Fixed Point Logics, Generalized Quantifiers, and Oracles

Henrik Imhof, Mathematisches Institut, Abteilung für Mathematische Logik, Universität Freiburg, Albertstr. 23b, D-79104 Freiburg, Germany
E-mail: imhof@sun1.mathematik.uni-freiburg.de
Keywords: Finite Model Theory, Complexity Theory, Generalized Quantifiers, Oracles

Abstract

The monotone circuit problem Q_{MC} is shown to be complete for fixed point logic IFP under quantifier free reductions. Enhancing the circuits with an oracle Q leads to a problem complete for $IFP(Q)$. By contrast, if $\mathcal{L}$ is any extension of FO with generalized quantifiers, one can always find a Q such that $IFP(Q)$ is not contained in $\mathcal{L}(Q)$. For $\mathcal{L} = FO(Q_{MC})$, we have $\mathcal{L} \equiv IFP$ but $\mathcal{L}(Q) < IFP(Q)$. The adjunction of Q reveals the difference between these two representations of the class of IFP–definable queries. Also for partial fixed point logic, PFP a complete problem based on circuits is given, and, concerning the adjunction of further quantifiers, similar results as for IFP are proved.

On ordered structures, where our results still hold, this reads as follows: For any given oracle Q, the complexity class $PTIME^Q$ (or $PSPACE^Q$ in a bounded oracle model) can be characterized by an extension of FO with a uniform sequence of quantifiers. However, there is no such logic $\mathcal{L}$ that satisfies $\mathcal{L}(Q) \equiv PTIME^Q$ (or $PSPACE^Q$) for all Q.

We also study second order logic. Each level Σ_i of the polynomial hierarchy has a complete circuit problem. Closing FO under this problem does not exceed Σ_{i+1}. Hence, the polynomial hierarchy collapses to a certain level if and only if there is a class Q such that $SO \equiv FO(Q)$ holds on finite structures.

1 Introduction

In recent years, extensions of first order logic with uniform sequences of Lindström quantifiers have become of great interest in finite model theory. As Dawar has shown, these logics of the form $FO(Q)$, generated by a class of structures Q and all its vectorizations, are appropriate to capture a great variety of classes of recursive queries [3]. In the following, we will refer to these logics $FO(Q)$ as *Lindström logics*. $PTIME$, for instance, has either such a representation, or no reasonable logical characterization at all.

Another method of increasing the expressive power of logics is to allow for certain recursion mechanisms, a prominent example being fixed point logic, IFP. It has turned out that the quantifier based approach is capable of replacing these mechanisms, i.e. one can find a class Q_{IFP} such that $FO(Q_{IFP}) \equiv IFP$, a fact that has been established independently in [9, 11, 2].[1]

[1] Here, the notation $\mathcal{L}_1 \equiv \mathcal{L}_2$ means that these logics define the same class of queries, i.e. each $\varphi_1 \in \mathcal{L}_1$, that may have free variables, is equivalent to a $\varphi_2 \in \mathcal{L}_2$ and vice versa. We will also denote the equivalence of complexity classes with logics in this way. To indicate only one inclusion we use the notation $\mathcal{L}_1 \leq \mathcal{L}_2$.

This raises the following questions:

1. Is there a natural class Q_{IFP} with $FO(Q_{IFP}) \equiv IFP$, say one of the problems well known to be $PTIME$ complete under $LOGSPACE$ reductions?

2. How about extensions $IFP(Q)$ of fixed point logics: are they all equivalent to logics of the form $FO(\hat{Q})$ for some $\hat{Q}$?

3. Does the equivalence $FO(Q_{IFP}) \equiv IFP$ relativize to $FO(Q_{IFP}, Q) \equiv IFP(Q)$ for arbitrary Q?

Questions 2 and 3 are important in the context of oracle complexity classes: On ordered structures, $IFP(Q)$ captures $PTIME^Q$, the set of queries computable in polynomial time with access to the oracle Q under a canonical encoding of structures as strings [7, 20]. The analogue holds for PFP and $PSPACE$ if the oracle tape is subject to the polynomial space bound.

So, one could ask briefly: Is there a Lindström logic for oracle complexity classes?

It was already known that there can be no translation between $IFP(Q)$ and $FO(Q_{IFP}, Q)$ independent of Q (for a fixed signature of Q, say) [17], but that does not rule out the equivalence of these logics.

Applying circuits as acceptors for first order formulas, we can answer the above questions. We state the results for IFP.

(1) The monotone circuit problem Q_{MC} is complete for IFP under quantifier free first order reductions.

(2) Enhancing the circuits with extra nodes for a class Q in a canonical way yields a quantifier $\hat{Q}$ for $IFP(Q)$. This generalizes a result of Hella [11] concerning extensions of LFP with *monotone* quantifiers Q.

(3) Applying a diagonal argument to the enhanced circuits results in a class Q with $FO(Q_{IFP}, Q) \not\equiv IFP(Q)$.

Hence, there is no Lindström logic for IFP (or $PTIME$) allowing for the adjunction of quantifiers in the same way as oracles are added to $PTIME$; more precisely, there is no Q' such that $PTIME^Q \leq FO(Q', Q)$ holds on ordered structures for every Q. Nevertheless, for each *fixed* oracle Q there is a Lindström logic for $PTIME^Q$.

From the completeness of $\hat{Q}$ for $IFP(Q)$ (see (2) above), we will derive a normal form for this logic where no nested quantifiers Q occur. The latter was known only for ordered structures [7]. Actually, the construction for (3) depends mainly on the fixed nesting of quantifiers Q in $FO(Q_{IFP}, Q)$ as opposed to a hidden polynomial nesting in $IFP(Q)$. Thus, (2) and (3) can be regarded as counterparts of each other. The answer (3) will be an extension of the mentioned result in [17] stating that there is no *uniform* correspondence between $PTIME^Q$ and extensions of any Lindström logic with Q.

Since circuit quantifiers can also be found for the levels of the polynomial hierarchy, we can give some criteria for second order logic to be representable by a uniform sequence of quantifiers (to this end, the normal form for $IFP(Q)$ will be useful).

2 Basic Definitions

All structures throughout this paper are finite but have at least two elements (this will provide us with nontrivial equality types; however, it is not hard to adapt the constructions to include the singleton case). Q will always denote a class of structures over a finite relational vocabulary $\sigma = \{R_1, \ldots, R_s\}$. Regarding Q as a quantifier allows for extending logics in a very natural way [15]. The following definition not only adds Q to a logic but also a uniform sequence of quantifiers generated by this class, which has proved useful in descriptive complexity theory.

Definition 1: Let $\mathcal{L}$ be a logic the syntax of which is given by a collection of certain formation rules (involving first order variables $x_1, x_2, \ldots$). Then, for each vocabulary τ, $\mathcal{L}(Q)[\tau]$ is the smallest set of formulas that is closed under the formation rules for $\mathcal{L}$ enhanced, for each $r \geq 1$, with the following clause:

If $\overline{x}_i$ is an rr_i–tuple of variables $(1 \leq i \leq s)$, $\psi_i(\overline{x}_i) \in \mathcal{L}(Q)[\tau]$, $\overline{x}$ an r–tuple, $\psi(\overline{x}) \in \mathcal{L}(Q)[\tau]$, then

$$Q\overline{x}, \overline{x}_1, \ldots, \overline{x}_s; \psi, \psi_1, \ldots, \psi_s$$

is again a formula in $\mathcal{L}(Q)[\tau]$. It is true in a τ-structure $\mathcal{A}$ if and only if $\psi^{\mathcal{A}} := \{\overline{a} \in A^r \mid \mathcal{A} \models \psi[\overline{a}]\}$ is not empty and the σ–structure

$$(\psi^{\mathcal{A}}; \psi_1^{\mathcal{A}} \cap (\psi^{\mathcal{A}})^{r_1}, \ldots, \psi_s^{\mathcal{A}} \cap (\psi^{\mathcal{A}})^{r_s})$$

belongs to Q. Here, for $1 \leq i \leq s$, $\psi_i^{\mathcal{A}}$ denotes the relation $\{\overline{a} \in A^{rr_i} \mid \mathcal{A} \models \psi_i[\overline{a}]\}$. This set is viewed as a set of r_i–tuples over A^r rather than rr_i–tuples over A. Thus, indeed, we have defined a σ–structure above.

Note that in the above formulas ψ, ψ_i there can occur some more free variables as parameters. We write $\psi(\overline{x})$ etc., only to indicate the variables to be bound by Q.

Extending the usual definition, a variable v is said to be free in $Q\overline{x}, \overline{x}_1, \ldots, \overline{x}_s; \psi, \psi_1, \ldots, \psi_s$ if it is free in any of the formulas ψ, ψ_i but is not amongst $\overline{x}, \overline{x}_1, \ldots, \overline{x}_s$.

If ψ, the relativization of the domain, is omitted in the above formula, then it is assumed to be true for all r–tuples, i.e. the domain will be A^r.

□

If $\Phi = (\psi, \psi_1, \ldots, \psi_s)$ consists only of quantifier free formulas, then it is called a *quantifier free reduction* (of arity r), and the above structure is denoted by $\mathcal{A}^\Phi$. As for *many one reductions* in complexity theory, one can prove completeness results under this notion of reduction, which we will do below.

As an example, let $\sigma = \{E, S, T\}$ with unary S, T, and binary E, and take for Q those digraphs $\mathcal{B}$ that have two nodes u, v, labelled by the relations S and

T respectively, such that there is a path from u to v. Then, Q can be defined in *Transitive Closure Logic*, TC, namely by the formula

$$\varphi = \exists uv(Su \wedge Tv \wedge [TC_{x,y}Exy]uv).$$

On the other hand, each formula $[TC_{\overline{x},\overline{y}}\chi]\overline{uv}$ is equivalent to the $FO(Q)$–formula

$$Q\overline{x},\overline{y};\overline{z};\overline{w}\ \ \chi;\overline{z}=\overline{u};\overline{w}=\overline{v}.$$

Here, vectorization was used to assert connectivity of a graph the vertices of which are tuples rather than single elements.

In the sense of Lindström [15], we have added to $\mathcal{L}$ not only the class Q itself but also the following classes $Q^{r,P}$, which are derived from Q by the simultaneously applied processes of vectorization and relativization:

Definition 2: Given $r \geq 1$, relation symbols S_i $(1 \leq i \leq s)$ of arity $\mathrm{ar}(S_i) = rr_i$, and P of arity r, let

$$Q^{r,P} := \{(A; P, S_1, \ldots, S_s) \mid (P; S_1 \cap P^{r_1}, \ldots, S_s \cap P^{r_s}) \in Q\} \qquad \square$$

Enriching first order logic, FO, by one or several classes Q according to Definition 1 leads, in a sense, to a minimal logic making the classes $Q^{r,P}$ expressible. To make this statement more precise, we consider regular logics $\mathcal{L}$ (for a thorough discussion see [4]), which can be regarded as classes of queries that are closed under first order operations and under substitutions of the form $R/\lambda\overline{x}\varphi(\overline{x})$ (with $\overline{x}$ and R of the same arity). That is, in any $\mathcal{L}$–formula involving a relation symbol R, one can replace R by a relation which is given by any other $\mathcal{L}$–formula φ.

Lemma 3 (Minimality Lemma): Let $\mathcal{L}$ be regular with $Q^{r,P} \in \mathcal{L}$ for all $r \geq 1$. Then $FO(Q) \leq \mathcal{L}$. Also, if several quantifiers are adjoint to FO, this leads to a minimal logic in this sense.

Proof: An easy induction on the calculus for $FO(Q)$. At the Q-step, take an $\mathcal{L}$–formula defining $Q^{r,P}$ and substitute $P, S_1, \ldots, S_s$ by their definitions $\psi, \psi_1, \ldots, \psi_s$, more precisely by $\mathcal{L}$–translations of these formulas. $\square$

It should be remarked that relativization of the domain (the formula ψ in Definition 1) becomes important only in Section 5 for the description of oracle classes. As we will prove, relativization cannot be dropped there. This is not surprising, as, for instance, Turing machines can form oracle queries of a size that does not necessarily equal the size of the input.

Next, we recall the **definition of fixed point logic:** *Inductive fixed point logic*, IFP allows for second order variables $X, Y, \ldots$, which may only occur bounded by fixed point operators. The meaning of $[IFP_{\overline{x},X}\theta(\overline{x}, X)]\overline{t}$ is $\overline{t} \in X_\infty$, where we set $X_0 := \emptyset$, $X_{j+1} := X_j \cup \{\overline{x} | \theta(\overline{x}, X_j)\}$, $X_\infty := \bigcup_{j\geq 0} X_j$. Note that θ itself may contain other variables apart from those emphasized above and that fixed point operators may be nested. However, in an $IFP[\tau]$–formula all relation variables, i.e. all relation symbols that are not in τ, must be bound by fixed point operators.

Partial fixed point logic, *PFP* is defined similarly, setting however $X_{j+1} := \{\overline{x} | \theta(\overline{x}, X_j)\}$, and letting $X_\infty := X_j$ for the smallest j such that $X_j = X_{j+1}$ if such a j exists, and $X_\infty := \emptyset$ otherwise.

Later on, we will make use of a normal form theorem saying that one can dispense with nesting of fixed point operators. For convenience, we state it in a version presented in [8]. A somewhat weaker normal form had already been given by Immerman [12], and would suffice for our purposes as well (the proofs getting slightly more complicated).

Fact 4: Each *IFP*–formula is equivalent to one of the form $\varphi = \exists z[IFP_{\overline{x},X}\chi(\overline{x}, X)]z \ldots z$, where χ is a first order formula. The corresponding normal form holds for *PFP*.

Both, inductive and partial fixed point logic play an eminent role in descriptive complexity theory, as, on ordered structures, they capture *PTIME* and *PSPACE* respectively. For a detailed exposition, the reader is referred to [5]. Occasionally, we will also make use of *least fixed point logic*, *LFP*, which is obtained from *IFP* by restricting the application of fixed point operators to the case of formulas that are positive in the relation variable in question. This fragment equals *IFP* in expressive power [10], and the normal form of Fact 4 can be adapted to this logic, i.e. χ can be chosen positive in X.

Recently, it has turned out that these results can be extended to oracle complexity classes, which are based on Turing machines with an additional tape and a designated oracle query state. Once the machine is in this state it checks in one step whether the content of that tape belongs to a certain class (the oracle) or not, and erases the oracle tape afterwards. Identifying quantifiers (i.e. classes of structures) and oracles via a canonical encoding of structures as strings, one gets the following.

On ordered structures, $IFP(Q)$ captures $PTIME^Q$, the set of queries computable in polynomial time with access to the oracle Q [7, 20]. The analogue holds for *PFP* and *PSPACE* if the oracle tape is also subject to the polynomial space bound. The second statement can be proved by the technique used in [20]. We omit the details, because, in order to investigate $PSPACE^Q$, we only make use of the fact $PTIME^Q \leq PSPACE^Q$, which holds in any reasonable oracle model.

Given two classes Q_i $(i = 1, 2)$ with $FO(Q_i) \equiv IFP$, and a third class Q, the minimality lemma yields $FO(Q_1, Q) \equiv FO(Q_2, Q)$. So, it will make no difference to which of them we compare $IFP(Q)$, see Section 5. This fact is also obtained from the following remark, which sheds a first light on the role of second order variables in this context. It is stated for *IFP* but also holds for *PFP*.

Remark 5: Let Q_{IFP} be a class of structures such that $IFP \equiv FO(Q_{IFP})$ holds. Then for any class Q we have $FO(Q_{IFP}, Q) \equiv IFP(Q)^*$, where $IFP(Q)^*$ denotes the fragment of $IFP(Q)$ which is obtained by allowing the formation of $Q\overline{x}, \overline{x}_1, \ldots, \overline{x}_s; \psi, \psi_1, \ldots, \psi_s$ only in case $\psi, \psi_1, \ldots, \psi_s$ have no *free* occurrences of *second order* variables.

Proof: $FO(Q_{IFP}, Q) \leq IFP(Q)^*$ is a consequence of the minimality lemma. To see the regularity of $IFP(Q)^*$, note that substitutions $R/\lambda\overline{x}\varphi(\overline{x})$ are only permitted for formulas φ without free second order variables, as $IFP(Q)$ and hence $IFP(Q)^*$ consists only of such formulas.

To obtain the other inclusion, proceed by induction over $IFP(Q)^*$, this time treating free second order variables as usual relation symbols. For the IFP–clause, let $\varphi = [IFP_{\overline{x},X}\chi]\overline{t}$ be an $IFP(Q)^*$–formula. We can assume that $\chi \in FO(Q_{IFP}, Q)$ and that no occurrence of X in χ is in the scope of a quantifier Q (note that the latter property remains unchanged in each induction step). For simplicity, assume that $\sigma_Q = \{P\}$ and ignore domain relativization. For each subformula $\theta = Q\overline{x}\psi_P(\overline{x}, \overline{z})$ of χ that is not in the scope of a quantifier Q, let S_θ be a new relation symbol of the same arity as $\overline{z}$ (for different occurrences of the same θ take different symbols). Since X does not occur in the formulas ψ_P, we can make the following substitutions: Replace each of the above θ in χ by $S_\theta\overline{z}$ thus obtaining a $\chi' \in FO(Q_{IFP})$. Clearly, $[IFP_{\overline{x},X}\chi']\overline{t}$ is equivalent to a $\varphi' \in FO(Q_{IFP})$. Replacing in turn each S_θ in φ' by θ yields an $FO(Q_{IFP}, Q)$–formula equivalent to φ. □

Finally, we need *circuits* in order to evaluate formulas. On the one hand, they will provide natural quantifiers for fixed point logics and their extensions. On the other hand, they will be used in a diagonal argument in Section 5. We concentrate on the case where no negations occur, i.e. on so called *monotone* circuits.

Definition 6: Let $\sigma_{MC} = \{O, E, K, D, I\}$, where O is unary (output), E binary (Eab means node a is connected as an input to node b), K and D unary (conjunctions and disjunctions), and I unary (inputs with value TRUE). Q_{MC} is the class of monotone circuits that produce output TRUE in at least one output node, that is

$$Q_{MC} := \mathrm{Mod}(\exists w(Ow \wedge \varphi_{TRUE}(w))), \text{ where}$$

$$\varphi_{TRUE} := [IFP_{x,Y}(Ix \vee (\exists y Eyx \wedge (\varphi_{CON} \vee \varphi_{DIS})))]w,$$

$$\varphi_{CON} := Kx \wedge \forall y(Eyx \rightarrow Yy),\ \varphi_{DIS} := Dx \wedge \exists y(Eyx \wedge Yy) \qquad \square$$

Note that we did not make any restrictions on the underlying graph $(A; E)$ of a circuit $(A; O, E, \ldots)$. By monotonicity, nodes with value FALSE and not yet evaluated ones can be treated in the same way (and do not require an extra symbol in σ_{MC}). Compared to the usual version of the monotone circuit value problem, where circuits must be acyclic and of fan-in 2, our definition is slightly more liberal. This will enable us to encode fixed point processes.

The situation changes drastically if negation nodes, that flip their input bit and are labelled by a new symbol N, are admitted: for arbitrary graphs, it is not even clear how to evaluate the circuits, as cycles may occur. In the case of trees, however, one can compute the truth values level by level keeping track of the classes of true, false, and not evaluated nodes. Let u, v be variables and consider interpretations of $u \neq v$. For a binary Y the sets $\{z \mid Y_j uz\}$ and

$\{z \mid Y_j vz\}$ will contain the nodes z with value TRUE respectively FALSE after j steps of the evaluation. Set

$$\varphi_{EVAL} := \exists y Eyx \wedge \forall y(Eyx \rightarrow (Yuy \vee Yvy)),$$

$$\varphi_{CON+} := Kx \wedge \forall y(Eyx \rightarrow Yuy),\ \varphi_{CON-} := Kx \wedge \exists y(Eyx \wedge Yvy),$$

$$\varphi_{DIS+} := Dx \wedge \exists y(Eyx \wedge Yuy),\ \varphi_{DIS-} := Dx \wedge \forall y(Eyx \rightarrow Yvy),$$

$$\varphi_{NEG+} := Nx \wedge \exists y(Eyx \wedge Yvy),\ \varphi_{NEG-} := Nx \wedge \exists y(Eyx \wedge Yuy),$$

$$\begin{aligned}\varphi^*_{TRUE}(u,v,w) := [IFP_{z,x,Y}(&z = u \wedge Ix) \vee \\ &(z = v \wedge \neg\exists y Eyx \wedge \neg Ix) \vee \\ &(z = u \wedge \varphi_{EVAL} \wedge (\varphi_{CON+} \vee \varphi_{DIS+} \vee \varphi_{NEG+})) \vee \\ &(z = v \wedge \varphi_{EVAL} \wedge (\varphi_{CON-} \vee \varphi_{DIS-} \vee \varphi_{NEG-}))]uw.\end{aligned}$$

In a circuit whose underlying graph is a tree with root w, the formula φ^*_{TRUE} says that w is evaluated to TRUE. Since we can define the root and quantify over u and v, we get:

Remark 7: The class of satisfied circuits is still *IFP*–definable if negation nodes are allowed provided that one restricts to the case where the underlying graph is a tree (or a collection of trees). □

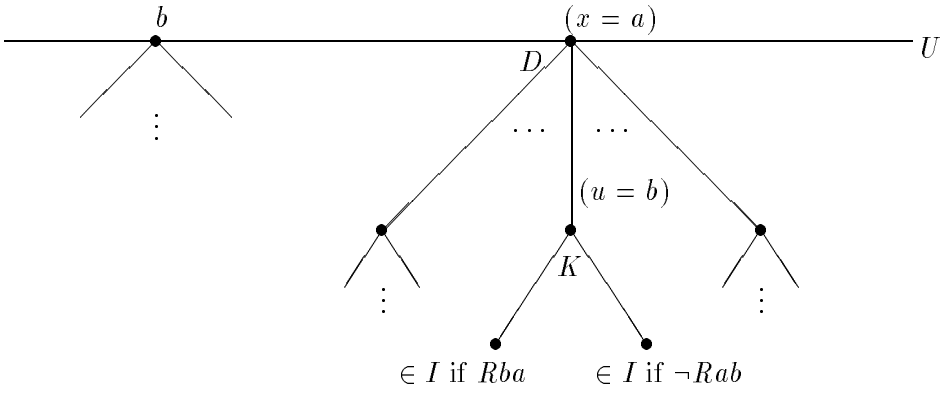
3 Circuit Problems and Fixed Point Logics

It is a well known technique to evaluate first order formulas by circuits. Constructing complete problems in this manner was first done by Lovász and Gács [16] for *NP*. Here we use circuits to deal uniformly with the first order part χ of formulas of the form $[IFP_{x,X}\chi]t$. Before going into details we sketch the main ideas:

Let $\mathcal{A}$ be a τ–structure and $\chi(x) \in FO[\tau]$ be in prenex normal form with the quantifier free part in disjunctive normal form. Expanding the leading quantifiers in χ into disjunctions and conjunctions over the elements of $\mathcal{A}$, one can pass to a quantifier free formula that, in turn, can be viewed as a circuit. The following structure $\mathcal{E} = \mathcal{E}(\chi, \mathcal{A})$ checks simultaneously for each $a \in A$ whether $\mathcal{A} \models \chi[a]$ holds:

Let $\mathcal{E}$ contain a copy U of the set A; each $a \in U$ is (in $\mathcal{E}$) the root of a circuit $\mathcal{C}_a$. This circuit is a tree with labels K, D and I, where I denotes leaves with input TRUE. $\mathcal{C}_a$ is to evaluate χ for $x = a$, whence its tree structure is derived from the syntactical structure of χ. The truth values are computed as in Definition 6, so leaves not in I are interpreted as inputs with value FALSE. Existential quantifiers $\exists u$ in χ are simulated by disjunction nodes in $\mathcal{C}_a$ having a subcircuit for each value $b \in A$ of the variable u. Universal quantifiers can be expanded into conjunctions. The disjunctions and conjunctions in the quantifier free part are evaluated in the obvious way, and the leaves of $\mathcal{C}_a$ will correspond to atomic or negated atomic formulas. Following the path from such a leaf λ to the root a of $\mathcal{C}_a$, one gets an assignment $\beta_\lambda : u_i \mapsto a_i$ for the occurring variables $u_1, \ldots, u_m$ (including $x \mapsto a$ if x is on that list). The leaf λ is being labeled

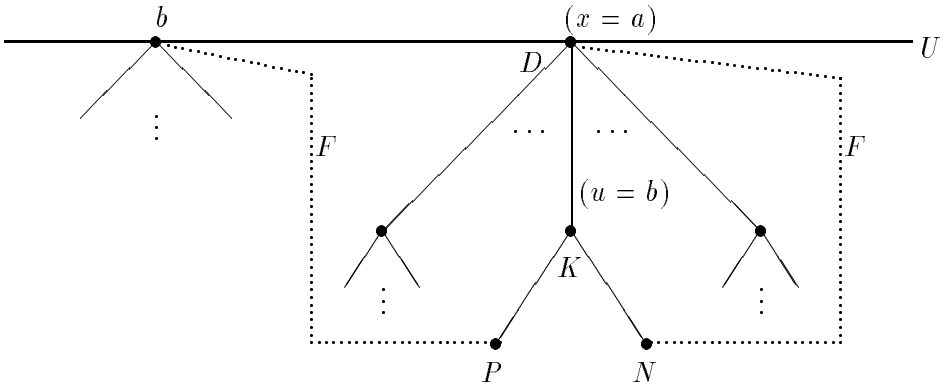
with I if and only if the corresponding atomic formula is true in $(\mathcal{A}, \beta_\lambda)$. For instance, in case $\chi = \exists u(Rux \wedge \neg Rxu)$, $\mathcal{E}$ looks as follows:



Now assume that a monadic second order variable X occurs in χ. We want to compute the fixed point $[IFP_{x,X}\chi]$ (or $[PFP_{x,X}\chi]$, which will make no difference for the construction). One might attempt to iterate the above construction by expanding the fixed point operation into a disjunction over the stages of the fixed point. However, we wish to be able to interprete the circuits in an arbitrary class of structures, below. Hence, we would have the problem of defining levels of a circuit corresponding to the stages. This seems to be impossible in structures that are not necessarily ordered.

Instead, we will make use of the copy of the domain, U, where we can store the current stage of the computation. In principle, we can define the $\mathcal{C}_a$ as before in order to evaluate one step of the iteration, facing difficulties only for those leaves corresponding to a formula Xy or $\neg Xy$, where y is a variable which has been assigned a certain value b via the path leading to the root in U.

We identify the levels $X_j \subset A$ with their copies in U. If $\mathcal{E} = \mathcal{E}(\chi, \mathcal{A})$ is to determine $X_{j+1} = \{a \in U \mid \mathcal{A} \models \chi[a, X_j]\}$, the leaves mentioned above must know first, whether they correspond to Xy or its negation, and second, to which b the variable y refers. Therefore we introduce unary relations P and N (positive and negative evaluation) and a binary relation F connecting the leaves with the values $b \in U$ (note that the use of N is restricted to the leaves here, and that the circuits in $\mathcal{E}$ are still monotone in contrast to Remark 7). As an example, let $\chi = \exists u(Xu \wedge \neg Xx)$. Then $\mathcal{E}(\chi, \mathcal{A})$ looks as follows:



Observe that the formula χ has been coded into $\mathcal{E}(\chi, \mathcal{A})$ and, as the evaluation of circuits is itself *IFP*–definable, we can characterize the copy of X_∞ in U by an *IFP*–formula not depending on χ. This independence is the crucial point in the construction of complete problems.

It remains to find a quantifier free reduction Φ_χ for each $\chi \in FO[\tau]$ such that $\mathcal{A}^{\Phi_\chi} \simeq \mathcal{E}(\chi, \mathcal{A})$ holds for each τ–structure $\mathcal{A}$. Then, making use of the *IFP*– normal form theorem (Fact 4), we can transfer the question of whether an *IFP*– formula is satisfied in a structure $\mathcal{A}$ to the question of whether $\mathcal{E}(\chi, \mathcal{A})$ is a satisfied circuit in a sense that has to be made precise.

We will obtain Φ_χ via equality types for the different isomorphism types of nodes. In the above example, let $r = 5$ be the arity of Φ_χ. The elements of $\mathcal{A}^{\Phi_\chi}$ are tuples $(a_1, \ldots, a_5)$, where the equality type of (a_1, a_2, a_3) encodes the isomorphism type of the node. There are four such types corresponding to the four subformulas of χ. The pair (a_4, a_5) can be used to specify the different trees in $\mathcal{E}$ and their branches. For a moment, fix $a_1 \neq a_2$ in A, and, for each $a \in A$, take the tuple $(a_1, a_1, a_1; a, a_1)$ for the root of $\mathcal{C}_a$ in $\mathcal{E}$. For each $b \in A$ it has the predecessor $(a_1, a_1, a_2; a, b)$, which has in turn the predecessors $(a_1, a_2, a_2; a, b)$ and $(a_1, a_2, a_1; a, b)$. Hence, the arity r of Φ_χ will be determined mainly by the quantifier rank of χ. There are two technical difficulties:

- The arity of the fixed point variable X might be $s > 1$.
- If in $\overset{r}{a} = (\overline{c}, \overline{d})$ the equality type of $\overline{c}$ codes the isomorphism type of this node and $\overline{d}$ represents the path, then there are many other tuples $\overline{c}'$ of the same equality type as $\overline{c}$, and one cannot distinguish between them. In fact, we cheated a little bit in the above example, where we fixed two elements and arranged them to tuples.

The first problem can be solved easily by taking for U a copy of A^s (but leaving the arity of the symbols U, P, N and F fixed). To overcome the second difficulty, let $\mathcal{A}^{\Phi_\chi}$ consist of a collection of isomorphic copies of $\mathcal{E}(\chi, \mathcal{A})$. Each of them will work in exactly the same way, whence our above argumentation will go through. In a final step, we will eliminate the symbols U, P, N and F and get the desired natural problem complete for *IFP*.

Theorem 8: Q_{MC} is complete for *IFP* with respect to quantifier free reductions, i.e. every $\varphi \in IFP$ is equivalent to a formula

$$Q_{MC}\overline{x}, \overline{y}_1, \overline{y}_2, \overline{z}, \overline{v}, \overline{w}\ \Phi,$$

where $\Phi = (\psi_O(\overline{x}), \psi_E(\overline{y}_1, \overline{y}_2), \psi_K(\overline{z}), \psi_D(\overline{v}), \psi_I(\overline{w}))$ is a tuple of quantifier free first order formulas.
In particular, $IFP \equiv FO(Q_{MC})$.

First, we modify the circuit problems according to our needs along the lines sketched above. Later we will come back to the original Q_{MC}. This modification will be a bit technical, but the construction will be a common basis for the monotone circuit problem as well as for a complete problem for *PFP*, and for the quantifiers of the next section.

Let τ be some vocabulary, $\chi \in FO[\tau]$, $\mathcal{A}$ a τ–structure and $\beta : V \to A$ an assignment, where V is a set of variables including those free in χ. Assume that all negations in χ have been pulled in front of atomic formulas. As pointed out above, there is a canonical way to define a monotone circuit $\mathcal{C}(\chi, \mathcal{A}, \beta)$ which is to evaluate χ on $(\mathcal{A}, \beta)$. To make this precise, we give an inductive definition:

If χ is atomic or negated atomic, then $\mathcal{C}(\chi, \mathcal{A}, \beta)$ is a single node w being both input and output, i.e. $O = \{w\}, E = K = \emptyset, I = \{w\}$ if $(\mathcal{A}, \beta) \models \chi$, $I = \emptyset$ otherwise. If $\chi = \exists x \chi'$ then let $\mathcal{C}(\chi, \mathcal{A}, \beta)$ be a tree with a single root $w \in O$, which is a disjunction and has the root of a subtree $\mathcal{C}(\chi, \mathcal{A}, \beta\frac{a}{x})$ as an E–predecessor for each $a \in A$ (clearly, these roots are no longer in O). Similarly, universal quantifiers, disjunctions and conjunctions can be evaluated.

Now, let X be a second order variable of arity s, and $\chi \in FO[\tau \cup \{X\}]$. For $M \subset A^s$, the following generalized circuit $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ checks whether $(\mathcal{A}, \beta) \models \chi[M]$ holds:

Let $\hat{\sigma}_{MC} = \sigma_{MC} \cup \{U, P, N, F\}$ for unary relation symbols U, P, N and a binary relation symbol F. $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ consists of a copy of the set A^s, marked as the U–part, and, disjoint from it, of a circuit which can be constructed inductively like $\mathcal{C}(\chi, \mathcal{A}, \beta)$: The only new steps to be considered are $\chi = X\overline{z}$ and $\chi = \neg X\overline{z}$ for some variables $\overline{z}$. In the first case, we choose a single node λ and let $\lambda \in P$, in the second case $\lambda \in N$ (all other relations are empty, apart from O). Having constructed these trees, we have to define F. Let $\lambda \in P \cup N$ be a leaf that corresponds to a formula $X\overline{z}$ or $\neg X\overline{z}$. We define a tuple $\overline{b} \in A^s$: If the occurrence of z_ν is free in χ, we set $b_\nu := \beta(z_\nu)$. Otherwise, it is bound, say by $\forall z_\nu$, and λ occurs in a subtree evaluating a subformula $\forall z_\nu \theta$ of χ for the case $z_\nu = b$ for some $b \in A$. Then we let $b_\nu := b$. So, $\overline{b}$ is the interpretation of $\overline{z}$ taken from the path which starts at λ and leads to the root. We take the pair $(\lambda, \overline{b})$ into F (note that in $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ the tuple $\overline{b}$ is a single element). As in $\mathcal{C}(\chi, \mathcal{A}, \beta)$, O holds only for the root of the tree. Now, it is easy to prove by induction on χ:

Lemma 9: Let $\chi, \mathcal{A}, \beta, M$ and $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ be as above and let

$$\hat{\varphi}_{TRUE}(w, X) :=$$

$$[IFP_{x,Y}\left(Ix \vee \left(Px \wedge \exists z(Fxz \wedge Xz)\right) \vee \left(Nx \wedge \exists z(Fxz \wedge \neg Xz)\right) \vee \left(\exists y Eyx \wedge (\varphi_{CON} \vee \varphi_{DIS})\right)\right)]w.$$

Then we have $(\mathcal{A}, \beta) \models \chi[M]$ iff $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta) \models \exists w(Ow \wedge \hat{\varphi}_{TRUE}(w))[M]$. □

Being able to evaluate first order formulas with a free second order variable, we can put $|A^s|$ many of these circuits together and evaluate a fixed point operator:

Definition 10: Given a τ–structure $\mathcal{A}$, an assignment β, a formula $\chi \in FO[\tau \cup \{X\}]$, where X is s–ary, an s–tuple $\overline{x}$ of variables and $\overline{a} \in \mathcal{A}^s$, let $\hat{\mathcal{C}}_{\overline{a}}$ denote the structure $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta\frac{\overline{a}}{\overline{x}})$. The $\hat{\sigma}_{MC}$–structure $\mathcal{E} = \mathcal{E}(\chi, \mathcal{A}, \beta)$, called a *generalized circuit*, is obtained as follows: The U–part $U^{\mathcal{E}}$ of $\mathcal{E}$ is a copy of the set A^s. For each $\overline{a} \in A^s$ we have a copy of $\hat{\mathcal{C}}_{\overline{a}}$, but all of

them use $U^{\mathcal{E}}$ as their U–part. Moreover, the root of $\hat{\mathcal{C}}_{\overline{a}}$ is now the element $\overline{a}$ ($\in U$), and O is the diagonal in U, i.e. those $\overline{a}$ with $a_1 = \ldots = a_s$. $\square$

Note that we have already seen an example of a generalized circuit above, namely the structure $\mathcal{E}(\chi, \mathcal{A})$ for $\chi = \exists u(Xu \wedge \neg Xx)$. Generalized circuits provide us with a method of constructing complete problems for various logics. We exhibit such problems for fixed point logic, and will give further applications in Sections 4 and 6.

Theorem 11: Let $\hat{Q}_{MC} := Mod(\exists z(Oz \wedge [IFP_{w,X}(Uw \wedge \hat{\varphi}_{TRUE})]z))$.

(i) For $\mathcal{A}, \chi, \beta$ as above we have

$$(\mathcal{A}, \beta) \models \exists z[IFP_{\overline{x},X}\chi]z \ldots z \text{ iff } \mathcal{E}(\chi, \mathcal{A}, \beta) \in \hat{Q}_{MC}.$$

(ii) $\hat{Q}_{MC}$ is complete for IFP with respect to quantifier free reductions.

(iii) Replacing in the above definition of $\hat{Q}_{MC}$ the operator IFP by PFP results in a problem complete for PFP with respect to quantifier free reductions.

Proof: (i) follows directly from Lemma 9: Carrying out the iteration $X_{j+1} := \{w \mid Uw \wedge \hat{\varphi}_{TRUE}(w, X_j)\}$ in $\mathcal{E}(\chi, \mathcal{A}, \beta)$ simulates exactly the iteration in $(\mathcal{A}, \beta)$ over χ, i.e. these levels $X_j \subset U$ are copies of the original levels $X_j \subset A^s$, whence the claim follows.
(ii) Given a formula in normal form, $\varphi = \exists z[IFP_{\overline{x},X}\chi(\overline{x}, X)]z \ldots z$, we show that there is a tuple $\Phi_\chi = (\psi_O, \ldots, \psi_F)$ such that $\mathcal{A}^{\Phi_\chi}$ is a collection of isomorphic copies of $\mathcal{E}(\chi, \mathcal{A}, \beta)$ plus some extra points having no E–neighbours. Then the claim follows from (i), since for membership in $\hat{Q}_{MC}$ it makes no difference whether we consider $\mathcal{E}(\chi, \mathcal{A}, \beta)$ itself or a structure that consists of a collection of copies of $\mathcal{E}(\chi, \mathcal{A}, \beta)$ and of isolated points.

Φ_χ can be obtained via equality types: Let m be the number of nodes in a $\hat{\mathcal{C}}_{\overline{a}}$ modulo their isomorphism type (in other terms, m is the number of subformulas in χ), and d be the quantifier rank of χ. Choose $e \geq 1$ such that there are at least m equality types of e–tuples over a set with two elements, not taking into account the diagonal type where all e elements are equal. Let $r := e + s + d$ be the arity of the reduction Φ_χ. In each r–tuple over A we let the first e components encode the isomorphism type of a node (that is, to which subformula of χ this node corresponds), thereby not using the diagonal equality type, which says that all elements are equal. The second part will encode the path to the root, consisting of the root itself and d choices for the quantified variables.

To make this specific, set

$$\psi_U(\overline{x}) := x_1 \neq x_e \wedge x_1 = x_2 \wedge x_1 = x_3 \wedge \ldots \wedge x_1 = x_{e-1}$$

$$\wedge x_1 = x_{e+s+1} \wedge \ldots \wedge x_1 = x_r.$$

The U–part of $\mathcal{A}^{\Phi_\chi}$ will be the set

$$\{(w_1, w_1, \ldots, w_1, w_2; \overset{s}{a}, w_1, \ldots, w_1) \mid a_\nu, w_1, w_2 \in A, w_1 \neq w_2\},$$

where the first tuple $w_1, \ldots, w_1, w_2$ of length e is to be understood as the equality type coding U for each copy of $\mathcal{E}(\chi, \mathcal{A}, \beta)$, and the last d repetitions of w_1 simply fill up $\overset{s}{a}$ to an $s+d$–tuple. Similarly, one can assign equality types to the other nodes in the trees. On the set of such we can now define E, O, K, D, I, P, N, and F by means of quantifier free formulas. Here, we can use the last d elements in each tuple to indicate the values of the quantified variables.

We illustrate this procedure by an example, letting $\chi = \exists u(Xu \wedge \neg Xx)$ as earlier in this section. We have $d = 1, m = 4$ and can choose $e := 4$. This yields $r = 4 + 1 + 1 = 6$. For any distinct elements $a_1, a_2 \in A$, we can now construct a copy of $\mathcal{E}(\chi, \mathcal{A})$ consisting of r–tuples. We do this as before, where we fixed a_1, a_2, except that this time we avoid the diagonal type consisting only of a_1. We write $\mathcal{E}(\chi, \mathcal{A})^{a_1, a_2}$ for a copy of $\mathcal{E}(\chi, \mathcal{A})$ defined by means of a_1, a_2. In the earlier construction we used triples for the node type. We extend these triples by inserting a_2 at the second position. Hence, the U–part of $\mathcal{E}(\chi, \mathcal{A})^{a_1, a_2}$ consists of the tuples $(a_1, a_2, a_1, a_1; a, a_1)$. These nodes, the roots, operate as disjunctions. As we have no other disjunctions in the circuits, we can set $\psi_D := \psi_U$. Each node $(a_1, a_2, a_1, a_1; a, a_1)$ is connected to the nodes $(a_1, a_2, a_1, a_2; a, b)$ (for $b \in A$). They represent the subformula $Xu \wedge \neg Xx$ of χ, with values $x = a, u = b$. Since they are the only conjunctions, we set

$$\psi_K := x_1 \neq x_2 \wedge x_1 = x_3 \wedge x_2 = x_4.$$

Finally, each node $(a_1, a_2, a_1, a_2; a, b)$ has two nodes as predecessors, namely $(a_1, a_2, a_2, a_2; a, b)$ and $(a_1, a_2, a_2, a_1; a, b)$, representing Xu and $\neg Xx$, respectively.

The formula $\psi_E(\overline{x}, \overline{y})$ will be a disjunction of three formulas, corresponding to the three types of edges described above. For instance, to describe the edge from $(a_1, a_2, a_1, a_2; a, b)$ to $(a_1, a_2, a_1, a_1; a, a_1)$, we take the following formula:

$$x_1 = y_1 \wedge x_2 = y_2 \wedge x_5 = y_5 \quad \wedge \quad \psi_K(\overline{x}) \wedge \psi_D(\overline{y}),$$

the first part asserting that $\overline{x}$ and $\overline{y}$ belong to the same copy of $\mathcal{E}(\chi, \mathcal{A})$, namely $\mathcal{E}(\chi, \mathcal{A})^{x_1, x_2}$ (here it is important that we used only equality types that involve both a_1 and a_2), and that they are in the same tree. The second part specifies the type of the two nodes, i.e. which subformula they represent. In this example, this amounts to say that the first one is a conjunction, the second a disjunction, as this determines the type completely. In the general case, one has to write down the equality types chosen for the two nodes.

Similarly, the other two disjuncts of ψ_E can be written. It should also be clear, how the formulas ψ_O, ψ_I, ψ_P, ψ_N, and ψ_F can be obtained.

We do not need a domain relativizing formula $\psi(\overline{x})$ (compare Definition 1). Instead, we admit additional points in $\mathcal{A}^{\Phi_\chi}$ that do not belong to any copy of $\mathcal{E}(\chi, \mathcal{A}, \beta)$ but have no influence on the circuits computation.

(iii) can be proved as (ii), as the PFP–analogue of (i) holds. This is, because the fixed point is computed in the U–parts of the circuits in exactly the same way as in the original structures. □

Proof of theorem 8: We make use of the fact that $IFP \equiv LFP$ [10] and of the stronger normal form associated to LFP. That is, in $\chi(\overline{x}, X)$, where all negations are pulled in front of atomic formulas, there are no subformulas of the form $\neg X\overline{y}$. So we have $N = \emptyset$ in the construction of $\mathcal{E}(\chi, \mathcal{A}, \beta)$ above and we can drop P and N. We also dispense with U and replace F–arcs by usual E–arcs.

Let $\Phi_\chi = (\psi_O, \psi_E, \psi_K, \psi_D, \psi_I, \psi_U, \psi_P, \psi_N, \psi_F)$ be the reduction given in the last proof. We pass to a reduction to the class Q_{MC} by setting $\Phi^*_\chi := (\psi_O, \psi^*_E, \psi_K, \psi_D, \psi_I)$, where

$$\psi^*_E(\overline{x}, \overline{y}) := x_1 = y_1 \wedge x_2 = y_2 \wedge (\psi_E(\overline{x}, \overline{y}) \vee \psi_F(\overline{x}, \overline{y})).$$

If $\mathcal{E}^*$ denotes the σ_{MC}–reduct of $\mathcal{E} := \mathcal{E}(\chi, \mathcal{A}, \beta)$ with additional arcs Euv for nodes u, v with $\mathcal{E} \models Fuv$, then $\mathcal{A}^{\Phi^*_\chi}$ consists of a collection of copies of $\mathcal{E}^*$ and of some isolated points. As before, we have $\mathcal{A}^{\Phi^*_\chi} \in Q_{MC}$ just in case $\mathcal{E}^* \in Q_{MC}$. We claim that $\mathcal{E}^* \in Q_{MC}$ if and only if $\mathcal{E} \in \hat{Q}_{MC}$, which concludes the proof.

To see this, consider the formulas defining Q_{MC} (Definition 6) and $\hat{Q}_{MC}$ (Theorem 11). Write $Y_l (l \geq 0)$ for the stages of the fixed point computation induced by the former formula in $\mathcal{E}^*$, and $X_j (j \geq 0)$ for those induced by the latter formula in $\mathcal{E}$. Remember, the X_j can be identified with the (s–ary) stages induced by χ in $(\mathcal{A}, \beta)$. If h denotes the height of the tree like circuits constructed from χ, then we claim that $X_j \subset Y_{hj}$ for all j (here we identify nodes in $\mathcal{E}$ and $\mathcal{E}^*$; we will even refer to the copies of roots in $\mathcal{E}$ as roots etc., although $\mathcal{E}^*$ is no longer acyclic). This can be shown by induction on j. In fact, $w \in X_{j+1}$ means that w is a root to which TRUE is assigned if the truth values of leaves in P are defined by X_j (remember, each leaf in P evaluates a certain s–tuple with respect to its membership of X_j). Assuming $X_j \subset Y_{hj}$ we find that w is still evaluated to TRUE if the restriction of Y_{hj} to leaves in P, Y^*_{hj} defines the input. By monotonicity, the application of h iteration steps for Y in $\mathcal{E}^*$, starting from Y^*_{hj}, yields a subset of Y_{hj+h}. As within the trees the computation of truth values is done in the same way in $\mathcal{E}$ as it is in $\mathcal{E}^*$, we find that w is contained in that subset, and hence in $Y_{h(j+1)}$.

On the other hand, an induction on l shows that if $v \in Y_l$, then there is a $j \geq 0$ such that v is assigned TRUE in its tree if X_j defines the values for the input. Hence, $v \in X_{j+1}$, and X_∞ and Y_∞ agree on the U–part of $\mathcal{E}$. Now, comparing the definitions of Q_{MC} and $\hat{Q}_{MC}$ yields $\mathcal{E}^* \in Q_{MC}$ iff $\mathcal{E} \in \hat{Q}_{MC}$. □

We made essential use of the fact $IFP \equiv LFP$. In turn, this fact follows from Theorem 8, as one can replace IFP in Definition 6 by LFP (and hence, rewrite the $FO(Q_{MC})$–formulas as LFP–formulas).

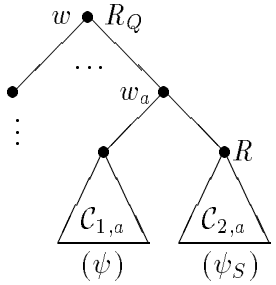
4 Adding quantifiers

Having defined the generalized circuits (Definition 10), we are going to enrich them by a new sort of nodes for additional quantifiers. These so called Q–nodes will evaluate an additional quantifier Q of signature σ_Q. They will check whether their input nodes code a structure in Q.

For simplicity, assume $\sigma_Q = \{S\}$ (S unary). It will become clear how to modify the construction for the case of an arbitrary σ_Q. Let R and R_Q be unary and $\hat{\sigma}_C^Q := \hat{\sigma}_{MC} \cup \{R, R_Q\}$. R_Q will denote the set of Q–nodes, and R will be used to distinguish between nodes that determine the domain of a query, on the one hand, and nodes that determine the relation S, on the other.

The class $\hat{Q}_C^Q$ of $\hat{\sigma}_C^Q$–structures will depend on the class Q (in a very uniform way, as long as σ_Q is fixed). Let us generalize the definition of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ (before Lemma 9) to the case of $\chi \in FO(Q)$, considering only the Q–step.

If $\chi = Qx, y; \psi(x), \psi_S(y)$, we take for each $a \in A$ a circuit $\mathcal{C}_{1,a} := \hat{\mathcal{C}}(\psi, \mathcal{A}, \beta\frac{a}{x})$ and a circuit $\mathcal{C}_{2,a} := \hat{\mathcal{C}}(\psi_S, \mathcal{A}, \beta\frac{a}{y})$ and let their roots be E–predecessors of a new node w_a. We take the roots of the $\mathcal{C}_{2,a}$ into R and let all the w_a be predecessors of a new node $w \in R_Q$, which is the root of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$.



It is clear, how χ is to be checked: Let $D := \{w_a \mid \mathcal{C}_{1,a}$ *evaluates to* TRUE$\}$ be the domain of a σ_Q–structure $\mathcal{D}$ with $S^D := \{w_a \in D \mid \mathcal{C}_{2,a}$ *evaluates to* TRUE$\}$. Then the whole circuit evaluates to TRUE if and only if $\mathcal{D} \in Q$, that is, if $(\mathcal{A}, \beta) \models \chi$. Similarly, if $\chi = Q\ \overset{r}{x}, \overset{r}{y}; \psi(\overset{r}{x}), \psi_S(\overset{r}{y})$, then we take $|A|^r$ many predecessors of w, each representing an r–tuple.

We aim for a class $\hat{Q}_C^Q$ that is derived from $\hat{Q}_{MC}$ by incorporating Q–nodes. As indicated by the subscript, we cannot achieve this by means of *monotone* circuits, since, unless Q is monotone, it is not possible to pull negations in front of atomic formulas. Recall, however, the definitions before Remark 7. They enabled us to deal with negation nodes in the circuits. We extend these definitions with the following clauses for Q:

$$\varphi_{Q+}(x) := R_Q x \wedge Qy_1, y_2\psi_1, \psi_2$$

$$\varphi_{Q-}(x) := R_Q x \wedge \neg Qy_1, y_2\psi_1, \psi_2, \text{ where}$$
$$\psi_1(y_1) := Ey_1x \wedge \exists y(Eyy_1 \wedge \neg Ry \wedge Yuy),$$
$$\psi_2(y_2) := \exists y(Eyy_2 \wedge Ry \wedge Yuy).$$

ψ_1 describes the domain of the Q–query, ψ_2 describes S on the set defined by ψ_1 (whence we need not write Ey_2x into ψ_2). Now, let

$$\hat{\varphi}_{TRUE}^Q(u, v, w, X) :=$$

$$[IFP_{z,x,Y}\left(z = u \wedge Ix\right) \vee \left(z = v \wedge \neg Px \wedge \neg\exists yEyx \wedge \neg Ix\right) \vee$$

$$\Big(z = u \wedge Px \wedge \exists y(Fxy \wedge Xy)\Big) \vee \Big(z = v \wedge Px \wedge \exists y(Fxy \wedge \neg Xy)\Big) \vee$$
$$\Big(\varphi_{EVAL}(x) \wedge \Big(z = u \wedge (\varphi_{CON+} \vee \varphi_{DIS+} \vee \varphi_{NEG+} \vee \varphi_{Q+})\Big)\Big) \vee$$
$$\Big(\varphi_{EVAL}(x) \wedge \Big(z = v \wedge (\varphi_{CON-} \vee \varphi_{DIS-} \vee \varphi_{NEG-} \vee \varphi_{Q-})\Big)\Big)](w).$$

Using this formula in place of $\hat{\varphi}_{TRUE}$, we will get an analogue of Lemma 9. To this end, we modify the definition of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ as outlined above, using Q–nodes and allowing for N–nodes within the circuits (we even will no longer have leaves marked with N; rather we encode a negated fixed point atom by a P–node connected to an N–node). Then, for any relation M assigned to the fixed point variable in χ, we have $(\mathcal{A}, \beta) \models \chi[M]$ if and only iff the generalized version of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ is a model of $\exists u, v, w(u \neq v \wedge Ow \wedge \hat{\varphi}^Q_{TRUE})[M]$. Again, this can be proved by an easy induction over $\chi \in FO(Q)$. Note that M is understood as an s–ary relation on the left side, and as a unary one on the right side. This is possible, as tuples over A are represented by single elements in the circuits. For the same reason, only a non vectorized application of Q occurs in $\hat{\varphi}^Q_{TRUE}$, as though the circuits can evaluate Q-queries of any arity. Of course, this must be compensated by the arities of the reductions that interprete the circuits in a given class of structures. We saw already in the previous section how the number of variables in a formula determines the arity of the reduction.

In analogy to Theorem 11 we set now $\hat{Q}^Q_C := Mod(\exists z_0(Oz_0 \wedge \exists uv(u \neq v \wedge [IFP_{w,X}Uw \wedge \hat{\varphi}^Q_{TRUE}(u, v, w, X)]z_0)))$. Observe that in $\hat{\varphi}^Q_{TRUE}$ there are no nested applications of Q. Coding the two nested IFP–operators into a single one (cf. [13]), we obtain a formula of the following shape.

Definition 12: (i) $IFP_1(Q)$ is the class of $IFP(Q)$–formulas, in which at most one IFP–operator occurs, and where all Q–quantifiers are in the scope of that IFP–operator.
(ii) $IFP_1(Q_1)$ is the class of $IFP_1(Q)$–formulas, where, furthermore, there is no nesting of Q–quantifiers. □

Lemma 13: There is a $\theta_Q \in IFP_1(Q_1)$ such that $\hat{Q}^Q_C = Mod(\theta_Q)$. □

To show the completeness of $\hat{Q}^Q_C$ we need the following analogue of the normal form for IFP. As already observed in [8], the proof of Fact 4 can be carried over to $IFP(Q)$, since the well known *transitivity lemma* [13] does not depend on the logic from where occurring subformulas stem.

Lemma 14 (Weak Normal Form): $IFP(Q) \equiv IFP_1(Q)$, more precisely, for every $\varphi \in IFP(Q)$ there is a φ' of the form $\exists z[IFP_{\overline{x},X}\chi]z \ldots z$, where $\chi \in FO(Q)$ such that $\models \varphi \leftrightarrow \varphi'$. □

Having the weak normal form available, we can proceed exactly as for Theorem 11 and get:

Theorem 15:
$\hat{Q}_C^Q$ is complete for $IFP(Q)$ with respect to quantifier free reductions. In particular, $IFP(Q) \equiv FO(\hat{Q}_C^Q)$. Replacing IFP with PFP in the definition of $\hat{Q}_C^Q$ yields a problem complete for $PFP(Q)$. □

Thus, we have found a Lindström logic for $PTIME^Q$ on ordered structures, as this oracle class corresponds to $IFP(Q)$ (similarly for $PSPACE^Q$ if the oracle tape is subject to the polynomial space bound). But it is noteworthy that Theorem 15 refers to arbitrary finite structures, not only to ordered ones.

Our characterization of oracle classes by Lindström logics has a close correspondence with a result of Gottlob [6] concerning relativizations of $LOGSPACE$. He found that for many important complexity classes $\mathcal{C}$, it suffices to adjoin a $\mathcal{C}$–complete problem Q to first order logic in order to obtain a logic for $LOGSPACE^{\mathcal{C}}$. So one might ask what the difference is between $FO(\hat{Q}_C^Q)$ and $FO(Q)$ (or $FO(Q_{MC}, Q)$), and whether this difference can be related to separating oracle classes. We come back to this question in the next section. In particular, we will show that, although $\hat{Q}_C^Q$ is derived from $\hat{Q}_{MC}$ and Q very uniformly, it is, in general, not definable by these two quantifiers.

For monotone classes Q, Hella has constructed problems complete for $LFP(Q)$ [11]. His result can also be restated by our circuit approach: working with additional nodes for the dual class of Q allows one to eliminate negation nodes as in the proof of Theorem 8 (alternatively, one could show that $IFP(Q) \equiv LFP(Q)$ holds for monotone classes Q).

As a corollary of our completeness proof, we can sharpen the above weak normal form.

Theorem 16 (Normal Form): (i) $IFP(Q) \equiv IFP_1(Q_1)$.
(ii) $PFP(Q) \equiv PFP_1(Q_1)$.

Proof: Let $\varphi \in IFP(Q)$. Then $\models \varphi \leftrightarrow \hat{Q}_C^Q \overline{v} \Phi$ for some tuple Φ of quantifier free FO–formulas. But $\hat{Q}_C^Q \overline{v} \Phi$ is equivalent to the $IFP_1(Q_1)$–formula obtained as follows: Take θ_Q from Lemma 13 and vectorize all operators to the arity r of Φ, that is, replace for instance $\exists u$ by $\exists \overset{r}{u}$, increase the arities of the fixed point variable, of the Q–operators, and of the symbols in $\hat{\sigma}_C^Q$. Then replace those symbols by the defining formulas in Φ. □

For ordered structures the above normal form has already been obtained via the equivalence to oracle classes [7]. The reason why nesting hierarchies for quantifiers collapse when fixed point operators are added is the following: the second order variables in $IFP(Q)$ can store the information gained in a Q–application and provide it for the next application. So $IFP_1(Q_1)$ allows even for polynomial nesting of Q–quantifiers. This will be the crucial point in the next section.

5 Constructing Oracles by Diagonalization

We are interested in a class Q such that the equivalence $IFP \equiv FO(Q_{MC})$ does not survive the adjunction of Q on both sides. For arbitrary finite structures one could apply a game theoretical argument and show that, for instance, $Q := EVEN$, the set of $\emptyset$-structures of even cardinality would do (this is part of a nesting hierarchy theorem for quantifiers and will be published elsewhere).

For the needs of descriptive complexity theory, it is more desirable to have a proof that goes through even if one restricts to linearly ordered structures. Here, IFP captures $PTIME$ and $IFP(Q)$ captures $PTIME^Q$ (the queries computable in polynomial time with access to an oracle Q) [7]. So, if we find Q as pointed out above, we have the following situation: $FO(Q_{MC})$ captures $PTIME$, but $FO(Q_{MC}, Q)$ does not capture all of $PTIME^Q$. At first glance, this looks like an argument against describing oracle complexity classes by means of quantifiers added to first order logic. However, as was shown in the previous section, $PTIME^Q$ is still captured by $FO(\hat{Q}_C^Q)$. So there is no lack of expressiveness of generalized quantifiers in this context. Rather, one has different notions of closure of classes of queries in logics and in complexity theory: $FO(Q_{MC}, Q)$ is the smallest regular logic containing $PTIME$ and Q, whereas the wider class $PTIME^Q$ is derived from a class of machines by closing it under a subroutine.

Under certain complexity theoretical assumptions, it is not hard to find classes Q as described above. Take, for instance Q_{HAM}, the class of graphs that have a Hamiltonian cycle. Then, by results of Stewart [19] and Gottlob [6], we have $FO(Q_{HAM}) \equiv LOGSPACE^{NP}$. Hence, as Q_{MC} is contained in this class, we find that $FO(Q_{MC}, Q_{HAM}) \equiv IFP(Q_{HAM})$ holds on ordered structures if and only if $LOGSPACE^{NP} \equiv PTIME^{NP}$.

To avoid any assumptions on complexity classes, we apply a diagonal argument, which is essentially due to J.F. Buss [1]. It was pointed out to the author by Y. Pnueli that this method might be useful in the context of fixed point logic.

In the following, let $\tau = \{\leq\}$, and consider only τ–structures $\mathcal{A}$ ordered by $\leq^{\mathcal{A}}$. $\mathcal{A}_n$ will denote an ordering $(A_n, \leq)$, where A_n has n elements. The oracle Q will be a class of $\sigma_Q := \{\preceq, R\}$–structures (for a unary R), that will be ordered by $\preceq$. Note, however, that our restriction to ordered structures refers only to the τ–structures, not to the quantifiers. So it may happen that in $\varphi = Q\overline{x}, \overline{y}_1, \overline{y}_2, \overline{z}; \psi(\overline{x}), \psi_{\preceq}(\overline{y}_1, \overline{y}_2), \psi_R(\overline{z})$ the formula $\psi_{\preceq}$ does not define an ordering on the domain $\{\overline{a} \in A^r \mid \mathcal{A} \models \psi[\overline{a}]\}$ for some $\mathcal{A}$, in which case φ will simply be false. Each σ_Q–structure which is ordered by $\preceq$ can be regarded as a word $w \in \{0,1\}^*$ (identify Rx with 1, $\neg Rx$ with 0 at position x).

Theorem 17: Assume $FO(Q_{IFP}) \equiv IFP$ (take, for instance Q_{MC} from Section 3 for Q_{IFP}). Then there is a class Q such that $FO(Q_{IFP}, Q) < IFP(Q)$ holds on ordered structures. Hence, $FO(Q_{IFP}) \equiv PTIME$ does not relativize to $FO(Q_{IFP}, Q) \equiv PTIME^Q$.

Proof: We are going to construct a class Q of strings which gives rise to a class $\mathcal{K}(Q)$ of pure orderings such that $\mathcal{K}(Q) \in IFP(Q)[\tau] \setminus FO(Q_{IFP}, Q)[\tau]$. $\mathcal{K}(Q)$ is

obtained from Q by feeding the oracle successively to itself, starting from the word 0 and adding one symbol at each step.

Let $s(Q)$ be the infinite string $(s(Q)_\nu)_{\nu\geq 1}$ with $s(Q)_1 = 0$, $s(Q)_{\nu+1} = 1$ if $s(Q)|_\nu := s(Q)_1 \ldots s(Q)_\nu \in Q$, $s(Q)_{\nu+1} = 0$ otherwise. Then, let $\mathcal{K}(Q) := \{\mathcal{A}_n \mid s(Q)_n = 1\}$. Storing the result of each step in a second order variable, it is easy to see that $\mathcal{K}(Q) \in IFP(Q)$, no matter what Q will be. Indeed, relativizing the domain of the Q–query, we can check words of length $1, 2, 3, \ldots, n-1$, one after the other.

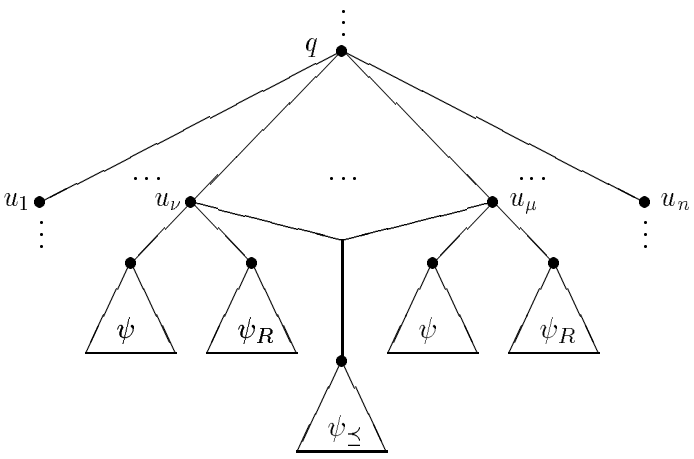
Next, we make sure that $\mathcal{K}(Q) \notin FO(Q_{IFP}, Q)$. Therefore, we diagonalize over this set of formulas. We will obtain Q in an infinite number of steps, each of which will affect only strings of a bounded length, be consistent with the previous steps, and will rule out one formula from defining $\mathcal{K}(Q)$.

Let $\varphi_1, \varphi_2, \ldots$ be an enumeration of $FO(Q_{IFP}, Q)[\tau]$. We aim for a Q and a sequence $n_1 < n_2 < \ldots$ with

$$(1)\ \ \mathcal{A}_{n_i} \models \varphi_i \ \text{ iff } \ s(Q)_{n_i} = 0 \ .$$

Then we are done, since the assumption $\mathcal{K}(Q) = Mod(\varphi_i)$ leads to:
$\mathcal{A}_{n_i} \models \varphi_i$ iff $\mathcal{A}_{n_i} \in \mathcal{K}(Q)$ iff $s(Q)_{n_i} = 1$ iff $\mathcal{A}_{n_i} \not\models \varphi_i$.

To make the left hand side in (1) easier to handle, let $\mathcal{C}_n^i$ be a circuit evaluating $\mathcal{A}_n \models \varphi_i$ similar to those of the previous section. So, $\mathcal{C}_n^i$ is a tree, the nodes of which are either conjunctions, disjunctions, negations, Q–nodes, Q_{IFP}–nodes or inputs labelled with TRUE or FALSE. A Q–node q for a subformula $Qx, y_1, y_2, z; \psi(x), \psi_\preceq(y_1, y_2), \psi_R(z)$, can here be realized as follows: It is connected to nodes $u_1, \ldots, u_n$ representing A_n. Each u_ν is connected to the roots of two subtrees, one evaluating $\mathcal{A} \models \psi[u_\nu]$, one checking $\mathcal{A} \models \psi_R[u_\nu]$. To each pair (u_ν, u_μ) a subtree is connected that evaluates $\mathcal{A} \models \psi_\preceq[u_\nu, u_\mu]$ (a Q–node can also be of higher arity r, in which case we have nodes $u_1, \ldots, u_{n^r}$):



The ψ–subtrees with output TRUE define a subset of $\{u_1, \ldots, u_n\}$. If this subset becomes an ordered σ_Q–structure in Q via the $\psi_\preceq$– and the ψ_R–subtrees, then the output of q is TRUE, otherwise it is FALSE. So, $\mathcal{C}_n^i$ does not depend on Q,

but its output does. Therefore we call it the *Q–output* of $\mathcal{C}_n^i$. The choice of Q_{IFP} and the realization of the Q_{IFP}–nodes is of no interest here.

The nesting of quantifiers Q in φ_i is now reflected in the position of the Q–nodes: q is said to be on level 1 if there are no Q–nodes in its subtrees. It is on level $j+1$ if the Q–nodes in its subtrees are all on levels $1, \ldots, j$ (at least one in level j).

Choose $d_1 < d_2 < \ldots$ such that the number of nodes in $\mathcal{C}_n^i$ is bounded by n^{d_i} and the height of this tree by d_i. Furthermore, choose $n_1 < n_2 < \ldots$ with

$$(2)\quad n_{i+1} - n_i^{d_i} - 1 > d_{i+1}^2 \log n_{i+1} \quad \text{for } i \geq 0$$

(in case $i = 0$ interpret $n_i^{d_i}$ as 0).

We are going to define classes Q_i of strings of length up to $n_i^{d_i}$ in a consistent way, that is, for $|w| \leq n_i^{d_i}$ and $j > i$ we will have $w \in Q_j$ iff $w \in Q_i$. Then, we let $Q := \bigcup_{i \geq 1} Q_i$. Assume, that for each $i \geq 1$:

$$(3)\quad \text{The } Q_i\text{–output of } \mathcal{C}_{n_i}^i \text{ is TRUE iff } s(Q_i)_{n_i} = 0.$$

Then (1) is also guaranteed, since $s(Q_i)_{n_i} = s(Q)_{n_i}$ (by consistency) and the Q–output of $\mathcal{C}_{n_i}^i$ is computed in the same way as the Q_i–output due to the choice of d_i. In fact, this output will only depend on Q–queries of length up to $n_i^{d_i}$.

Arguing in $\mathcal{C}_{n_1}^1$, Q_1 is constructed in l_1 steps, where $l_1 \leq d_1$ is the maximum number of nested quantifiers Q in φ_1. At each step, we choose some words w to be in Q_1 and some not, determining up to $d_1 \log n_1$ bits of $s(Q_1)$ at the same time. If we make sure that the string $s(Q)|_{n_1 - 1}$ is not tested in $\mathcal{C}_{n_1}^1$ then the bit $s(Q_1)_{n_1}$ can be chosen freely without influencing the Q_1–output in $\mathcal{C}_{n_1}^1$.

First step: Let $\{w_1^1, \ldots, w_{r_1}^1\}$ be the set of strings tested by Q–nodes on level 1 (this set of strings does not depend on Q). Since $r_1 \leq n_1^{d_1}$, there is a string v_1 such that $|v_1| \leq d_1 \log n_1$ and $0v_1$ is not an initial segment of any w_ν^1 (determine v_1 via majority vote: if more than half of the w_ν^1 start with 00 then let v_1 start with a 1, and so on). By taking the appropriate decisions (of belonging to Q_1 or not) for the w with $|w| \leq |v_1|$ we let $s(Q_1)|_{|v_1|+1} = 0v_1$. Paying attention only to these decisions (note that the w_ν^1 may very well be initial segments of $0v_1$), we determine the w_ν^1 to be elements of Q_1 or not, which gives us $\{w_1^2, \ldots, w_{r_2}^2\}$, the set tested on level 2.

Second step: Choose v_2 with $|v_2| \leq d_1 \log n_1$ such that $0v_1v_2$ is not a prefix of any w_ν^2, and make sure that $s(Q_1)|_{|v_1v_2|+1} = 0v_1v_2$. This can be done by determining Q_1 on the words w with $|v_1| + 1 \leq |w| \leq |v_1v_2|$ that have the prefix $0v_1$. Note that $w \notin \{w_1^1, \ldots, w_{r_1}^1\}$ for these w, whence we are free to choose.

Continuing this way, we have, **after l_1 steps**, determined the Q_1–output of $\mathcal{C}_{n_1}^1$, and we have chosen $v_1, \ldots, v_{l_1}$ with $|v_\nu| \leq d_1 \log n_1$ such that $0v_1 \ldots v_{l_1}$ is not a prefix of any of the tested words and $s(Q_1)$ begins with $0v_1 \ldots v_{l_1}$. By (2), we have $|0v_1 \ldots v_{l_1}| \leq 1 + d_1^2 \log n_1 < n_1$. Now, considering only words with prefix $0v_1 \ldots v_l$ we choose Q_1 consistent with the decisions made so far such that (3) holds for $i = 1$ (the left side of (3) is fixed, now).

To prove the induction step, suppose that Q_i is defined. We can construct Q_{i+1} in a consistent way guaranteeing (3): Whereas above, only one bit of $s(Q_1)$ was fixed, now there are $n_i^{d_i}+1$ bits of $s(Q_{i+1})$ fixed. By (2), one can, in $l_{i+1} \leq d_{i+1}$ steps, find $v'_1, \ldots, v'_{l_{i+1}}$ such that $s(Q_{i+1})$ begins with $s(Q_i)|_{n_i^{d_i}} v'_1 \ldots v'_{l_{i+1}}$, the length of this string is smaller than n_{i+1}, and it is not a prefix of a word tested in $\mathcal{C}^{i+1}_{n_{i+1}}$. Note that the left side in (2) is the number of new bits to be determined, the right side is the number of bits necessary for this procedure, which consists of up to d_{i+1} steps each determining up to $d_{i+1} \log n_{i+1}$ bits. □

By diagonal arguments of the same kind as in the last proof, one gets sharp conditions for logics to characterize *PTIME plus oracles*:

Theorem 18: a) For each countable family $(Q_\nu)_{\nu \in I}$ of quantifiers, there exists a class Q of strings such that $PTIME^Q$ is not contained in $FO((Q_\nu)_{\nu \in I}, Q)$ (not even on ordered structures).
b) There exists a class Q of strings such that $PTIME^Q$ is not captured by $IFP(Q)$ (on ordered structures) without domain relativization in the Q–rule.
c) The analogues of a) and b) hold for *PFP* and *PSPACE* (no matter, whether the oracle tape is subject to the space bound or not).

Proof: a) The proof of Theorem 17 did not depend on the form of the additional nodes in the $\mathcal{C}^i_n$.
b) The last proof depends only on the fact that a certain word of length $n_i - 1$ is not tested in $\mathcal{C}^i_{n_i}$. But without relativization, $IFP(Q)$–formulas can only test words of length n_i^r for some $r \geq 1$. Hence, one can diagonalize over this set of formulas and get the analogue of (1) in the last proof.
c) follows from a) and b). Note that the examples to prove undefinability results remain $IFP(Q)$–definable (only the diagonalization adapts to *PFP*). Hence, we make only use of the fact $PTIME^Q \leq PSPACE^Q$, which holds in any reasonable oracle model. □

Summing up, in the known characterization of $PTIME^Q$ by $IFP(Q)$ one can neither drop relativization of the domain, nor replace the second order variables by countably many Lindström quantifiers. The same holds for $PSPACE^Q$ and $PFP(Q)$.

Part a) of the theorem can be regarded as a general diagonalization method for relativized complexity classes, subsuming a variety of known separation results. For instance, if $(Q_\nu)_{\nu \in I}$ consists only of the ATC-quantifier (or of Q_{MC} from Section 3), we have $FO(ATC) \equiv ALOGSPACE$ [14], but $PTIME^Q > FO(ATC, Q) \equiv ALOGSPACE^Q$ (the latter equivalence holds for an appropriate oracle model, see [18]).

Furthermore, the result reveals the (well known) difference between *Turing* and *many one* reductions. This is, because $IFP(Q)$ captures those problems reducible to Q in the former way, whereas those problems that are many one reducible to Q, are already contained in $FO(Q_{IFP}, Q)$ (this can be seen by transferring the equivalence of *PTIME* with $FO(Q_{IFP})$ to the case of reductions).

Hence, the class $\mathcal{K}(Q)$ constructed above is *PTIME* Turing but not *PTIME* many one reducible to Q.

For the following corollary of Theorem 18 a), note that the syntax of $FO(Q)$ is defined as soon as σ_Q is fixed. Thus, Q can be regarded as a variable, similar to its role in oracle Turing machines.

Corollary 19 [17]: Let σ_Q contain a unary R and a binary $\preceq$. For each countable family $(Q_\nu)_{\nu\in I}$ of quantifiers, $FO((Q_\nu)_{\nu\in I}, Q)$ does not capture $PTIME^Q$ uniformly, i.e., there is no mapping f associating to each *PTIME* oracle machine M a formula $f(M) \in FO((Q_\nu)_{\nu\in I}, Q)$ such that $f(M)$ and M define the same query for each Q. □

6 Applications

We return to the question of whether a given logic $\mathcal{L}$ can be represented in the form $FO(Q)$ for a certain quantifier Q. We gave a positive answer to this for fixed point logics and circuit problems. Actually we proved considerably more, the *completeness* of these problems. In other words, it turned out that for those quantifiers no nesting is needed. As we now show, the phenomenon that a fixed nesting suffices, does not depend on our choice of the quantifier. In analogy to Definition 12, let $FO(Q_k)$ denote the formulas of $FO(Q)$ with up to k nested quantifiers Q (of any arity).

Theorem 20: Let Q be a class of structures with $FO(Q) \equiv IFP$ (or PFP). Then there is a $k \geq 1$ such that $FO(Q) \equiv FO(Q_k)$.

Proof: Let φ be the definition of Q_{MC} given in Definition 6, and let $\rho \in FO(Q)$ be an equivalent formula. Choose k with $\rho \in FO(Q_k)$. If ψ is an arbitrary $FO(Q)$–formula, then ψ is equivalent to $Q_{MC}\Phi$ for a quantifier free reduction Φ that has some arity $r \geq 1$. The latter formula is equivalent to the formula $\varphi\prime$ which is obtained from φ by r–vectorization and substitution of the circuit symbols by the definitions in Φ (compare the proof of Theorem 16). Applying the same procedure to ρ yields a $\rho\prime \in FO(Q_k)$ which is equivalent to $\varphi\prime$ and thus to ψ. □

Remark 21: A closer look at the proof of Theorem 20 shows that it goes through for all regular logics $\mathcal{L}$ that admit vectorization[2]. In particular, if for any pair of quantifiers Q, Q', we have $FO(Q) \equiv FO(Q')$, and the nesting hierarchy of $FO(Q)$ collapses, then so does that of $FO(Q')$.

Leaving the area of fixed point logics, we give another example of obtaining complete problems via circuits. By modifying the construction of Section 3, one gets complete problems for all levels of the polynomial hierarchy. For Σ^1_1, e.g., the circuits can be constructed as for $\hat{Q}_{MC}$, only the condition of being a *satisfied* circuit has to be replaced by being a *satisfiable* one (this case was

[2] We say that $\mathcal{L}$ admits vectorization if with each $\mathcal{L}$–definable class $\mathcal{K}$ also $\mathcal{K}^r$ is $\mathcal{L}$–definable. $\mathcal{K}^r$ is defined similarly to $Q^{r,P}$. E.g. if $\mathcal{K}$ is a class of graphs (A, E) then $\mathcal{K}^r := \{(A, E') \mid E' \subset A^{2r},\ (A^r, E') \in \mathcal{K}\}$

essentially treated in [16] though not with quantifier free reductions in the sense of Theorem 8).

Theorem 22: For each level Σ^1_k of the Polynomial Hierarchy there is a complete problem with respect to quantifier free reductions. Hence, the same holds for the co-classes Π^1_k.

Proof: We sketch how the circuits have to be adapted to this case. Let $\sigma_{SAT} := \{R, O, E, I, N, K, D\}$. R is binary; the remaining symbols have the same arity and intended meaning as in Section 3. Given $\chi = \exists X \psi$ with $\psi \in FO[\tau \cup \{X\}]$ for some τ and a relation variable X of arity $s \geq 1$, we can find a tuple Φ_χ of quantifier free τ–formulas such that for each $\mathcal{A} \in \mathrm{Str}[\tau]$, $\mathcal{A}^{\Phi_\chi}$ looks as follows:

- It is a collection of isomorphic copies of a circuit which is inductively constructed so as to evaluate ψ. Hence, the leaves are either in I, representing the input TRUE (we do not need input FALSE, as we allow for negations in the circuits), or undefined, i.e. corresponding to a subformula $X \overset{s}{z}$ evaluated at a tuple $\overset{s}{a}$ over A.
- The roots of the circuits constitute the O–part.
- If, within one circuit, there occur two undefined leaves both of which evaluate $X \overset{s}{a}$ for the same $\overset{s}{a}$ (possibly for different variables in the formulas), then there is a (symmetric) R–edge between these leaves.

Let Y be unary, and let Q_{SAT} be the class of models of

$$\exists Y(\exists x(Yx \wedge Ox) \wedge \varphi_{SAT}(Y) \wedge \varphi_{CONS}(Y)),$$

where $\varphi_{SAT}(Y)$ asserts that Y amounts to the nodes with value TRUE if the circuits are inductively evaluated, and $\varphi_{CONS}(Y) := \forall xy(Rxy \rightarrow (Yx \leftrightarrow Yy))$ says that Y is consistent with R. We can set

$$\varphi_{SAT} := \forall x((Ix \rightarrow Yx) \wedge$$
$$(Nx \rightarrow (Yx \leftrightarrow \exists y(Eyx \wedge \neg Yy))) \wedge$$
$$(Dx \rightarrow (Yx \leftrightarrow \exists y(Eyx \wedge Yy))) \wedge$$
$$(Kx \rightarrow (Yx \leftrightarrow \forall y(Eyx \rightarrow Yy)))).$$

We claim that Q_{SAT} is complete for Σ^1_1 under quantifier free reductions. In fact, for Φ_χ as given above, we have

$$\mathcal{A} \models \exists X \psi \text{ iff } \mathcal{A}^{\Phi_\chi} \in Q_{SAT}.$$

To conclude from left to right, assume $\mathcal{A} \models \psi[M]$ for a set M of s–tuples. Take for Y the nodes in $\mathcal{A}^{\Phi_\chi}$ that are assigned the value TRUE if the undefined inputs are defined according to M. The roots will then all be in Y, since the circuits evaluate to TRUE. For the converse, let Y be a subset of $\mathcal{A}^{\Phi_\chi}$ witnessing $\mathcal{A}^{\Phi_\chi} \in Q_{SAT}$. As asserted by φ_{CONS}, the restriction of Y to the undefined leaves defines a subset M of A^s. Obviously, we have $\mathcal{A} \models \psi[M]$.

To cope with the other levels of the Polynomial Hierarchy, we note that they are captured by first order formulas preceded by a prefix of second order quantifiers, namely $\exists X_1 \forall X_2 \exists X_3 \ldots \exists (or \forall) X_k$ for Σ^1_k. For this case, we can introduce k new relations in σ_{SAT} denoting k types of undefined input nodes, and define Q_{SAT} similarly, using however a prefix $\exists Y_1 \forall Y_2 \exists Y_3 \ldots \exists (or \forall) Y_k$ of monadic variables. Again, the restricted arity will be compensated by the quantifier free reductions. □

On the other hand, if second order logic, SO, has itself a presentation $SO \equiv FO(Q)$, then it collapses to a certain level: assume $Q \in \Sigma^1_i$. Then, as IFP-operators can be described by an existential second order variable, we have $IFP_1(Q_1) \subset \Sigma^1_{i+1}$ and, by Theorem 16, $SO \subset IFP(Q) \subset IFP_1(Q_1) \subset \Sigma^1_{i+1}$. (For $i = 1$ this conclusion has already been observed by Enderton.)

Theorem 23 The following statements are equivalent (the equivalence of (i) and (iii) is well known):

(i) The polynomial hierarchy collapses.

(ii) There is a class Q with $SO \equiv FO(Q)$.

(iii) There is a complete problem for the polynomial hierarchy with respect to $PTIME$ reductions.

(iv) There is a complete problem for the polynomial hierarchy with respect to quantifier free reductions.

Proof: We derived the equivalence of (i), (ii) and (iv) from the normal form for $IFP(Q)$ and from Theorem 22. (iv)→ (iii) is clear, and (iii)→(i) follows, for instance, from the equivalence of $PTIME$ reductions to IFP reductions. □

Acknowledgements

The author would like to thank H.-D. Ebbinghaus, E. Grädel, M. Grohe, L. Hella and Y. Pnueli for very helpful discussions on the material presented in this paper.

References

[1] J.F. Buss. Relativized alternation and space-bounded computation. *Journal of Computer and System Sciences*, 36:351–378, 1988.

[2] E. Dahlhaus. Skolem normal forms concerning the least fixpoint. In E. Börger, editor, *Computation theory and Logic, Lecture Notes in Computer Science 270*, pages 101–106. Springer-Verlag, 1987.

[3] A. Dawar. Generalized quantifiers and logical reducibilities. *Journal of Logic and Computation*, 5:213–226, 1995.

[4] H.-D. Ebbinghaus. Extended logics: The general framework. In J. Barwise and S. Feferman, editors, *Model–Theoretic Logics*, pages 25–76. Springer-Verlag, 1985.

[5] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1995.

[6] G. Gottlob. Relativized logspace and generalized quantifiers over finite structures. In *Proc. 10th IEEE Symp. on Logic in Computer Science*, pages 65–78, 1995.

[7] E. Grädel. On logical descriptions of some concepts in structural complexity classes. In E. Börger, H. Kleine Buning, and M.M. Richter, editors, *CSL '89: 3rd Workshop on Computer Science Logic*, volume 440 of *LNCS*, pages 163–175. Springer-Verlag, 1990.

[8] M. Grohe. *The Structure of Fixed–Point Logics*. PhD thesis, Albert-Ludwigs Universität Freiburg, 1994.

[9] M. Grohe. Complete problems for fixed–point logics. *Journal of Symbolic Logic*, 60(2):517–527, 1995.

[10] Y. Gurevich and S. Shelah. Fixed point extensions of first–order logic. *Annals of pure and applied logic*, 32:265–280, 1986.

[11] L. Hella. Fixpoint logic vs generalized quantifiers, 1994. Manuscript.

[12] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[13] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[14] N. Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[15] P. Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966.

[16] L. Lovász and P. Gács. Some remarks on generalized spectra. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 23:27–144, 1977.

[17] J.A. Makowsky and Y.B. Pnueli. Logics capturing relativized complexity classes uniformly. In *Proceedings of LCC*, 1994.

[18] J.A. Makowsky and Y.B. Pnueli. Computable quantifiers and logics over finite structures. In M. Krynicki, M. Mostowski, and L.W. Szczerba, editors, *Quantifiers: Logics, Models and Computation*, volume I, pages 313–357. Kluwer Academic Publishers, 1995.

[19] I. A. Stewart. Using the Hamiltonian path operator to capture NP. *J. of Comp. and System Sci.*, 45:127–151, 1992.

[20] I. A. Stewart. Incorporating generalized quantifiers and the least fixed point operator. In E. Boerger, Y. Gurevich, and K. Meinke, editors, *Proc. Computer Science Logic - CSL '93*, pages 318–333. Springer-Verlag, Lecture Notes in Computer Science 832, 1994.