Partial Fixed-Point Logic on Infinite Structures

Stephan Kreutzer

LuFG Mathematische Grundlagen der Informatik
RWTH Aachen
kreutzer@informatik.rwth-aachen.de

Abstract. We consider an alternative semantics for partial fixed-point logic (PFP). To define the fixed point of a formula in this semantics, the sequence of stages induced by the formula is considered. As soon as this sequence becomes cyclic, the set of elements contained in every stage of the cycle is taken as the fixed point. It is shown that on finite structures, this fixed-point semantics and the standard semantics for PFP as considered in finite model theory are equivalent, although arguably the formalisation of properties might even become simpler and more intuitive. Contrary to the standard PFP semantics which is only defined on finite structures the new semantics generalises easily to infinite structures and transfinite inductions. In this generality we compare - in terms of expressive power - partial with other known fixed-point logics. The main result of the paper is that on arbitrary structures, PFP is strictly more expressive than inflationary fixed-point logic (IFP). A separation of these logics on finite structures would prove PTIME different from PSPACE.

1 Introduction

Logics extending first-order logic by fixed-point constructs are well studied in finite model theory. Introduced in the early eighties, it soon became clear that there are tight connections between the various forms of fixed-point logics and such important complexity classes as polynomial time and space. This relationship is made precise in the results by Immerman [Imm86] and Vardi [Var82] that, on finite ordered structures, *least fixed-point logic* (LFP) provides a logical characterisation of polynomial time computations in the sense that a class of finite ordered structures is decidable in polynomial time if, and only if, it is definable in LFP. Other complexity classes such as polynomial or logarithmic space can also be characterised in this way, using different fixed-point logics. Since the discovery of these results, fixed-point logics play a fundamental role in finite model theory, arguably even more important than first-order logic itself. We give precise definitions of these logics in Section 2. See [EF99] for an extensive study of fixed-point logics on finite structures. A survey that also treats infinite structures can be found in [DG02].

The best known of these logics is least fixed-point logic (LFP), which extends first-order logic (FO) by an operator to form least fixed-points of positive formulae (which define monotone operators.) But there are other fixed-point

logics. Besides fragments of LFP, such as transitive closure logic and existential or stratified fixed-point logic, which all have in common that they form fixed points of monotone operators, there are also fixed-point logics that allow the use of non-monotone operators. One such logic is the *inflationary fixed-point logic* (IFP), which allows the definition of inflationary fixed points of arbitrary formulae. It is the simplest logic allowing non-monotone operators, as it is still equivalent to LFP (see [GS86,Kre02].)

As mentioned above, on finite ordered structures, LFP and IFP capture PTIME. To characterise complexity classes above PTIME, like PSPACE for instance, a more liberal notion of fixed points has to be used. One such logic that is likely to be more expressive than IFP is *partial fixed-point logic*, where there are no restrictions on the formulae used within the fixed point operator. Thus it is no longer guaranteed that the sequence of stages induced by such a formula reaches a fixed point. However, if it does, this fixed point is taken as the semantics of the formula. Otherwise, i.e. if the sequence does not become stationary, the result is defined as being empty.

It has been shown by Abiteboul and Vianu [AV91a] that partial fixed-point logic provides a precise characterisation of PSPACE on finite ordered structures. Thus, showing that there are properties of finite ordered structures definable in PFP but not in IFP would yield a separation of polynomial time and space. However, on unordered structures, neither IFP nor PFP can express all of PTIME. For instance, it is easy to see that it cannot be decided in PFP whether a finite set is of even cardinality, a problem that from a complexity point of view is extremely simple. It is therefore remarkable that a separation of PTIME and PSPACE follows even from a separation of IFP and PFP on arbitrary finite structures, not necessarily being ordered. This result is due to Abiteboul and Vianu [AV91b]. See also [Daw93].

Theorem. PTIME = PSPACE *if, and only if,* IFP = PFP.

There are also fixed-point logics capturing the complexity classes NP and EXPTIME, namely *non-deterministic* and *alternating non-inflationary fixed-point logic* (see [AVV97].) For these logics, similar theorems as above have been shown. Thus, the most important questions in complexity theory, the separation of complexity classes, have direct analogues in logic, namely in the comparison of the expressive power of various fixed-point logics. A profound understanding of the nature and limits of the various kinds of fixed-point operators is therefore important and necessary. In this line of research, the main contribution of this paper is to introduce a semantics for partial fixed-point logic that is equivalent to the standard semantics on finite structures, but contrary to the standard semantics, is also well defined on infinite structures. On infinite structures, we will then be able to compare partial and inflationary fixed-point logic and show that there are properties definable in PFP which are not definable in IFP. Thus, IFP is strictly contained in PFP. We also argue that the alternative semantics for PFP allows a more intuitive formulation of queries than the standard semantics.

2 Preliminaries

In this section we present the basic definitions for the explorations in the later sections. Let τ be a signature and $\mathfrak{A} := (A, \tau)$ be a τ-structure with universe A. Let $\varphi(R, \overline{x})$ be a first-order formula with free variables $\overline{x}$ and a free relation symbol R not occurring in τ. The formula φ defines an operator

$$\begin{aligned} F_\varphi : \mathcal{P}(A) &\longrightarrow \mathcal{P}(A) \\ R &\longmapsto \{\overline{a} : (\mathfrak{A}, R) \models \varphi[\overline{a}]\}. \end{aligned}$$

A fixed point of the operator F_φ is any set R such that $F_\varphi(R) = R$. Clearly, as φ is arbitrary, the corresponding operator F_φ need not to have any fixed points at all. For instance, the formula $\varphi(R, \overline{x}) := \neg\forall\overline{y}\, R\overline{y}$ defines the operator F_φ mapping any set $R \subsetneqq A^k$ to A^k and the set A^k itself to the empty set. Thus F_φ has no fixed points. However, if the class of admissible formulae is restricted, the existence of fixed points can be guaranteed. One such restriction is to require that the formulae are positive in the fixed-point variable. As positiveness implies monotonicity, an operator F_φ defined by a positive formula φ always has fixed points, in fact even a least fixed point $\mathbf{lfp}(F_\varphi) := \bigcap\{P : F_\varphi(P) = P\}$. This forms the basis of the most common fixed-point logic, the least fixed-point logic.

To obtain more general logics, i.e. logics allowing non-monotone operators also, one has to consider suitable semantics to guarantee the existence of meaningful fixed-points. The simplest such logic is the *inflationary fixed-point logic*.

Definition 2.1 (Inflationary Fixed-Point Logic). *Inflationary fixed-point logic* (IFP) is defined as the extension of first-order logic by the following formula building rule. If $\varphi(R, \overline{x})$ is a formula with free first-order variables $\overline{x} := x_1, \ldots, x_k$ and a free second-order variable R of arity k, then

$$\psi := [\mathbf{ifp}_{R,\overline{x}}\, \varphi](\overline{t})$$

is also a formula, where $\overline{t}$ is a tuple of terms of the same length as $\overline{x}$. The free variables of ψ are the variables occurring in $\overline{t}$ and the free variables of φ other than $\overline{x}$.

Let $\mathfrak{A}$ be a structure with universe A providing an interpretation of the free variables of φ other than $\overline{x}$. Consider the following sequence of sets induced by φ on $\mathfrak{A}$.

$$\begin{aligned} R^0 &:= \emptyset \\ R^{\alpha+1} &:= R^\alpha \cup F_\varphi(R^\alpha) \\ R^\lambda &:= \bigcup_{\beta<\lambda} R^\beta \qquad \text{for limit ordinals } \lambda. \end{aligned}$$

The sets R^α are called the *stages* of the induction on φ and $\mathfrak{A}$. Clearly the sequence of stages is increasing and thus leads to a fixed point R^∞. For any tuple $\overline{a} \in A$,

$$\mathfrak{A} \models [\mathbf{ifp}_{R,\overline{x}}\, \varphi](\overline{a}) \text{ if, and only if, } \overline{a} \in R^\infty.$$

As usual, we also allow simultaneous fixed-point formulae, i.e. formulae of the form $\psi(\overline{x}) := [\mathbf{ifp}\ R_i : S](\overline{x})$, where

$$S := \begin{cases} R_1\overline{x}_1 \leftarrow \varphi_1(R_1, \ldots, R_k, \overline{x}_1) \\ \qquad\vdots \\ R_k\overline{x}_k \leftarrow \varphi_k(R_1, \ldots, R_k, \overline{x}_k) \end{cases}$$

is a system of formulae. Each formula φ_i in S induces an operator $F_{\varphi_i} : \mathrm{Pow}(A)^{r_1} \times \cdots \times \mathrm{Pow}(A)^{r_k} \to \mathrm{Pow}(A)^{r_i}$, taking sets $R_1, \ldots, R_k$ of appropriate arity to the set $\{\overline{a} : (\mathfrak{A}, R_1, \ldots, R_k) \models \varphi_i[\overline{a}]\}$, where the r_i denote the arities of the relations R_i. The stages of an induction on such a system S of formulae are now k-tuples of sets defined by

$$\begin{aligned} R_i^0 &:= \emptyset \\ R_i^{\alpha+1} &:= R_i^\alpha \cup F_{\varphi_i}(R_1^\alpha, \ldots, R_k^\alpha) \\ R_i^\lambda &:= \bigcup_{\beta<\lambda} R_i^\beta \qquad \text{for limit ordinals } \lambda. \end{aligned}$$

The formula ψ is true for a tuple $\overline{a}$ of elements interpreting the variables $\overline{x}$ if, and only if, $\overline{a} \in R_i^\infty$, where R_i^∞ denotes the i-th component of the simultaneous fixed point of the system S. Simultaneous inductions can easily be eliminated in favour of simple inductions by increasing the arity of the involved fixed-point variables (See [EF99].)

Proposition 2.2. *Any formula in* IFP *with simultaneous inductions is equivalent to a formula without simultaneous inductions.*

Nevertheless, formulae making use of simultaneous inductions are often much simpler to read than the equivalent simple formulae and we will extensively use simultaneous inductions in the sequel.

3 Partial Fixed-Point Logic

In this section we introduce *partial fixed-point logic*, which in some sense is the most general fixed-point extension of first-order logic. We first define the syntax, which is the same as for IFP, except that we write **pfp** for the fixed-point operator.

Definition 3.1 (Partial Fixed-Point Logic - Syntax). Partial fixed-point logic (PFP) *is defined as the extension of first-order logic by the following formula building rule. If $\varphi(R, \overline{x})$ is a formula with free first-order variables $\overline{x} := x_1, \ldots, x_k$ and a free second-order variable R of arity k, then*

$$\psi := [\mathbf{pfp}_{R,\overline{x}}\,\varphi](\overline{t})$$

is also a formula, where $\overline{t}$ is a tuple of terms of the same length as $\overline{x}$. The free variables of ψ are the variables occurring in $\overline{t}$ and the free variables of φ other than $\overline{x}$.

Having defined the syntax, we now turn to the definition of the semantics. We first present the standard definition of partial fixed-point semantics as common in finite model theory.

Definition 3.2 (Finite Model Semantics). *Let $\psi := [\mathbf{pfp}_{R,\overline{x}}\varphi](\overline{t})$ be a formula and let $\mathfrak{A}$ be a finite structure with universe A providing an interpretation of the free variables of φ other than $\overline{x}$. Consider the following sequence of stages induced by φ on $\mathfrak{A}$.*

$$R^0 := \emptyset$$
$$R^{\alpha+1} := F_\varphi(R^\alpha)$$

As there are no restrictions on φ, this sequence need not reach a fixed point. In this case, ψ is equivalent on $\mathfrak{A}$ to false. Otherwise, if the sequence becomes stationary and reaches a fixed point R^∞, then for any tuple $\overline{a} \in A$,

$$\mathfrak{A} \models [\mathbf{pfp}_{R,\overline{x}}\varphi](\overline{a}) \text{ if, and only if, } \overline{a} \in R^\infty.$$

Again we allow simultaneous inductions and as with IFP these can always be eliminated in favour of simple inductions. This semantics for PFP is standard in finite model theory and the basis of the results mentioned in the introduction. However, actually writing a formula in this logic is sometimes unnecessarily complicated. This is demonstrated by an example for *modal* partial fixed-point logic. The example is taken from [DK] where also more on modal partial fixed-point logic can be found.

We briefly recall the definition of modal logic and its extension by partial fixed-point operators. Modal logics are interpreted on *transition systems*, also called *Kripke structures*, which are edge and node labelled graphs. The labels of the edges come from a set $\mathcal{A}$ of actions, whereas the nodes are labelled by sets of propositions from a set $\mathcal{P}$.

Modal logic (ML) is built up from atomic propositions $p \in \mathcal{P}$ using boolean connectives $\wedge$, $\vee$, and $\neg$ and the so-called *next-modalities* $\langle a\rangle$, $[a]$ for each $a \in \mathcal{A}$. Formulae $\varphi \in$ ML are evaluated at a particular node in a transition system. We write $\mathcal{K}, v \models \varphi$ if φ holds at the node v in the transition system $\mathcal{K} := (V, (E_a)_{a\in\mathcal{A}}, (p)_{p\in\mathcal{P}})$. The semantics of ML-formulae is as usual with $\mathcal{K}, v \models p$, for $p \in \mathcal{P}$, if $v \in p^{\mathcal{K}}$, $\mathcal{K}, v \models \langle a\rangle\varphi$ if there is an a-successor u of v such that $\mathcal{K}, u \models \varphi$ and, dually, $\mathcal{K}, v \models [a]\varphi$ if for all a-successors u of v, $\mathcal{K}, u \models \varphi$. Now modal partial fixed-point logic (MPC) is defined analogously to PFP, i.e. formulae $\psi := [\mathbf{pfp}\ P : \varphi(P)]$ are allowed defining the set of elements in the partial fixed point of φ.

Consider the following problem, known as the *unary trace- or language equivalence problem*. It is defined as the problem of deciding whether two given finite automata over an unary alphabet accept the same language. This is formalised as follows. The input is a directed, rooted graph. The root is labelled by w and is not reachable from any other node in the graph. Further, there are disjoint subgraphs rooted at successors of the root. In each subgraph some nodes are marked as final states, e.g. coloured by a colour f, whereas the other nodes are

not coloured at all. Two subgraphs rooted at successors of the root are *trace equivalent*, if for each $n < \omega$, whenever in one of the graphs there is a path of length n from the root to a final state such a path also exists in the other.

We aim at defining in MPC the class $\mathcal{C}$ of structures as above such that all subgraphs rooted at successors of the root are trace equivalent. A simple idea to formalise this is the following. Consider the formula ψ defined as

$$\psi := [\mathbf{pfp}\ Z : \begin{array}{l} X \leftarrow (f \wedge \neg Y) \vee \Diamond X \\ Y \leftarrow f \\ Z \leftarrow (w \wedge \Diamond X \wedge \Diamond \neg X) \vee Z \end{array}]$$

In the first stage, X contains all final states, i.e. those labelled by f. In the successive stages, those elements are selected, which have a successor in X. Thus, the stage X^n contains exactly those elements from which there is a path of length $n-1$ to a node labelled by f. The variable Y is only used to ensure that the nodes labelled by f are added to X only once at the beginning, so that the induction is not started over and over again. Now, the root of the structure is added to Z if, for some n, in one subgraph there is a path of length n from its root to a final state but not in the other. Obviously, once the root is added to Z, it stays in forever. Thus, ψ is true at the root if, and only if, the subgraphs rooted at its successors are not trace equivalent. However, if at least one of the sub-structures is cyclic, the induction on X never becomes stationary and thus, by definition, the fixed point is empty. To rescue the formula, we have to think about some way to guarantee that the induction process becomes stationary although the only information we are interested in, namely whether the root eventually occurs in Z, is independent of this.

This suggest a different way to define partial fixed-point inductions. Consider the sequence of induction stages defined by ψ. Obviously, this sequence must eventually become cyclic. Now consider the set of elements that occur in all stages of this cycle and take this as the defined fixed point[1]. Applying this idea to the example above, we get that the fixed point of X becomes empty (unless there are self loops), the fixed point of Y contains all final states, and the fixed point of Z contains the root just in case there are two successors of it which are not trace equivalent. Thus, $\neg\psi$ is true in $\mathcal{K}, v$ if, and only if, $\mathcal{K}, v \in \mathcal{C}$. This motivates an alternative semantics for partial fixed-point logic based on these ideas.

Besides this problem of formalising properties, the standard semantics for PFP has the disadvantage that it does not generalise to infinite structures. For instance, as the sequence of stages induced by PFP-formulae is not necessarily increasing, it makes no sense to define limit stages as the union of the previous stages as in IFP. Therefore, so far partial fixed-point logic has only been considered on finite structures.

The drawback of this is that it also restricts the possibilities to study PFP and its properties and to compare it to other logics to finite structures. As

[1] Note that this set does not necessarily has to be a fixed point. Nevertheless we use this name to keep consistent with the other fixed-point logics.

mentioned in the introduction, the relationship between the various fixed-point logics is closely related to important complexity theoretical questions and thus a profound understanding of what the logics can and can not do is necessary and important. To achieve a better understanding of the logics, their properties on infinite structures might prove useful for the study on finite structures also. This is the second motivation for considering an alternative semantics for PFP, namely to give a semantics that generalises to infinite structures and transfinite inductions.

We are now ready to formally define a general semantics for partial fixed-point logic.

Definition 3.3 (General Semantics). *Let $\psi := [\mathbf{pfp}_{R,\overline{x}}\varphi](\overline{t})$ be a formula and let $\mathfrak{A}$ be a structure with universe A providing an interpretation of the free variables of φ other than $\overline{x}$. Consider the following sequence of stages induced by φ on $\mathfrak{A}$.*

$$\begin{aligned} R^0 &:= \emptyset \\ R^{\alpha+1} &:= F_\varphi(R^\alpha) \\ R^\lambda &:= \text{final}((R^\alpha)_{\alpha<\lambda}) \qquad \textit{for limit ordinals } \lambda, \end{aligned}$$

where $\text{final}((R^\alpha)_{\alpha<\lambda})$ *denotes the set of elements $\overline{a}$ such that there is a $\beta < \lambda$ and for all $\beta < \gamma < \lambda$, $\overline{a} \in R^\gamma$.*

Obviously, the sequence $(R^\alpha)_{\alpha\in\text{Ord}}$ must eventually become cyclic. Let $\beta_1 < \beta_2$ be minimal such that $R^{\beta_1} = R^{\beta_2}$. Then, for any tuple $\overline{a} \in A$,

$$\mathfrak{A} \models [\mathbf{pfp}_{R,\overline{x}}\varphi](\overline{a}) \textit{ if, and only if, } \overline{a} \in R^\gamma \textit{ for all } \beta_1 \leq \gamma < \beta_2.$$

We also allow simultaneous inductions and again the proof that this does not increase the expressive power is straight forward.

Theorem 3.4. *Any formula in* PFP *under the general semantics with simultaneous inductions is equivalent to a formula without simultaneous inductions.*

According to the definition, the fixed point of a formula φ is defined as the set of elements which occur in every stage of the first cycle in the sequence of stages induced by φ. Note that this is not equivalent to saying that the fixed point consists of those elements $\overline{a}$ such that there is a stage β and $\overline{a}$ occurs in all stages greater than β. For instance, consider a structure $\mathfrak{A} := (\{0,1,2,3\})$ and the formula defining an operator taking $\emptyset \mapsto \{0,1\}$, $\{0,1\} \mapsto \{0,2\}$ and $\{0,2\} \mapsto \{0,1\}$. Further, it takes $\{0\} \mapsto \{2\}$ and $\{2\}$ to itself. Now consider the induction stages $(R^\alpha)_{\alpha\in\text{Ord}}$ induced by this operator. Clearly, for all $0 < n < \omega$, $R^n = \{0,1\}$ if n is odd and $R^n = \{0,2\}$ if n is even. Thus, the partial fixed point as defined above is $\{0\}$. However, $R^\omega = \{0\}$ and for all $\alpha > \omega$, $R^\alpha = \{2\}$. Thus, defining the fixed point as the set of elements which are contained in all stages greater than some β yields a different set than the partial fixed point as defined above.

We now prove that in the restriction to finite structures both semantics, i.e. the semantics in Definition 3.2 and 3.3 are equivalent.

Notation. *To distinguish between the two semantics, we denote* PFP *under the finite model semantics as* $\mathrm{PFP}_{\mathrm{fin}}$ *and write the operator as* $\mathbf{pfp}^f$. *We write* $\mathrm{PFP}_{\mathrm{gen}}$ *and* $\mathbf{pfp}^g$ *whenever we speak about the general semantics. Further, if* φ *is any formula in* PFP, *we write* $fin(\varphi)$ *to denote the formula under the finite model semantics and* $gen(\varphi)$ *for the general semantics.*

We first prove a technical lemma that establishes the main step for the proof of the theorem below.

Lemma 3.5. *Let* $\varphi(R, \overline{x})$ *be a formula in* $\mathrm{PFP}_{\mathrm{gen}}$ *and* $\mathfrak{A}$ *be a structure. There is a formula* $\text{fixed-point}_\varphi(R, \overline{x})$ *depending on* φ *such that for any stage* R^α *of the induction on* φ *and* $\mathfrak{A}$ *and all* $\overline{a} \in A$,

$$(\mathfrak{A}, R^\alpha) \models \text{fixed-point}_\varphi[\overline{a}] \qquad \textit{iff} \qquad \begin{array}{l}\textit{there are } \beta < \gamma \leq \alpha \textit{ such that } (R^\xi)_{\beta \leq \xi \leq \gamma} \\ \textit{is a cycle, i.e. } R^\beta = R^\gamma, \textit{ and } \overline{a} \in \varphi^\infty.\end{array}$$

Further, if $\mathfrak{A}$ *is finite and* $\varphi \in \mathrm{PFP}_{\mathrm{fin}}$, *then* $fin(\text{fixed-point}_\varphi) \equiv gen(\text{fixed-point}_\varphi)$, *i.e. the result of* $\text{fixed-point}_\varphi$ *under the finite model and the general semantics is the same.*

Proof. Consider the formula $\textit{fixed-point}_\varphi(R, \overline{x}) := [\mathbf{pfp}\ Q_2 : S](\overline{x})$, where S is defined as

$$S := \left\{ \begin{array}{l} Q\overline{x} \ \leftarrow \varphi(Q, \overline{x}) \\ Q_1\overline{x} \leftarrow (Q_1 = \emptyset \wedge Q = R \wedge R\overline{x}) \vee Q_1\overline{x} \\ Q_2\overline{x} \leftarrow Q_2\overline{x} \vee (Q_1 \neq \emptyset \wedge Q = R \wedge \\ \qquad\qquad Z \ \leftarrow (Z = \emptyset \wedge \varphi(R, \overline{x})) \vee (Z = R \wedge R\overline{x}) \vee \\ \qquad [\mathbf{pfp}\ Z' : \quad (Z \neq \emptyset \wedge Z \neq R \wedge \varphi(Z, \overline{x})) \qquad\qquad](\overline{x})) \\ \qquad\qquad Z' \leftarrow (Z' = \emptyset \wedge Q_1\overline{x}) \vee (Z' \neq \emptyset \wedge Z'\overline{x} \wedge Z\overline{x}) \end{array} \right.$$

In the course of the induction on S, the variable Q runs through the stages of φ. The first time where $Q = R$, i.e. the stage R is reached, Q_1 is initialised to R. If there is another stage in the induction on Q such that $Q = R$, i.e. if the induction on φ becomes cyclic the first time, Q_2 gets all elements which are contained in all stages between the two occurrences of R. Thus, the fixed point Q_2^∞ contains exactly the elements of the fixed point of φ. □

We are now ready to prove the equivalence of the two partial fixed-point semantics defined above.

Theorem 3.6. *On finite structures,* $\mathrm{PFP}_{\mathrm{fin}}$ *and* $\mathrm{PFP}_{\mathrm{gen}}$ *are equivalent, i.e. for every* PFP*-formula under the finite model semantics there is an equivalent* PFP*-formula under the general semantics and vice versa.*

Proof. The forth direction follows easily by induction on the structure of the formula. In the main step, let $\psi := [\mathbf{pfp}^f_{R,\overline{x}}\, \varphi(R, \overline{x})](\overline{t})$ be a formula in $\mathrm{PFP}_{\mathrm{fin}}$. It is equivalent to the formula

$$\psi^g := [\mathbf{pfp}^g\ Q : \begin{array}{l} R\overline{x} \leftarrow \varphi^g(R, \overline{x}) \\ Q\overline{x} \leftarrow \forall \overline{x}(\varphi^g(R, \overline{x}) \leftrightarrow R\overline{x}) \wedge R\overline{x}. \end{array}](\overline{t})$$

where φ^g is a $\mathrm{PFP}_{\mathrm{gen}}$-formula equivalent to φ. By induction, such a formula always exists. Assume first that a fixed point of φ is reached on a structure $\mathfrak{A}$. In this case, both semantics are equivalent for trivial reasons and thus $\psi \equiv \psi^g$. Now assume that the fixed point of φ does not exist. Then at no stage $\forall \overline{x}(\varphi^g(R, \overline{x}) \leftrightarrow R\overline{x})$ becomes true and thus ψ^g defines the empty set.

The other direction is also proved by induction on the structure of the formulae. In the main step, assume that $\psi := [\mathbf{pfp}^g_{R,\overline{x}} \varphi(R, \overline{x})](\overline{t})$ is a formula under the general semantics. By induction, φ is equivalent to a formula φ^f in $\mathrm{PFP}_{\mathrm{fin}}$. Then, $[\mathbf{pfp}^g_{R,\overline{x}} \varphi^g(R, \overline{x})]\overline{t}$ is equivalent to

$$\psi^f := [\mathbf{pfp}^f \; Q : \begin{array}{l} R\overline{x} \leftarrow \varphi^f(R, \overline{x}) \\ Q\overline{x} \leftarrow \mathit{fixed\text{-}point}_{(\varphi^f)}(R, \overline{x}) \end{array}]\overline{t}$$

By Lemma 3.5, the formula $\mathit{fixed\text{-}point}_{(\varphi^f)}(R)$ can be chosen from $\mathrm{PFP}_{\mathrm{fin}}$. Thus, as $\varphi^f \in \mathrm{PFP}_{\mathrm{fin}}$, we get that ψ^f is itself a formula in $\mathrm{PFP}_{\mathrm{fin}}$. The equivalence of ψ^f and ψ is an immediate consequence of Lemma 3.5. □

The theorem allows us to transfer the results on $\mathrm{PFP}_{\mathrm{fin}}$ mentioned in the introduction, in particular the theorems by Abiteboul, Vianu, Immerman, and Vardi to $\mathrm{PFP}_{\mathrm{gen}}$. Thus, we immediately get the following corollary.

Corollary 3.7.

(i) $\mathrm{PFP}_{\mathrm{gen}}$ *has* PSPACE *data-complexity and captures* PSPACE *on ordered structures.*

(ii) $\mathrm{PFP}_{\mathrm{gen}} = \mathrm{IFP}$ *on finite structures if, and only if,* PTIME = PSPACE.

(iii) On finite structures, every $\mathrm{PFP}_{\mathrm{gen}}$ *formula is equivalent to a formula with only one application of a fixed-point operator.*

Proof. The corollary follows immediately from the fact that every $\mathrm{PFP}_{\mathrm{fin}}$ formula is equivalent to one with only one fixed-point operator and that the translation of $\mathrm{PFP}_{\mathrm{fin}}$-formulae to $\mathrm{PFP}_{\mathrm{gen}}$-formulae as presented in the proof of Theorem 3.6 does not increase the number of fixed-point operators. □

Using a diagonalisation argument as in Section 4 below, it is clear that for any fixed-point logic like LFP, IFP, or PFP, the alternation or the nesting depth hierarchy must be strict on arbitrary structures, i.e. allowing the nesting of fixed-point operators or the alternation of fixed-point operators and negation must strictly increase the expressive power. Thus, Part *(iii)* of the preceding corollary fails on infinite structures. We close the section by establishing a negation normal form for $\mathrm{PFP}_{\mathrm{gen}}$ formulae. Thus, the alternation of fixed points and negation does not provide more expressive power than just nesting fixed-points.

Theorem 3.8. *Every* $\mathrm{PFP}_{\mathrm{gen}}$ *formula is equivalent to one where negation occurs only in front of atoms.*

Proof. The proof follows easily using the formula defined in Lemma 3.5. However, we present a general proof for this that also works for IFP and shows that for

these logics the concept of negated fixed points does not add anything to the expressive power.

Let $\psi(\overline{t}) := \neg[\mathbf{pfp}_{R,\overline{x}}\varphi(R,\overline{x})](\overline{t})$ be a formula in PFP. Obviously, it is equivalent to the formula

$$\psi'(\overline{t}) := \exists 0 \exists 1\ [\mathbf{pfp}\ Q : \begin{array}{l} P\overline{x}y \leftarrow y = 1 \vee (y = 0 \wedge [\mathbf{pfp}_{R,\overline{x}}\varphi](\overline{x})) \\ Q\overline{x} \ \leftarrow P \neq \emptyset \wedge \neg P\overline{x}0 \end{array}](\overline{t}),$$

where $0, 1$ are variables not occurring in φ. The theorem now follows immediately by induction on the structure of the formulae. □

As discussed above, this implies that nesting fixed points strictly increases the expressive power, i.e. nested fixed points can not be eliminated in favour of a single fixed point.

4 Separating partial and inflationary fixed-point logic

In this section we prove the main result of this paper, the separation of $\mathrm{PFP}_{\mathrm{gen}}$ and IFP. As we are not considering the finite model semantics anymore, we simply write PFP and **pfp** instead of $\mathrm{PFP}_{\mathrm{gen}}$ and $\mathbf{pfp}^g$.

We first present a class of structures called *acceptable* (See [Mos74, Chapter 5].) These structures are particularly well suited to be used with diagonalisation arguments.

4.1 Acceptable structures

Definition 4.1. *Let A be an infinite set. A* coding scheme *on A is a triple $(\mathcal{N}, \leq, <>)$, for some $\mathcal{N} \subseteq A$, where the structure $(\mathcal{N}, \leq)$ is isomorphic to $(\omega, \leq)$ and $<>$ is an injective map of $\bigcup_{n<\omega} A^n$ into A.*

With each coding scheme we associate the following decoding relations and functions:

- *(i) $seq(x)$ which is true for x if, and only if, x is the code of some sequence $x_1, \ldots, x_n$.*
- *(ii) $lh(x) = n$ if x is the code of a sequence of length n and otherwise, i.e. if $\neg\mathrm{seq}(x)$, $lh(x) = 0$.*
- *(iii) $q(x,i) = x_i$ if $x =< x_1, \ldots, x_l >$ and $l \geq i$. Otherwise $q(x,i) = 0$. We write $(x)_i = a$ for $q(x,i) = a$.*

Here, the numbers $0, 1, \ldots$ refer to the corresponding elements in $\mathcal{N}$.

An elementary coding scheme *$\mathcal{C}$ on a structure $\mathfrak{A}$ is a coding scheme on its universe where the relations $\mathcal{N}, \leq, seq, lh$, and q are elementary, i.e., first-order definable.*

A structure $\mathfrak{A}$ admitting an elementary coding scheme is called acceptable. *We call $\mathfrak{A}$* quasi-acceptable *if there exists an acceptable expansion $\mathfrak{A}'$ of $\mathfrak{A}$ by a finite set of* PFP*-definable relations.*

Observe that quasi-acceptable structures are those which admit an PFP-definable coding scheme, i.e., one where the relations $<$, seq, lh, and q are PFP-definable. See [Mos74, Chapter 5] for more on elementary and inductive coding schemes.

4.2 Coding and Diagonalisation

We show now how formulae can be encoded by elements of acceptable structures. For the rest of this section let $\mathfrak{A}$ be an acceptable τ-structure, where $\tau := \tau_{\text{rel}} \dot\cup \tau_{\text{const}}$ is the disjoint union of a finite set $\tau_{\text{rel}} := \{P_1, \ldots, P_l\}$ of relation symbols and a finite set $\tau_{\text{const}} := \{c_1, \ldots, c_m\}$ of constant symbols. W.l.o.g. we assume that no fixed-point variable is bound twice in the same formula and that the involved fixed-point variables R_i are numbered from 1 to the number k of fixed-point operators occurring in the formula such that for no $i < j \leq k$, φ_i is a sub-formula of φ_j, where φ_i and φ_j are the formulae defining the fixed point inductions on R_i and R_j respectively. Further, we assume that all formulae are of the form $[\mathbf{ifp}_{R_1,\overline{x}_1}\, \varphi_1](\overline{x}_1)$. We also assume that all fixed-point operators are of the form $[\mathbf{ifp}_{R,\overline{x}}\, R\overline{x} \vee \varphi(R,\overline{x})]$, i.e. the operators are syntactically made inflationary. Finally, we assume that if $\psi := [\mathbf{ifp}_{R,x_{i_1},\ldots,x_{i_k}}\, \varphi]$ occurs as a sub-formula of a formula χ, then the sub-formulae of φ may use atoms in which R occurs only in the form $Rx_{i_1}, \ldots, x_{i_k}$. It is clear that any IFP-formula can be brought into this form.

The actual encoding of formulae is based on a function $||\varphi||$ taking formulae or terms in IFP$[\tau]$ to elements of $\mathcal{N}$. The function is inductively defined as follows.

$$
\begin{array}{rll}
||c_i|| & := < \mathbf{c}, i > & c_i \in \tau_{\text{const}} \\
||x_i|| & := < \mathbf{var}, i > & \\
||P_i\overline{a}|| & := < \mathbf{rel}, i, < ||\overline{a}|| >> & P_i \in \tau_{\text{Rel}} \\
||\varphi_1 \vee \varphi_2|| & := < \mathbf{or}, ||\varphi_1||, ||\varphi_2|| > & \\
||\neg\varphi|| & := < \mathbf{neg}, ||\varphi|| > & \\
||R_i\overline{a}|| & := < \mathbf{fp\text{-}var}, i, < ||\overline{a}|| >> & \text{for fixed-point variables } R_i \\
||\,[\mathbf{ifp}_{R_i,\overline{x}}\,\varphi](\overline{a})\,|| & := < \mathbf{fp\text{-}op}, i, < ||\overline{a}|| >>, &
\end{array}
$$

where $\mathbf{c}, \mathbf{var}, \ldots$ denote arbitrary but fixed and distinct elements of $\mathcal{N}$. Here $< ||\overline{a}|| >$ is an abbreviation for $< ||a_1||, \ldots, ||a_k|| >$ where k is the arity of $\overline{a}$. In this encoding of formulae, sub-formulae involving fixed-point variables are only coded by the number of the involved fixed-point variable but no code of the formula defining it is stored. The next definition deals with this.

Definition 4.2. *Let φ be a formula in* IFP$[\tau]$ *and let the fixed-point operators occurring in it be $[\mathbf{ifp}_{R_1,\overline{x}_1}\, \varphi_1]$, $\ldots$, $[\mathbf{ifp}_{R_n,\overline{x}_n}\, \varphi_n]$. The formulae φ_i, for $1 \leq i \leq n$, are called the* defining formulae of φ *and each individual φ_i is called the* defining formula of the fixed-point variable R_i.

The function code taking formulae to their codes in $\mathcal{N}$ is defined as

$$
\begin{array}{rcl}
\mathit{code}: \text{IFP}[\tau] & \longrightarrow & \mathcal{N} \\
\varphi & \longmapsto & < ||\varphi_1||, \ldots, ||\varphi_k|| >,
\end{array}
$$

where $\varphi_1, \ldots, \varphi_k$ are the defining formulae of φ.

Below, we will use encodings of formulae to show that there are relations on acceptable structures which are PFP but not IFP-definable. We first fix some notation that will be used in the sequel.

Definition 4.3. *Let $\varphi(\overline{x})$ be a formula with free variables $\overline{x}$, where $\overline{x} := x_{i_1}, \ldots, x_{i_k}$ for some k. The* code *a* of a sequence matches *φ, if $lh(a) \geq \max\{i_j : 1 \leq j \leq k\}$.*

We write $a \models \varphi$, if a matches φ and φ is true in $\mathfrak{A}$ under the variable assignment

$$\beta : x_i \longmapsto \begin{cases} (a)_i & \text{for all } 1 \leq i \leq \mathrm{lh}(x) \\ 0 & \text{otherwise.} \end{cases}$$

If c is the code of φ we also write $a \models c$ for $a \models \varphi$.

We state the following lemma whose proof is technical but not very difficult.

Lemma 4.4. *There is a* PFP*-formula* formula(x) *that is true for all c which are valid codes of* IFP*-formulae.*

4.3 Separating Inflationary and Partial Fixed-Point Logic

In this section we show that partial fixed-point logic is strictly more expressive than inflationary fixed-point logic. The result uses the methods introduced in the sections above.

Definition 4.5. *The relation* $\textsc{Sat}_{\mathrm{IFP}} \subseteq A^2$ *is defined as*

$$\textsc{Sat}_{\mathrm{IFP}} := \{(c, a) : c \text{ is the code of an } \mathrm{IFP}[\tau]\text{-formula } \varphi \text{ and } \varphi \models c\}.$$

Clearly, $\textsc{Sat}_{\mathrm{IFP}}$ is not IFP-definable.

Lemma 4.6. $\textsc{Sat}_{\mathrm{IFP}}$ *is not definable in* IFP*.*

Proof. Suppose, $\textsc{Sat}_{\mathrm{IFP}}$ were definable in IFP. Then the relation $R(x) := \neg Sat(x, < x >)$ would be definable in IFP as well, by a formula $\varphi(x)$ say. Let c be the code of φ. Thus, as φ defines R, for all x, $R(x) \iff Sat(c, < x >)$ but, by definition of R, for all x, $R(x) \iff \neg Sat(x, < x >)$. For $x = c$ we get a contradiction. □

We show now that $\textsc{Sat}_{\mathrm{IFP}}$ is definable in PFP by inductively defining a ternary relation $R(c, i, a) \subseteq A^3$ such that $(c, i, a) \in R$ if, and only if, c is the code of a formula $\varphi \in \mathrm{IFP}[\tau]$ with defining formulae $\varphi_1, \ldots, \varphi_k$, i is an element of $\{1, \ldots, k\}$, and a is the code of a variable assignment matching the free variables in φ such that

$$(\mathfrak{A}, stage(c, 1), \ldots, stage(c, k)), a \models \varphi_i,$$

i.e. φ_i is true under the variable assignment a if all free fixed-point variables R_j are interpreted by the sets $stage(c,j)$ defined as $stage(c,j) := \{\overline{a} : (c,j,a) \in R,$ where a is the code of $\overline{a}\}$.

This relation will be built up by a partial fixed-point induction such that the following invariance property is preserved:

Invariance Property 4.7.

- *For all c,i,a, if $(c,i,a) \in R$ then c is the code of a formula $\varphi \in \mathrm{IFP}[\tau]$, with defining formulae $\varphi_1,\ldots,\varphi_k$, i is an element of $\{1,\ldots,k\}$, and a is the code of a variable assignment matching the free variables in φ such that*

$$(\mathfrak{A}, \mathrm{stage}(c,1),\ldots,\mathrm{stage}(c,k)), a \models \varphi_i,$$

  i.e. φ_i is true under the variable assignment a where all free fixed-point variables R_j are interpreted by the sets $\mathrm{stage}(c,j)$.
- *At each stage α of the induction on R, and all i and c as above, the set* $\mathrm{stage}(c,i)$ *occurs as a stage of the induction on φ_i where all free fixed-point variables R_j of φ_i are interpreted by* $\mathrm{stage}(c,j)$.

Before presenting a formula defining R we introduce some auxiliary formulae *first-order* and *fpr*. The formula *first-order*(R,c,i,a) assumes that the invariance property in 4.7 is satisfied by R. In this case, it defines the set of all (c,i,a) such that $a \models \varphi_i$, under the assumption that all free fixed-point variables R_j are interpreted by $stage(c,j)$ and for all sub-formulae of φ_i of the form $[\mathbf{ifp}_{R_j,\overline{x}_j}\,\varphi_j]$ the fixed point defined by this formula is $stage(c,j)$. Obviously, these assumptions are too optimistic for all i, as the second assumption will generally be true only for some, but not for all i. This formula will be used in a formula defining the relation R described above and there it will be guaranteed that *first-order* will only be "called" for values of i for which both assumptions are satisfied. In the following, we treat variables $t, t_1, \ldots$ as boolean variables, i.e. the only values they can take are 0 and 1, and we use expressions like $t = t_1 \vee t_2$ with the obvious semantics. We also use notation like "$c\widehat{=}\varphi_{c_1} \vee \varphi_{c_2}$" which means that c is the code of a formula $\varphi := \varphi_1 \vee \varphi_2$ and c_1, c_2 are the codes of the sub-formulae.

$$
\begin{aligned}
&\textit{first-order}(c,i,a) := \\
&\quad [\mathbf{pfp}_{Q,c,a,t}\ \text{“}c\widehat{=}\exists x_j \varphi_{c'}\text{”} \wedge ((\exists a'\, Qc'a'1 \wedge \forall i\,((a)_i = (a')_i \vee i = j) \wedge t = 1) \vee \\
&\qquad\qquad (\forall a'\,(\forall i\,((a)_i = (a')_i \vee i = j) \rightarrow Qc'a'0) \wedge t' = 0)) \vee \\
&\qquad\quad \text{“}c\widehat{=}\varphi_{c_1} \vee \varphi_{c_2}\text{”} \wedge (\exists t_1 \exists t_2 (Qc_1at_1 \wedge Qc_2at_2 \wedge t = t_1 \vee t_2) \vee \\
&\qquad\quad \text{“}c\widehat{=}\neg\varphi_{c'}\text{”} \wedge (\exists t'\, Qc'at' \wedge t = \neg t') \vee \\
&\qquad\quad \text{“}c\widehat{=}P_i x_{i_1} \ldots x_{i_k}\text{”} \wedge (t \leftrightarrow P_i(a)_{i_1} \ldots (a)_{i_k}) \vee \\
&\qquad\quad \text{“}c\widehat{=}R_i\overline{x}\text{”} \wedge (t \leftrightarrow Rcia) \vee \\
&\qquad\quad \text{“}c\widehat{=}[\mathbf{ifp}_{R_i,\overline{x}}\,\varphi_i]\text{”} \wedge (t \leftrightarrow Rcia) \\
&\quad](c_i, a, 1)
\end{aligned}
$$

The correctness of the construction is proved in the following lemma.

Lemma 4.8. *Let R be a ternary relation satisfying the invariance property in 4.7. Then for all c,i,a, such that c is the code of a formula φ with defining*

sub-formulae $\varphi_1, \ldots, \varphi_k$ *and* $i \in \{1, \ldots, k\}$,

$$(\mathfrak{A}, R) \models \text{first-order}(c, i, a) \quad \textit{if, and only if,} \quad a \models \varphi_i,$$

where all free fixed-point variables R_j *and all sub-formulae of the form* $[\mathbf{ifp}_{R_j, \overline{x}_j} \varphi_j]$ *are interpreted by the sets* $\text{stage}(R, j)$.

Proof. The lemma is proved by induction on the structure of φ. As this is a standard argument, we do not give the full proof here but refer to [Mos74, Chapter 5] for details. We demonstrate the idea behind the formula by proving the case for existential quantification. Suppose c is the code of a formula $\exists x_j \varphi_{c'}$ and c' is the code of $\varphi_{c'}$. Then "$c \widehat{=} \exists x_j \, \varphi_{c'}$" is satisfied and the formula checks whether there is (the code a' of) a variable assignment satisfying $\varphi_{c'}$, i.e. $(c', a', 1) \in Q$, such that a and a' agree on all variables except x_j. By induction, if there is such an a', then $a' \models \varphi'$ and thus $a \models \varphi$. In this case t is required to be 1. Otherwise, i.e. if there is no such a', $a \not\models \varphi$ and thus $t = 0$.

Note also how the truth of sub-formulae involving fixed points is directly read from the relation R. □

We also need a formula $\mathit{fpr}(R, c, i)$ that is true for c and i if $\mathit{stage}(c, i)$ is the fixed point of the induction on φ_i where all free fixed-point variables R_j of φ_i are interpreted by $\mathit{stage}(c, j)$.

$$\mathit{fpr}(R, c, i) := \forall a(\mathit{first\text{-}order}(R, c, i, a) \rightarrow R(c, i, a)).$$

Clearly, under the same assumptions as in Lemma 4.8, $(\mathfrak{A}, R) \models \mathit{fpr}(c, i)$ if, and only if, $\mathit{stage}(c, i)$ is the fixed-point of φ_i. We are now ready to define the main formula.

$$\begin{aligned}
&\mathit{compute}(c, a) := \\
&\quad [\mathbf{pfp}_{R,c,i,a} \, (\exists l \in \{1, \ldots, \text{lh}(c)\} \, \forall l < j \leq k \; \mathit{fpr}(R, c, j) \wedge \neg \mathit{fpr}(R, c, l) \wedge \\
&\qquad\qquad ((i = l \wedge \mathit{first\text{-}order}(c, i, a)) \vee (i < l \wedge \mathit{Rciat})) \wedge \mathit{formula}(c)) \vee \\
&\qquad\qquad (\forall l \in \{1, \ldots, \text{lh}(c)\} \; \mathit{fpr}(R, c, j)) \wedge \mathit{Rcia} \\
&\quad](c, 1, a).
\end{aligned}$$

The formula $\mathit{formula}(c)$ has been defined in Lemma 4.4 above. Recall the way formulae φ are coded by $c := < ||\varphi_1||, \ldots, ||\varphi_k|| >$. The formula *compute* first defines the unique l such that the fixed points of all formulae φ_j with $j > l$ are already computed in R but the induction on φ_l has not yet reached its fixed point. For this l, the formula $\mathit{first\text{-}order}(c, l, a)$ is evaluated, i.e the next stage of the induction on φ_j is computed. Further, all triples (c, j, a) such that $j < l$ are kept in R, i.e. the current stages of the induction on φ_j with $j < l$ are left untouched. On the other hand, all triples (c, j, a) for $j > l$ are removed from R, i.e. the fixed-point induction on the formulae φ_j, which might depend on R_l, are set back to the empty set.

Thus, in the end there will be no such l as all fixed points are already computed. In this case the relation R is left untouched and thus the fixed point of *compute* has been reached. This proves the following lemma.

Lemma 4.9. SAT_{IFP} *is definable in* PFP.

The proof of the following theorem and its corollary is now immediate.

Theorem 4.10. PFP *is more expressive than* IFP *on acceptable structures.*

Corollary 4.11. PFP *is more expressive than* IFP *on all structures in which an acceptable structure is* PFP*-interpretable.*

Among the structures in which an acceptable structure is PFP-interpretable are $(\omega, <)$ and $(\mathbb{R}, <, +)$ and all expansions of it, e.g. the ordered field of reals. Examples of structures not interpretable in an acceptable structure are structures over the empty signature or a signature containing constant symbols only, but also the real line $(\mathbb{R}, <)$.

References

[AV91a] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43:62–124, 1991.

[AV91b] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proc. of the 23rd ACM Symp. on the Theory of Computing*, 1991.

[AVV97] S. Abiteboul, M. Vardi, and V. Vianu. Fixpoint logics, relational machines, and computational complexity. *Journal of the ACM*, 44(1):30–56, 1997. An extended abstract appeared in the Proc. 7th IEEE Symp. on Structure in Complexity Theory, 1992.

[Daw93] A. Dawar. *Feasible Computation Through Model Theory.* PhD thesis, University of Pennsylvania, 1993.

[DG02] A. Dawar and Y. Gurevich. Fixed-point logics. *Bulletin of Symbolic Logic*, 8(1):65–88, 2002.

[DK] A. Dawar and S. Kreutzer. Partial and Alternating Fixed Points in Modal Logic. Unpublished.

[EF99] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer, 2nd edition, 1999.

[GS86] Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.

[Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986. Extended abstract in Proc. 14th ACML Symp. on Theory of Computing, pages 147-152, 1982.

[Kre02] S. Kreutzer. Expressive equivalence of least and inflationary fixed-point logic. Proc. of the 17th Symp. on Logic in Computer Science (LICS), 2002.

[Mos74] Y.N. Moschovakis. *Elementary Induction on Abstract Structures.* North Holland, 1974. ISBN 0 7204 2280 9.

[Var82] M. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on the Theory of Computing*, pages 137–146, 1982.