}[\tau]$ in polynomial time.*

3 Fixpoint logic and infinitary logic with counting

Definition 3.1 Let $L^r_{\omega\omega}$ be first-order logic with only r variable symbols. Let $L^r_{\infty\omega}$ be the corresponding infinitary logic, i.e. with infinite disjunctions and conjunctions. Extending these by the set of all the counting quantifiers $\exists^{\geq m}$, for $m \in \omega$, we obtain $C^r_{\omega\omega} := L^r_{\omega\omega}(\exists^{\geq m})_{m\in\omega}$ and $C^r_{\infty\omega} := L^r_{\infty\omega}(\exists^{\geq m})_{m\in\omega}$, respectively. Also put $L^\omega_{\omega\omega} := \bigcup_r L^r_{\omega\omega}$, and similarly for the others: $C^\omega_{\omega\omega}$, $L^\omega_{\infty\omega}$, and $C^\omega_{\infty\omega}$.

Lemma 3.2 (Immerman) (FP + C) $\subsetneq C^\omega_{\infty\omega}$.

PROOF. We want to show that every formula of (FP + C) without free variables of the second sort can be translated into a formula in $C^\omega_{\infty\omega}$. Both the fixpoint generation and the occurences of variables of the second sort have to be expanded in a uniform way which takes into account the size of the structure under consideration. So the formulae we get in $C^\omega_{\infty\omega}$ will be of the form

$$\bigvee_{n\in\omega} (\exists^{\geq n} x\,(x = x) \wedge \neg\exists^{\geq n+1} x\,(x = x) \wedge \varphi_n),$$

so that the φ_n just have to capture the meaning of the given formula over structures of size n.

Remark. It is not difficult to see that this is a normal form for $C^\omega_{\infty\omega}$, with the $\varphi_n \in C_{\omega\omega}$ even, in the sense that any formula in $C^\omega_{\infty\omega}$ is equivalent (in the finite) to one of these. (Use the fact that on every fixed finite structure, every formula of $C^\omega_{\infty\omega}$ is equivalent to one in $C^\omega_{\omega\omega}$.)

We only briefly sketch how to deal with second-sort variables, since the expansion of fixpoint constructions is standard. Consider a formula $\psi(x, \nu)$. W.r.t. ν we expand to a sequence of formulae $(\psi_\nu(x))_{\nu\in\omega}$ such that the mixed-type global relation given by $\psi(x, \nu)$ on structures of size n is just $\bigcup_{\nu\leq n}\{x|\psi_\nu(x)\} \times \{\nu\}$. In this way the translation is straightforward. To indicate one typical step in the inductive treatment, let $\psi(x, \nu) \equiv \nu \leq \#_y \chi(x, y)$. Suppose $\chi'_n(x, y)$ captures the meaning of $\chi(x, y)$ on size n

structures. Then the desired family for ψ consists of $\psi_{n,\nu} :\equiv \exists^{\geq \nu} y\, \chi'_n(x,y)$, for $\nu \leq n$, n the parameter for the size of the structure.

$C^{\omega}_{\infty\omega}$ is much stronger than (FP + C), since it expresses non-recursive properties: For any predicate $R \subset \omega$ consider the sentence $\bigvee_{n\in R}(\exists^{\geq n} x\,(x=x) \wedge \neg\exists^{\geq n+1} x\,(x=x))$.
■

It is possible to isolate the fragment of $C^{\omega}_{\infty\omega}$ which corresponds to (FP + C) precisely. The required restriction is given in terms of a very natural uniformity condition:

Definition 3.3 Let P-$C^{\omega}_{\infty\omega}$ be the sublogic of $C^{\omega}_{\infty\omega}$ which consists of all formulae

$$\bigvee_{n\in\omega} (\exists^{\geq n} x\,(x=x) \wedge \neg\exists^{\geq n+1} x\,(x=x) \wedge \varphi_n),$$

for sequences $(\varphi_n)_{n\in\omega}$ in some $C^{r}_{\omega\omega}$ such that the map $n \mapsto \varphi_n$ is PTIME constructible.

Proposition 3.4 $\qquad$ (FP + C) = P-$C^{\omega}_{\infty\omega}$.

PROOF. One inclusion is settled by the proof of the previous lemma. The proof of the remaining inclusion, from right to left, is easiest as an application of the algorithmic characterization of (FP + C). The machines in $\mathcal{M}^r$ can evaluate formulae corresponding to PTIME sequences in $C^{r}_{\infty\omega}$ quite naturally, see [19]. Alternatively, it can be shown that the PTIME construction of the relevant formula can – by means of a suitable encoding – be simulated over the second, ordered sort in (FP + C); similarly the evaluation of the encoded formula over the first sort is rendered definable in (FP + C). ■

Remark. This is quite unlike the situation for FP itself. Let P-$L^{\omega}_{\infty\omega}$ be the logic consisting of all formulae $\bigvee_{n\in\omega}(\exists^{\geq n} x\, x=x \wedge \neg\exists^{\geq n+1} x\, x=x \wedge \varphi_n)$, for PTIME constructible families in some $L^{r}_{\omega\omega}$, $r \in \omega$. This logic can be shown to be strictly more expressive than FP. Consider the following example. Let $R \subset \omega$ be a numerical predicate. Associate with R the class K_R of structures represented by $\{\mathfrak{A}_{mn} := (n, m, <\restriction_m) \mid m < \log n, m \in R\}$. There are at most $(m+r)^r$ distinct $L^{r}_{\infty\omega}$-types realized over $\mathfrak{A}_{mn}$. It follows from the considerations in [3] that any FP formula over $\mathfrak{A}_{mn}$ closes in time polynomial in $(m+r)^r$ and hence in m. For R such that $m \in R$ is decidable in time 2^m but not in time $p(m)$ for any polynomial p, K_R can thus be shown to be definable in P-$L^{\omega}_{\infty\omega}$, but not in FP.

One of the nice features of the logics $C^{r}_{\infty\omega}$ is the simple Ehrenfeucht-Fraïssé type of characterization of $\equiv_{C^{r}_{\infty\omega}}$, the relation of indistinguishability of structures w.r.t. the logic $C^{r}_{\infty\omega}$. This characterization is best understood in its game formulation, in terms of suitably adapted pebble games. These were introduced in [16].

We fix a relational vocabulary τ. The game is played by two players, **I** and **II**, on a pair of finite τ-structures $(\mathfrak{A}, \mathfrak{A}')$. There are two sets of r pebbles each, one for each structure. Denote them by $p_1, \ldots, p_r$ and $p'_1, \ldots, p'_r$ resp. A move in the game comprises the following steps:

- **I** chooses one of the structures, $\mathfrak{A}$ say, a pair of corresponding pebbles (possibly a pair already placed on the sructures), say (p_j, p'_j), and a non-empty subset of the universe of the chosen structure, say $A_0 \subset A$ — **II** has to answer with a subset of the same size in the other structure, $A'_0 \subset A'$ say, with $|A'_0| = |A_0|$.

- **I** places the pebble p'_j within A'_0 — **II** must place the pebble p_j in A_0.

The game continues as long as the second player can maintain the following condition:

Let $\overline{a}$ and $\overline{a}'$ be the positions of the pebbles in $\mathfrak{A}$ and $\mathfrak{A}'$ resp., then the map taking a_j to a'_j for $j < r$, is a partial isomorphism, i.e. these tuples have the same atomic types in their respective structures.

I wins the game as soon as **II** violates this condition.

It can be shown (cf. [16, 5]) that **II** has a winning strategy in the game on $(\mathfrak{A}, \mathfrak{A}')$ (i.e. a way to maintain the above condition indefinitely) iff $\mathfrak{A}$ and $\mathfrak{A}'$ satisfy exactly the same sentences in $C^r_{\infty\omega}$. Or, that $\overline{a}$ satisfies exactly the same formulae in $\mathfrak{A}$ as does $\overline{a}'$ in $\mathfrak{A}'$ iff **II** has a strategy to continue the game on $(\mathfrak{A}, \mathfrak{A}')$ indefinitely from the situation in which the pebbles are placed on $\overline{a}$ and $\overline{a}'$, respectively. We shall denote the game played in this latter fashion, from starting positions with pebbles initially placed on $\overline{a}$ and $\overline{a}'$, as the game on $(\mathfrak{A}, \overline{a}; \mathfrak{A}', \overline{a}')$.

In the form presented here, the number of possible moves in a given situation is not polynomial since the choice of a set introduces exponential variation. In this sense the games are not suited for an exhaustive search for a strategy (as can be done directly for $L^r_{\infty\omega}$ and the associated ordinary pebble game). For a closer analysis, define the following equivalence relations:

Definition 3.5 Let $\sim^r_i$ be the equivalence relation defined (on the class of finite relational structures with a distinguished r-tuple of elements) through:

$$(\mathfrak{A}, \overline{a}) \sim^r_i (\mathfrak{B}, \overline{b})$$

iff in the game on $(\mathfrak{A}, \overline{a}; \mathfrak{B}, \overline{b})$, player **II** has a strategy to maintain the winning condition for at least i moves. Let $\sim^r$ be the corresponding relation for a strategy in the infinite game, or the common refinement of all the $\sim^r_i$.

It has to be checked of course that this relation is actually transitive. We point out that $\sim^r_0$ is just equality of atomic types.

Lemma 3.6 (Immerman/Lander) $(\mathfrak{A}, \overline{a}) \sim^r (\mathfrak{B}, \overline{b})$ *iff* $(\mathfrak{A}, \overline{a}) \equiv_{C^r_{\infty\omega}} (\mathfrak{B}, \overline{b})$. [1]

The proof is an application of the usual techniques related to Ehrenfeucht-Fraïssé games, see [16].

Another consequence of the transitivity of this notion of equivalence is that $\sim^r_i$ itself factorizes w.r.t. its own restriction to one and the same structure: For a single structure $\mathfrak{A}$, also denote by $\sim^r_i$ the equivalence relation on A^r induced by

$$\overline{a} \sim^r_i \overline{a}' \quad \text{iff} \quad (\mathfrak{A}, \overline{a}) \sim^r_i (\mathfrak{A}, \overline{a}').$$

Writing α, β for arbitrary elements of the quotients $A^r/\sim^r_i$ and $B^r/\sim^r_i$ resp., we use the notation $(\mathfrak{A}, \alpha) \sim^r_i (\mathfrak{B}, \beta)$ to say that for some (and hence all) $\overline{a} \in \alpha$ and $\overline{b} \in \beta$ we have $(\mathfrak{A}, \overline{a}) \sim^r_i (\mathfrak{B}, \overline{b})$.

Say that for $\alpha \in A^r/\sim^r_i$ and $\overline{a} = (a_1, \ldots, a_r) \in A$, α is j-close to $\overline{a}$ if there is an a in A such that $\overline{a}\frac{a}{a_j} \in \alpha$. The crucial point in our analysis of the game is the following claim:

$(\mathfrak{A}, \overline{a}) \sim^r_{i+1} (\mathfrak{B}, \overline{b})$ if and only if the following condition $(*)$ as well as its counterpart, with the roles of the structures exchanged, hold

[1]The notation on the right is to say that the tuples satisfy the same formulae over their respective structures (i.e. it should not be read in the sense of first expanding the structures with constants).

(∗) For all $1 \leq j \leq r$, for all $\alpha \in A^r/\sim^r_i$, which are j-close to $\overline{a}$, there is some $\beta \in B^r/\sim^r_i$, which is j-close to $\overline{b}$, such that:
$(\mathfrak{A}, \alpha) \sim^r_i (\mathfrak{B}, \beta)$ and $\Big| \{a \in A \mid \overline{a}\frac{a}{a_j} \in \alpha\} \Big| = \Big| \{b \in B \mid \overline{b}\frac{b}{b_j} \in \beta\} \Big|$.

Note that the first of the conditions in (∗) already determines the appropriate β uniquely.

The claim follows from an analysis of a single exchange of moves in the game. E.g., in the situation $(\mathfrak{A}, \overline{a}; \mathfrak{B}, \overline{b})$ with pebble pair j and a subset $B_0 \subset B$ chosen by **I**, decompose B_0 into disjoint parts according to: $B_0 = \bigcup_{\beta \in B^r/\sim^r_i} \{b \in B_0 \mid \overline{b}\frac{b}{b_j} \in \beta\}$. The correct response of **II** then is to take portions of matching sizes from the respective $\sim^r_i$-equivalent classes over A to form A_0.

For future reference, we also exhibit the special simplified form which (∗) takes for the case that we are working in just one structure $\mathfrak{A}$:

(∗∗) $(\mathfrak{A}, \overline{a}) \sim^r_{i+1} (\mathfrak{A}, \overline{a}')$ iff for all $1 \leq j \leq r$, and for all $\alpha \in A^r/\sim^r_i$:
$\Big| \{a \in A \mid \overline{a}\frac{a}{a_j} \in \alpha\} \Big| = \Big| \{a \in A \mid \overline{a}'\frac{a}{a'_j} \in \alpha\} \Big|$.

This criterion suggests an inductive process for the generation of the limit $\sim^r$ of the $\sim^r_i$ on a single given structure $\mathfrak{A}$, and even of a linear ordering between the classes. Inductively we define pre-orderings $\preceq^r_i$ on the r-th power of the universe, which induce the equivalence relation $\sim^r$ on that domain (i.e.: $\overline{a} \sim^r_i \overline{a}'$ iff $\overline{a} \preceq^r_i \overline{a}' \wedge \overline{a}' \preceq^r_i \overline{a}$); the pre-ordering is best thought of as a strict linear ordering of the equivalence classes. Let $\prec^r_i$ denote the accompanying strict ordering: $\overline{a} \prec^r_i \overline{a}'$ iff $\overline{a} \preceq^r_i \overline{a}' \wedge \neg \overline{a}' \preceq^r_i \overline{a}$.

The following procedure inductively defines the stages $\prec^r_i$ uniformly on finite $\mathfrak{A}$:

- On the zero level, just fix any linear ordering of the atomic types.
- To pass from $\prec^r_i$ to $\prec^r_{i+1}$, we can inductively suppose that $\prec^r_i$ already induces a strict linear ordering on $A^r/\sim^r_i$, and put for $\overline{a}, \overline{a}' \in A^r$: $\overline{a} \prec^r_{i+1} \overline{a}'$ iff not $\overline{a} \sim^r_{i+1} \overline{a}'$, and for the least j such that condition (∗∗) is violated, and for the $\prec^r_i$-least class $\alpha \in A^r/\sim^r_i$ violating (∗∗) for that j we have: $\Big| \{a \in A \mid \overline{a}\frac{a}{a_j} \in \alpha\} \Big| < \Big| \{a \in A \mid \overline{a}'\frac{a}{a'_j} \in \alpha\} \Big|$.

This process will, in polynomially many steps, lead to saturation. The limit $\prec^r$ of the $\prec^r_i$ thus obtained will be a linear ordering of the equivalence classes w.r.t. $\sim^r$. In other words, this procedure uniformly gives a linear ordering of the $C^r_{\infty\omega}$-types.

A closer analysis of this procedure shows that its outcome $\prec^r$, regarded as a global relation of arity $2r$ on the universe, is almost definable by a formula of FP; the only ingredient FP lacks for this is simple cardinality comparison. The Rescher quantifier exactly introduces this extra expressive power, [20]:

Definition 3.7 The Rescher quantifier Q_R is the Lindström quantifier, which binds two formulae, via one single variable each: From $\varphi_1(x, \overline{z})$ and $\varphi_2(y, \overline{z})$ the new formula $\psi(\overline{z}) \equiv Q_R xy\ (\varphi_1(x, \overline{z}), \varphi_2(y, \overline{z}))$ is formed. Its semantics is defined such that $\psi(\overline{z})$ expresses that $|\{x|\varphi_1(x, \overline{z})\}| < |\{y|\varphi_2(y, \overline{z})\}|$. Let FP($Q_R$) be the extension of FP obtained by closing w.r.t. quantification with Q_R.

Lemma 3.8 *The linear ordering $\prec^r$ can be defined as a global relation of arity $2r$, by a formula in the extension of FP by just the Rescher quantifier, FP(Q_R).*

In particular, $\prec^r$ is definable in (FP + C). Coming back to the analysis of (FP +C), we want to combine the structural information contained in $(\mathfrak{A}, \prec^r)$, in one derived structure associated with $\mathfrak{A}$. Since this abstracted structure will be of a purely numerical nature, we apply the term "arithmetical invariant" to the functor which takes $\mathfrak{A}$ to this derived structure. It is an invariant in the sense that the value depends only on the isomorphism type of $\mathfrak{A}$ (as it should, of course). To prepare the definition of this functor introduce the following notation: For $1 \leq k \leq k' \leq r$, the global relations $\Delta_{kk'} = \{\overline{x} | x_k = x_{k'}\}$ and for $P \in \tau, 1 \leq \bar{k} \leq r$ the global relations $P_{\bar{k}} = \{\overline{x} | Px_{k_1} \dots x_{k_r}\}$ are defined in terms of atomic types. It follows that they factorize w.r.t. $\sim^r$. As these classes have been definably ordered, the above relations can be presented as numerical relations. For $\mathfrak{A}$, let $\alpha_0, \alpha_1, \dots, \alpha_s$ be an enumeration of $A^r / \sim^r$, ordered w.r.t. $\prec^r$. Then put

$$\underline{P_{\bar{k}}}(\mathfrak{A}) := \{i \leq s \,|\, \alpha_i \subset P_{\bar{k}}\},$$

similarly for the Δ. In an analogous way, abstract the relevant counting information from $\mathfrak{A}$, putting, for $1 \leq j \leq r$:

$$\underline{C_j}(\mathfrak{A}) := \Big\{(i, i', k) \;\Big|\; \text{for } \overline{a} \in \alpha_i : |\{a | \overline{a}\frac{a}{a_j} \in \alpha_{i'}\}| = k\Big\}.$$

Definition 3.9 Let I^r be the following functor on the class of all finite τ-structures:

$$I^r\colon \quad \mathfrak{A} \longmapsto \Big(n^r, <, (\underline{\Delta_{kk'}})_{1 \leq k \leq k' \leq r}, (\underline{P_{\bar{k}}})_{P \in \tau, 1 \leq \bar{k} \leq r}, (\underline{C_j})_{1 \leq j \leq r}\Big)(\mathfrak{A}),$$

where n is the cardinality of the domain A.

Theorem 3.10 *For the arithmetical invariant I^r:*

(i) I^r characterizes all finite τ-structures exactly up to $\equiv_{C^r_{\infty\omega}}$:
$\mathfrak{A} \equiv_{C^r_{\infty\omega}} \mathfrak{A}'$ iff $I^r(\mathfrak{A}) = I^r(\mathfrak{A}')$.

(ii) I^r can be computed in PTIME.

(iii) There is a formula of (FP + C) *which globally interprets (an encoding of) $I^r(\mathfrak{A})$ in the second sort of $\mathfrak{A}^*$.*

(iv) (FP + C) = *FP(I^r). Here FP(I^r) is shorthand for the class of those properties of finite τ-structures $\mathfrak{A}$, which are FP-definable over the corresponding $I^r(\mathfrak{A})$.*

PROOF. *(i)* follows from the game characterization of $\equiv_{C^r_{\infty\omega}}$: if $I^r(\mathfrak{A}) = I^r(\mathfrak{B})$ then a strategy for player **I** in the game on $(\mathfrak{A}; \mathfrak{B})$ can be extracted; recall how the equivalence classes of $\sim^r$ relate to the game and how they are represented in I^r.

(ii) is obvious from the FP(Q_R)-definability of $\prec^r$, and the way in which $I^r(\mathfrak{A})$ can be obtained from $(\mathfrak{A}, \prec^r)$ by simple counting.

(iii) again follows directly from observations already made, or from the proof of *(ii)*, together with the fact that $I^r(\mathfrak{A})$ is encoded as a numerical predicate within n^r. In fact, also the projection $\pi : A^r \rightarrow A^r / \sim^r$ is definable in this sense.

(iv): the inclusion from right to left is a consequence of *(iii)*. One way to obtain the converse is presented in [19], where it is shown that the transition to the arithmetical invariant I^r provides a normal form for computations in $\mathcal{M}^r$. Then the result follows from the fact that PTIME(I^r) = FP(I^r) and Theorem 2.13 above.

We here want to indicate a direct proof very briefly: Since (FP + C) $\subset C^{\omega}_{\infty\omega}$ by Lemma 3.2, any relation definable in (FP + C) over $\mathfrak{A}$ will be a union of equivalence classes in $A^r / \sim^r$, for some r. (Here we treat second-sort variables as external parameters as we did in Lemma 3.2.) It is thus possible to replace any such relation R (w.l.o.g. of arity r) by its representation over $I^r(\mathfrak{A})$: $\underline{R} := \{i < n^r \,|\, \alpha_i \subset R\}$, where, as in the definition of I^r, α_i refers to the ordered enumerastion of $A^r / \sim^r$. In this way, any (FP + C) formula can be "simulated" over I^r in a uniform way. The arithmetic governing the variables of the second sort is all PTIME and hence FP-definable over the linearly ordered structures I^r (note that $|I^r(\mathfrak{A})| = n^r$). The evaluation of counting terms, finally, uses just the information encoded in the $\underline{C_j}$ in I^r. ∎

4 Inflationary and partial fixpoints

As pointed out above, in Theorem 2.13, (FP + C) and (PFP + C), or the extensions of *fixpoint* and *while* by counting, correspond to the PTIME and PSPACE restrictions, resp., of the machine model $\mathcal{M}$. In particular, we have that the PTIME restriction of $\mathcal{M}$ captures exactly those properties which are expressible in (PFP + C) under the additional restriction that the PFP-generations must all close in polynomial time.

Definition 4.1 Let PFP $|_P$ stand for this extension of first-order logic by partial fixpoint operators which must close within polynomially many steps. Similarly define (PFP + C)$|_P$.

As already mentioned, from the algorithmic characterization, we directly get:

Proposition 4.2 $\quad$ (PFP + C)$|_P$= (FP + C) .

No generic model is known with equally nice restrictions to *fixpoint* and *while* in the absence of counting. In fact the following result of Abiteboul and Vianu, [3], makes it rather unlikely that a natural model with these restriction properties exists:

Theorem 4.3 (Abiteboul/Vianu) PFP$|_P$= FP *iff* PTIME = PSPACE

Another important result of [3] survives the extension to the case with counting:

Lemma 4.4 *If* PSPACE = PTIME*, then* (PFP + C) = (FP + C).

PROOF. As already indicated above, and excplicitly proved in [19], the properties definable in (PFP + C) or (FP + C) are just those which can be computed in PSPACE or PTIME from the invariants I^r (by an ordinary Turing machine, say). This immediately proves the claim. ∎

Together with the obvious converse to this lemma (consider linearly ordered structures; here (PFP + C) coincides with PFP or PSPACE, similarly (FP + C) = FP = PTIME), we get an extension of the above-mentioned theorem of Abiteboul/Vianu:

Theorem 4.5 $\quad$ (PFP + C) = (FP + C) $\quad$ *iff* $\quad$ PSPACE = PTIME .

References

[1] S. Abiteboul, M. Vardi and V. Vianu, *Fixpoint Logics, Relational Machines and Computational Complexity*, Proceedings of 7th IEEE Symposium on Structure in Complexity Theory (1992).

[2] S. Abiteboul and V. Vianu, *Datalog Extensions for Database Queries and Updates*, J. Comp. Syst. Sciences **43** (1991), 62–124.

[3] S. Abiteboul and V. Vianu, *Generic Computation and Its Complexity*, Proceedings of 23rd ACM Symposium on Theory of Computing (1991).

[4] A. Blass and Y. Gurevich, *Existential fixed-point logic*, in: "Computation Theory and Logic" (E. Börger, Ed.), Lecture Notes in Computer Science Nr. 270, Springer 1987, 20–36.

[5] J. Cai, M. Fürer and N. Immerman, *An Optimal Lower Bound on the Number of Variables for Graph Identification*, Proceedings of 30th IEEE Symposium on Foundations of Computer Science (1989), 612–617.

[6] A. Chandra and D. Harel, *Structure and Complexity of Relational Queries*, J. Comp. Syst. Sciences **25** (1982), 99–128.

[7] E. Dahlhaus, *Skolem Normal Forms Concerning the Least Fixed Point*, in: "Computation Theory and Logic" (E. Börger, Ed.), Lecture Notes in Computer Science Nr. 270, Springer 1987, 101–106.

[8] A. Dawar, S. Lindell and S. Weinstein, *Infinitary Logic and Inductive Definability over Finite Structures*, Technical Report, University of Pennsylvania, (1991).

[9] P. Dublish and S. Maheshwari, *Query Languages which Express all PTIME Queries for Trees*, Hildesheimer Informatik-Berichte 2 (1989).

[10] S. Grumbach and C. Tollu, *The Generic Complexity of Query Languages with Counters*, Proceedings of Third Workshop on Foundations of Models and Languages for Data and Objects, Aigen (Austria) 1991.

[11] Y. Gurevich, *Algebras of feasible functions*, Proceedings of 24$^{\text{th}}$ IEEE Symposium on Foundations of Computer Science 1983, 210–214.

[12] Y. Gurevich, *Logic and the Challenge of Computer Science*, in: E. Börger (Ed), Trends in Theoretical Computer Science, Computer Science Press (1988), 1–57.

[13] Y. Gurevich and S. Shelah, *Fixed Point Extensions of First Order Logic*, Annals of Pure and Applied Logic **32** (1986), 265–280.

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

[15] N. Immerman, *Expressibility as a Complexity Measure: Results and Directions*, Proc. of 2nd Conf. on Structure in Complexity Theory (1987), 194–202.

[16] N. Immerman and E. Lander, *Describing Graphs: A First Order Approach to Graph Canonization*, in: A. Selman (Ed), Complexity Theory Retrospective. (In Honor of Juris Hartmanis), Springer, New York 1990, 59–81.

[17] Ph. Kolaitis, *The Expressive Power of Stratified Logic Programs*, Information and Computation **90** (1991), 50–66.

[18] D. Leivant, *Inductive Definitions over Finite Structures*, Information and Computation **89** (1990), 95–108.

[19] M. Otto, *The Expressive Power of Fixed-Point Logic with Counting*, submitted for publication.

[20] N. Rescher, *Plurality Quantification*, JSL **27** (1962), 373–374.

[21] M. Vardi, *Complexity of Relational Query Languages*, Proc. of 14th ACM Symposium on Theory of Computing (1982), 137–146.