Introduction

Complexity Theory investigates the computational resources required to solve decision problems. A decision problem is usually given as a set of strings. Typical measures of complexity are time or space, that is the number of steps, or the amount of tape cells respectively, used by a Turing machine that accepts this set of strings. In many cases however, decision problems represent *structural* properties, say the property of a graph having a Hamiltonian cycle. Hence, it is natural to generalize the notion of a decision problem from sets of strings to classes of finite structures (it is a generalization, because a string can also be seen as a finite structure).

Classes of structures can be characterized in logical languages. Descriptive complexity theory aims at investigating the syntactical means that are required to describe a decision problem and at relating these means to the computational complexity of the problem.

The early work of Fagin [13] established such a link. He showed that the problems in *NP* correspond exactly to those that are definable in existential second order logic. Since then, many complexity classes have been given a logical characterization. Enriching first order logic with a transitive closure operator leads to a logic, denoted TC, which captures $NLOGSPACE$, as was shown by Immerman [23]. A natural fragment of that logic, denoted DTC, captures $LOGSPACE$ [23]. Immerman and Vardi independently gave a characterization of $PTIME$ by means of inductive fixed point logic, *IFP*, a logic which is also based on first order logic and incorporates the possibility of forming fixed points of increasing sequences of sets, where each set is definable from its predecessor. Dropping the requirement of these sequences to be increasing leads to partial fixed point logic, *PFP*, which turned out to capture $PSPACE$ [1].

On the one hand, these results provide machine independent characterizations of complexity classes. On the other, they pave the way for applying model theoretic methods to the study of complexity.

There are two major drawbacks. First, model theory deals with (possibly) infinite structures. In fact, to a large extent, model theory and first order logic are tools to understand phenomena concerning the infinite. Many important properties of first order logic, like compactness, get lost under the restriction to finite structures. Moreover, first order logic proved too weak to express some properties of a very low complexity. Consequently, new tools, for instance variations of the Ehrenfeucht-Fraïssé method have been developed for finite model theory.

The second difficulty lies in the fact that the above mentioned logical characterizations of $LOGSPACE$, $NLOGSPACE$, $PTIME$ and $PSPACE$ were established for classes of linearly ordered structures, only. Whereas for $PSPACE$ (as for *NP*) there are also logical characterizations that do not require the structures to be ordered, it is an important open problem whether this holds for $PTIME$ and the classes below it. For many model theoretic arguments the restriction to ordered structures can be a problem, because every single element of an ordered structure has a different type. To give an example, winning strategies in Ehrenfeucht-Fraïssé games often rely on the fact that certain elements of a structure are indistinguishible, which is no longer the case in the presence of an ordering.

One contribution of this thesis is to exhibit the difficulties arising from linear orderings in the context of arity hierarchies. These hierarchies result from restricting the length of tuples to which transitive closure or fixed point operators are applied. They have been investigated thoroughly by Grohe [18]. He developed a general method to show that increasing the bound on the arities by one results in a proper increase of expressive power for each of the logics DTC, TC, IFP, and PFP.

For ordered structures, we establish a close correspondence between PFP^k, the k–ary fragment of PFP, and the class Space(n^k) for all $k \geq 1$. Similar work was done by Immerman for a variant of PFP that admits for simultaneous fixed point computation [24]. On the one hand, our results give a new proof for the strictness of the PFP–arity–hierarchy and also refine Immerman's result in some directions. On the other hand, they show that Grohe's method cannot directly be carried over to ordered structures.

For the IFP–arity–hierarchy on ordered structures, we get the following. Collapse would imply $PTIME \neq PSPACE$, whereas strictness would imply $LOGSPACE \neq PTIME$.

For second order logic, which captures the polynomial hierarchy, PH [37], we prove that collapse of the arity hierarchy (defined by the maximum arities of relation variables) would imply $PH \neq PSPACE$, whereas strictness would imply $NLOGSPACE \neq PH$.

Another important tool of enhancing the expressive power of logics (other than by recursion mechanisms) is to add generalized quantifiers. For various reasons, generalized quantifiers have become increasingly important in the study of finite models during recent years. One reason is that there is a close analogy between adding an oracle to a complexity class and adding a generalized quantifier to a logic. This line of investigation was started by Grädel [15], who showed that the Immerman-Vardi result $PTIME \equiv IFP$ remains true if, for any class Q of finite structures, $PTIME$ is enhanced with the oracle corresponding to Q, and IFP is enhanced with the quantifier that corresponds to Q. More precisely, for this result IFP has to be augmented by a uniform sequence of quantifiers derived from Q by simultaneous relativization and vectorization. For a graph property Q, this means we are not only allowed to form expressions like

$$Qx, y; \psi(x, y)$$

asserting in a structure $\mathcal{A}$ that the graph

$$(A; \{(a, b) \in A^2 \mid \mathcal{A} \models \psi[a, b]\})$$

has this property (this would be the classical way to adjoin a quantifier as suggested by Lindström in [29]). But also may we form

$$Q\ \overset{r}{z}, \overset{r}{x}, \overset{r}{y}; \theta(\overset{r}{z}), \psi(\overset{r}{x}, \overset{r}{y}) \quad (\overset{r}{z} \text{ abbreviates } z_1, \ldots, z_r \text{ and so on}),$$

meaning in a structure $\mathcal{A}$ that the graph $(V; E)$ with domain

$$V := \{\overset{r}{a} \mid \mathcal{A} \models \theta[\overset{r}{a}]\}$$

and edge relation

$$E := \{(\vec{a}, \vec{b}) \in V^2 \mid \mathcal{A} \models \psi[\vec{a}, \vec{b}]\}$$

is an element of Q. We write $\mathcal{L}(Q)$ in the following for the extension of a logic $\mathcal{L}$ with all relativized vectorizations of Q. Hence Grädel's result reads as

$$\mathit{PTIME}^Q \equiv \mathit{IFP}(Q) \text{ on ordered structures.}$$

This approach of generating uniform sequences of quantifiers from a class of structures by vectorization has proved fruitful in many areas of finite model theory. For example, adding such a sequence, based on an appropriate class, Q_{IFP}, to first order logic yields a logic equivalent to fixed point logic, written

$$\mathit{FO}(Q_{\mathit{IFP}}) \equiv \mathit{IFP}.$$

This has been shown independently in [9, 19, 21].

A major part of this thesis investigates the interaction between fixed point logic and generalized quantifiers. We will establish both further applications of generalized quantifiers in descriptive complexity and limitations of that approach.

As for the applications of quantifiers, we develop a general method of constructing quantifiers based on Boolean circuits. Added to first order logic, these quantifiers yield representations of fixed point logic (*IFP* or *PFP*) as well as of extensions thereof with further quantifiers ($\mathit{IFP}(Q)$ and $\mathit{PFP}(Q)$). In fact, even a stronger property holds for the classes of structures constructed. They turn out to be complete for the respective logics under quantifier free reductions (the same holds for the classes given in [9, 19, 21]). It is a variant of the monotone circuit value problem (well known to be *PTIME*–complete under *LOGSPACE*–reductions) which turns out to be *IFP*–complete under quantifier free reductions.

The quantifiers based on circuits evaluate the first order part of formulas in a similar way as this is done in the framework of circuit complexity classes (see [4] for an overview and further literature). This analogy can be further extended. Oracles are incorporated into circuit complexity classes usually by introducing gates that check whether or not the output of their input gates represents an element of the oracle set. Likewise, circuit based quantifiers can be modified in this way. This will enable us to construct for each given Q a problem $\hat{Q}$ complete for $\mathit{IFP}(Q)$ under quantifier free reductions.

These completeness results are closely related to normal forms. First, to prove the completeness requires normal forms for *IFP* and $\mathit{IFP}(Q)$ without nested fixed point operators. Such normal forms have been obtained by Immerman [22] and have been refined by Grohe [19]. In turn, the completeness of the class $\hat{Q}$ for $\mathit{IFP}(Q)$ yields a stronger normal form for this logic, where not only is it the case that just a single fixed point operator occurs, but furthermore the occurence of Q is confined to unnested occurence. For ordered structures, this strong normal form had already been obtained by Grädel as a corollary of $\mathit{IFP}(Q) \equiv \mathit{PTIME}^Q$.

We also study completeness results from a more abstract point of view. What distinguishes logics that have a complete problem (under quantifier free reductions)? We characterize them as so called self definable logics. This property captures the notion of a logic being able to define its own semantics. This takes up a familiar concept of mathematical logic, to give inner semantic definitions, as opposed to definitions in a meta–language (such as natural language). This concept has a computational counterpart, namely the existence of a universal machine for a given complexity class. As our approach will treat a logic in this context simply as a class of queries (regardless of its syntax), the equivalence of self definability with complete problems will include the case of structural complexity classes. As an example, we will restate and analyze a result of Dawar [10] saying that there is a logic of the form $FO(Q)$ that captures $PTIME$ (without an ordering available) provided there is any reasonable logic with that property.

On ordered structures, logics of the form $FO(Q)$, in the following referred to as Lindström logics, provide a tool to describe oracle complexity classes. For instance, the above mentioned class $\hat{Q}$ complete for $IFP(Q)$ satisfies

$$FO(\hat{Q}) \equiv PTIME^{Q}$$

on ordered structures. But is there a more uniform way to capture such classes, say a Lindström logic $\mathcal{L}$ that, for each further class Q, satisfies

$$\mathcal{L}(Q) \equiv PTIME^{Q}$$

on ordered structures? More generally speaking, is there a Lindström logic $\mathcal{L}$ that, for each further class Q, satisfies

$$\mathcal{L}(Q) \equiv IFP(Q)$$

for arbitrary finite structures?

We give negative answers to both questions and hence establish limitations to the use of generalized quantifiers in finite model theory. For both questions, this will be done in a slightly more general context.

For the ordered case, we can even start with $\mathcal{L} = FO(Q_\nu \mid \nu \in I)$ for any countable family $(Q_\nu)_\nu \in I$ of quantifiers and will still be able to find a Q such that $PTIME^{Q}$ is not contained in $\mathcal{L}(Q)$. This harks back to a theorem of Makowsky and Pnueli [33] who showed that there cannot be a translation between machines for $PTIME^{Q}$ and formulas of $\mathcal{L}(Q)$ that does not depend on the interpretation of Q. In fact, our proof was inspired by their construction.

For the arbitrary (finite) case, we show that not even $\mathcal{L} := \mathcal{L}^{\omega}_{\infty\omega}$, an infinitary logic with a finite bound on the number of variables in each formula, is a candidate for the inclusion $\mathcal{L}(Q) \geq IFP(Q)$ as long as the nesting of Q is subject to a finite bound in each $\mathcal{L}(Q)$–formula. To achieve this, we introduce an Ehrenfeucht-Fraïssé game. It combines the indefinitely played pebble game for $\mathcal{L}^{\omega}_{\infty\omega}$ as introduced by Barwise [5] with a game for generalized quantifiers introduced by Weese [40], and respects the nesting of quantifiers.

Our results will also hold for least fixed point logic, LFP. To cover this case, we develop a general method to pass from a given quantifier to an equivalent monotone quantifier.

1 Descriptive Complexity Theory

1.1 Background

The central notion of descriptive complexity is that of a Turing machine accepting a class of structures, as opposed to a set of strings. We give a definition of that concept. Our notation will mainly be based on that in [12]. *Structures*, denoted $\mathcal{A}$, $\mathcal{B}, \ldots$, will always be *finite* and of a *finite relational vocabulary*, denoted $\sigma, \tau, \ldots$. So we consider vocabularies (sometimes called signatures) of the form

$$\sigma = \{R_1, \ldots, R_s, c_1 \ldots, c_t\},$$

where (for $1 \leq i \leq s$) R_i is a relation symbol, equipped with a certain arity $r_i \geq 1$, and (for $1 \leq j \leq t$) c_j is a constant symbol. A σ–structure is then a tuple

$$\mathcal{A} = (A; R_1^{\mathcal{A}}, \ldots, R_s^{\mathcal{A}}, c_1^{\mathcal{A}} \ldots, c_t^{\mathcal{A}})$$

for a finite, nonempty set A (the *domain* of $\mathcal{A}$), relations $R_i^{\mathcal{A}} \subset A^{r_i}$ and constants $c_j^{\mathcal{A}} \in A$ ($1 \leq i \leq s$, $1 \leq j \leq t$).

If we consider classes of structures, we will always assume them to be closed under isomorphism. The class of all (finite) σ–structures for a given σ will be denoted $\mathrm{Str}[\sigma]$.

Occasionally, we will also deal with *interpretations*. These are pairs $\Im = (\mathcal{A}, \beta)$, where $\mathcal{A}$ is a structure and β an *assignment*, i.e. a mapping defined on a set of variables, V. V will always contain all (first or second order) variables that are relevant for the particular use of $\Im$, and β will assign a value in A to each first order variable x, and a subset of A^r to each second order variable X of arity r. If t is a term (i.e. a constant or a variable), then $\Im(t)$ denotes the value in A assigned to t.

Definition 1.1.1. Let σ be a vocabulary, say

$$\sigma = \{R_1, \ldots, R_s, c_1 \ldots, c_t\}.$$

- For $k \geq 1$, a *k–ary σ–query* is a mapping Q defined on $\mathrm{Str}[\sigma]$ such that
  - ◇ $Q(\mathcal{A}) \subset A^k$ for each $\mathcal{A} \in \mathrm{Str}[\sigma]$,
  - ◇ $\overset{k}{a} \in Q(\mathcal{A})$ iff $\pi(\overset{k}{a}) \in Q(\mathcal{B})$ for all $\mathcal{A}, \mathcal{B} \in \mathrm{Str}[\sigma]$ and each isomorphism $\pi : \mathcal{A} \xrightarrow{\sim} \mathcal{B}$.
- A *0–ary σ–query* is a class $Q \subset \mathrm{Str}[\sigma]$ (closed under isomorphism by our general assumptions).
- Let $\mathcal{A} \in \mathrm{Str}[\sigma]$, and let (for a binary symbol $\prec \notin \sigma$) $\prec^{\mathcal{A}} \subset A^2$ such that $(A, \prec^{\mathcal{A}})$ is a (total, irreflexive) linear order. Then the *encoding of $\mathcal{A}$ with respect to $\prec^{\mathcal{A}}$* is the string

  $$\mathrm{enc}(\mathcal{A}, \prec^{\mathcal{A}}) := 1 \ldots 1\$w_1\$w_2\$ \ldots \$w_s\$v_1\$v_2\$ \ldots \$v_t \ \in \{0, 1, \$\}^*,$$

  where

◇ the initial prefix $1\ldots1$ is of length $n := |A|$;

◇ for $1 \le i \le s$, $w_i \in \{0,1\}^*$ is a string of length n^{r_i} that encodes $R_i^{\mathcal{A}}$ in the following way: let $\overline{a}_1, \ldots, \overline{a}_{n^{r_i}}$ be an enumeration of all r_i–tuples over A in the lexicographical order with respect to $\prec^{\mathcal{A}}$; then for $1 \le \nu \le n^{r_i}$, the ν–th symbol of w_i is a 1 just in case $\overline{a}_\nu \in R_i^{\mathcal{A}}$;

◇ for $1 \le j \le t$, $v_j = \lambda_l \lambda_{l-1} \ldots \lambda_0$ encodes $c_j^{\mathcal{A}}$ in binary, that is if $c_j^{\mathcal{A}}$ is the μ–th element of the ordering $\prec^{\mathcal{A}}$ (now we start counting from 0), then $\mu = (v_j)_2 = \lambda_0 + 2\lambda_1 + \ldots + 2^l \lambda_l$.

- If, in addition, a tuple $\overset{k}{a} \in A$ is given, then we set

$$\mathrm{enc}(\mathcal{A}, \overset{k}{a}, \prec^{\mathcal{A}}) := \mathrm{enc}(\mathcal{A}, \prec^{\mathcal{A}})\$z_1\$z_2\$\ldots\$z_k,$$

  a string obtained by concatenating the encoding of $\mathcal{A}$ with binary encodings z_i of the elements a_i.

- Let M be a Turing machine with input alphabet $\{0,1,\$\}$ (we do not specify M more tightly; let us only assume that it has a separate input tape). For a k–ary σ–query Q, we say that M *accepts* Q if the following holds:

$$\text{for all } \mathcal{A} \in \mathrm{Str}[\sigma], \overset{k}{a} \in A, \text{ and every ordering } \prec^{\mathcal{A}} \text{ on } A,$$

$$\overset{k}{a} \in Q(\mathcal{A}) \text{ iff } M \text{ accepts } \mathrm{enc}(\mathcal{A}, \overset{k}{a}, \prec^{\mathcal{A}}).$$

  (Note that the left side does not depend on the choice of $\prec^{\mathcal{A}}$; hence the right side must not depend on $\prec^{\mathcal{A}}$ either.)

- Restricting the resources of M, or admitting nondeterminism, leads to definitions of *LOGSPACE*, *PTIME*, *NP*, *PSPACE* etc. as classes of queries.

□

It is well known that the (in-)equality of complexity classes in the above sense, i.e. as classes of queries, is equivalent to the (in-)equality of their versions as classes of languages. This can be shown by means of a straightforward encoding of strings as structures, see [12]. There is, however, a considerable difference: for classes of languages it is easy to equip Turing machines with clocks so as to obtain a recursive set of machines that represent, for example, the languages in *PTIME* (see [34]). In the context of queries, it is no longer clear how this should be done because built in clocks can come into conflict with the order invariance of a machine. We come back to this problem in 3.3, where we have a closer look at the invariance condition.

A natural way of describing queries is provided by logical languages. If $\mathcal{L}$ is a logic and σ a vocabulary, we let $\mathcal{L}[\sigma]$ denote the class of $\mathcal{L}$–formulas of vocabulary σ. A formula $\varphi(\overset{k}{x}) \in \mathcal{L}[\sigma]$ defines the k–ary σ–query Q given as

$$Q(\mathcal{A}) := \{\overset{k}{a} \in A \mid \mathcal{A} \models \varphi[\overset{k}{a}]\} \ (\mathcal{A} \in \mathrm{Str}[\sigma]).$$

Here, and throughout this thesis, we assume that the syntax of $\mathcal{L}$ involves individual variables $x_1, x_2, \ldots$, so that the above notation makes sense. We do not, however, use the notation $\varphi(\overset{k}{x})$ to indicate that $\{x_1, \ldots, x_k\}$ is the set of free variables of φ. Rather, we use it to define substitutions of variables with terms or assignments of values to variables (a_i to x_i in the above example).

Let $\mathcal{K}$ be either a logic or a complexity class. By identifying $\mathcal{K}$ with the $\mathcal{K}$–definable queries, or $\mathcal{K}$–computable queries respectively, we can use the notation $Q \in \mathcal{K}$ (Q a query).

Definition 1.1.2. Let $\mathcal{K}_i$ ($i = 1, 2$) be either a logic or a complexity class. We write

$$\mathcal{K}_1 \leq \mathcal{K}_2$$

to indicate that for all queries Q,

$$\text{if } Q \in \mathcal{K}_1, \text{ then } Q \in \mathcal{K}_2.$$

$K_1 \equiv K_2$ means $\mathcal{K}_1 \leq \mathcal{K}_2$ and $\mathcal{K}_2 \leq \mathcal{K}_1$. $K_1 < K_2$ means $\mathcal{K}_1 \leq \mathcal{K}_2$ and $\mathcal{K}_2 \not\leq \mathcal{K}_1$. □

Using existential second order logic, denoted Σ^1_1, the early work of Fagin established a logical characterization of a complexity class.

Theorem 1.1.3. **[13]** $\Sigma^1_1 \equiv NP$. □

For equivalences of the above type,

$$\mathcal{L} \equiv \mathcal{C},$$

for a logic $\mathcal{L}$ and a complexity class $\mathcal{C}$, we will also say

"$\mathcal{L}$ *strongly* captures $\mathcal{C}$"[1], or "$\mathcal{L}$ captures $\mathcal{C}$ *on arbitrary structures*".

In contrast to this, the classes below $PTIME$ have been given weaker logical characterizations, which apply only to *ordered structures.* For the sake of simplicity, the following definition deals with 0–ary queries, that is classes of structures, only.[2]

Definition 1.1.4. Let $<$ be a fixed binary relation symbol. For σ with $<\in \sigma$ we set

$$\mathcal{O}[\sigma] := \{\mathcal{A} \in \mathrm{Str}[\sigma] \mid (A, <^{\mathcal{A}}) \text{ is a (total, irreflexive) linear order}\},$$

[1] This terminology is due to [12], where it is introduced in a slightly different manner, namely by restricting the notion $Q \in \mathcal{C}$ to the case of *ordered* queries Q (the encoding of input structures refers to the given ordering, there) and then passing to ordered expansions $Q^<$ (to be defined below) of arbitrary queries. It is easy to see that "$\mathcal{L}$ strongly captures $\mathcal{C}$" is indeed equivalent to our notion of $\mathcal{L} \equiv \mathcal{C}$ provided that $\mathcal{C}$ is closed under $LOGSPACE$–reductions, which will be the case for all $\mathcal{C}$ to be encountered in this thesis.

[2] For $k \geq 1$, one could extend this definition by calling a k–ary query Q ordered, if it satisfies $Q(\mathcal{A}) = \emptyset$ in case $\mathcal{A}$ is not ordered. However, we will not need this general notion. The main reason why k–ary queries have been introduced at all, is that logical formulas are usually allowed to have free variables.

$$\mathcal{O} := \bigcup_{<\in\sigma} \mathcal{O}[\sigma].$$

Let $\mathcal{K}_i$ $(i = 1, 2)$ be either a logic or a complexity class. We write

$$\mathcal{K}_1 \leq_{\mathcal{O}} \mathcal{K}_2$$

to indicate that for all classes $Q \in \mathcal{O}$,

$$\text{if } Q \in \mathcal{K}_1, \text{ then } Q \in \mathcal{K}_2.$$

The notations $K_1 \equiv_{\mathcal{O}} K_2$ and $K_1 <_{\mathcal{O}} K_2$ are derived from $\leq_{\mathcal{O}}$ in the obvious way. □

Again, the case

$$\mathcal{L} \equiv_{\mathcal{O}} \mathcal{C},$$

for a logic $\mathcal{L}$ and a complexity class $\mathcal{C}$, will be of particular interest. For this relation we also say

"$\mathcal{L}$ captures $\mathcal{C}$"[3], or "$\mathcal{L}$ captures $\mathcal{C}$ *on ordered structures*".

We can always pass from a class of arbitrary structures to one of ordered structures. For the following definition, assume $<\notin \sigma$ (if $<\in \sigma$, replace it with a different binary symbol $\prec$).

Definition 1.1.5. For $Q \subset$Str$[\sigma]$, let

$$Q^< := \{(\mathcal{A}, <^{\mathcal{A}}) \mid (A, <^{\mathcal{A}}) \text{ is a (total, irreflexive) linear order}\}.$$

□

It is easy to see that for complexity classes $\mathcal{C}_1, \mathcal{C}_2$ that are closed under *LOGSPACE*–reductions, the statements $\mathcal{C}_1 \leq \mathcal{C}_2$ and $\mathcal{C}_1 \leq_{\mathcal{O}} \mathcal{C}_2$ are equivalent. That $\mathcal{C}_1 \leq \mathcal{C}_2$ implies $\mathcal{C}_1 \leq_{\mathcal{O}} \mathcal{C}_2$ is trivial. For the other implication, observe that for any class Q and $\mathcal{C} \in \{\mathcal{C}_1, \mathcal{C}_2\}$, we have

$$Q \in \mathcal{C} \text{ iff } Q^< \in \mathcal{C}.$$

To see this equivalence, remember that ordered expansions also occurred in Definition 1.1.1. We used the symbol $\prec$ for the orderings needed to encode structures as strings. Now, if a machine accepts Q, then it can be turned into an acceptor for $Q^<$ simply by adding the initial instructions "check whether $<$ is an ordering on the input structure (reject if it is not)" and "delete the encoding of the symbol $<$". For the converse, if M accepts $Q^<$, let M' on input enc$(\mathcal{A}, \prec^A)$ generate the string enc$((\mathcal{A}, <^A), \prec^A)$, where we set $<^{\mathcal{A}} := \prec^{\mathcal{A}}$. Then let M' simulate M. Since $Q^<$ is order invariant, M' will accept the input just in case $\mathcal{A} \in Q$.

[3]Again, in [12] this notion is obtained in a slightly different way, but the equivalence to our definition is obvious, as long as $\mathcal{C}$ is closed under *LOGSPACE*–reductions.

Now, assuming $\mathcal{C}_1 \leq_{\mathcal{O}} \mathcal{C}_2$, one can conclude as follows: If, for any class Q, $Q \in \mathcal{C}_1$ holds, then due to the above equivalence we have $Q^< \in \mathcal{C}_1$. Hence $Q^< \in \mathcal{C}_2$, which in turn implies $Q \in \mathcal{C}_2$. (To cover k–ary queries ($k \geq 1$), one can either extend the definition of ordered queries to this case and argue as before, or alternatively use the standard technique of reducing k–ary queries to 0–ary queries by means of constants.)

Consequently, we will use the notation $\leq_{\mathcal{O}}$ only if logics are involved. We are going to state results of the kind $\mathcal{L} \equiv_{\mathcal{O}} \mathcal{C}$ for the complexity classes *LOGSPACE*, *NLOGSPACE*, *PTIME*, and *PSPACE*. Apart from the case of *PSPACE*, there are no logics known that strongly capture one of these classes (of course, trivial logics without any conditions of computability are excluded here; see [10] or section 3.3 for details).

Definition 1.1.6. *Transitive closure logic, TC, is obtained by augmenting first order logic with the following formation rule:*

$$(TC) \quad \frac{\psi(\overset{k}{u},\overset{k}{v})}{[TC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u},\overset{k}{v})]\ \overset{k}{t}_1\overset{k}{t}_2} \quad (\overset{k}{t}_1,\ \overset{k}{t}_2 \text{ are } k\text{–tuples of terms}).$$

For this clause, the set of free variables of a formula is given as

$$\mathrm{free}([TC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u},\overset{k}{v})]\ \overset{k}{t}_1\overset{k}{t}_2) \ := \ (\mathrm{free}(\psi) \setminus \{\overset{k}{u},\overset{k}{v}\}) \cup \mathrm{free}(\{\overset{k}{t}_1,\overset{k}{t}_2\}).$$

The meaning of the above formula in a given interpretation $\Im = (\mathcal{A}, \beta)$ is defined as follows:

$$\Im \models [TC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u},\overset{k}{v})]\ \overset{k}{t}_1\overset{k}{t}_2 \ \text{ if and only if}$$

there is an $l \geq 0$ and a sequence $\overset{k}{a}_0 = \Im(\overset{k}{t}_1), \overset{k}{a}_1, \ldots, \overset{k}{a}_{l-1}, \overset{k}{a}_l = \Im(\overset{k}{t}_2)$ in A^k with

$$\Im \models \psi[\overset{k}{a}_i, \overset{k}{a}_{i+1}] \text{ (for each } i < l).$$

□

Note that further TC–operators may occur in ψ itself since the above rule may be applied repeatedly. Furthermore, by our convention, free(ψ) is not necessarily identical with the set $\{\overset{k}{u},\overset{k}{v}\}$. These two features will accordingly hold for the logics to be defined below.

Definition 1.1.7. *Deterministic transitive closure logic, DTC, is obtained by augmenting first order logic with the following formation rule:*

$$(DTC) \quad \frac{\psi(\overset{k}{u},\overset{k}{v})}{[DTC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u},\overset{k}{v})]\ \overset{k}{t}_1\overset{k}{t}_2} \quad (\overset{k}{t}_1,\ \overset{k}{t}_2 \text{ are } k\text{–tuples of terms}).$$

The set of free variables is defined for this clause as above. The meaning of the formula $[DTC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u},\overset{k}{v})]\,\overset{k}{t}_1\overset{k}{t}_2$ is that of the formula

$$[TC_{\overset{k}{u},\overset{k}{v}}(\psi(\overset{k}{u},\overset{k}{v}) \wedge \forall \overset{k}{z}\,(\psi(\overset{k}{u},\overset{k}{z}) \rightarrow \overset{k}{z}=\overset{k}{v}))]\,\overset{k}{t}_1\overset{k}{t}_2 \quad \text{(for a tuple } \overset{k}{z}\notin \mathrm{free}(\psi)\text{)}.$$

This amounts to saying that the path which witnesses the fact that $\overset{k}{t}_1$ and $\overset{k}{t}_2$ are in the (reflexive) transitive closure of the relation defined by ψ, must be uniquely determined at each step. □

Theorem 1.1.8. **[23]** *$DTC \equiv_{\mathcal{O}}$ LOGSPACE, $TC \equiv_{\mathcal{O}}$ NLOGSPACE.* □

For the following variants of *fixed point logic*, assume that the formation of first order formulas admits (free) second order variables, denoted $X, Y \ldots$, to be bound by the fixed point operators. Hence an interpretation, $\Im$, is tacitly assumed to assign values also to all free second order variables.

Definition 1.1.9. *Partial fixed point logic, PFP,* is obtained by augmenting first order logic with second order variables and the following formation rule (*PFP*). Even though free second order variables may occur for the formation of formulas at intermediate steps, each *PFP*–formula is required to have *no free second order variables.*

$$(PFP)\quad \frac{\varphi(\overset{k}{x},X)}{[PFP_{\overset{k}{x},X}\psi(\overset{k}{x},X)]\,\overset{k}{t}} \quad (k \geq 1,\ \overset{k}{t} \text{ is a } k\text{–tuple of terms, } X \text{ of arity } k).$$

For this clause, the set of free variables of a formula is given as

$$\mathrm{free}([PFP_{\overset{k}{x},X}\psi(\overset{k}{x},X)]\,\overset{k}{t}) \;:=\; (\mathrm{free}(\psi) \setminus \{\overset{k}{x},X\}) \cup \mathrm{free}(\{\overset{k}{t}\}).$$

The meaning of the above formula in a given interpretation $\Im = (\mathcal{A},\beta)$ is defined as follows:

$$\Im \models [PFP_{\overset{k}{x},X}\psi(\overset{k}{x},X)]\,\overset{k}{t} \text{ if and only if}$$

$$\Im(\overset{k}{t}) \in X^{\Im}_{\infty}, \text{ where}$$

$X^{\Im}_0 := \emptyset$
$X^{\Im}_{j+1} := \{\overset{k}{a}\in A^k \mid \Im \models \psi[\overset{k}{a}, X^{\Im}_j]\}$
$X^{\Im}_{\infty} := \begin{cases} X^{\Im}_j \text{ for the smallest } j \text{ with } X^{\Im}_j = X^{\Im}_{j+1} \text{ if it exists} \\ \emptyset, \text{ otherwise.} \end{cases}$ □

Definition 1.1.10. *Least fixed point logic, LFP*, is obtained in a similar way to *PFP*, except that the rule (*PFP*) is replaced with

$$(LFP)\ \frac{\varphi(\overset{k}{x}, X)}{[LFP_{\overset{k}{x},X}\psi(\overset{k}{x}, X)]\overset{k}{t}}\ (\overset{k}{t} \text{ is a } k\text{–tuple of terms, } X\ k\text{–ary, } \psi \text{ positive in } X).$$

The positivity condition requires that all occurrences of X in ψ are in the scope of an even number of negation signs. The semantics of *LFP* is defined as for *PFP*. It is, however, well known (and proved by an easy induction) that the positivity condition results in the monotonicity of the operator

$$F_\psi^{\Im} : M \mapsto \{\overset{k}{a} \in A^k \mid \Im \models \psi[\overset{k}{a}, M]\}\ (M \subset A^k).$$

As a consequence, the stages of the fixed point computation satisfy
$X_j^{\Im} \subset X_{j+1}^{\Im}$ (for all $j \geq 0$),
$X_\infty^{\Im} = \bigcap\{M \subset A^k \mid F_\psi^{\Im}(M) = M\} = \bigcup_{j\geq 0} X_j^{\Im} = X_{n^k}^{\Im}$ $(n := |A|)$. □

Definition 1.1.11. *Inductive* (or *inflationary*) *fixed point logic, IFP*, is defined as *PFP*, replacing however the rule (*PFP*) with a rule (*IFP*) which only differs from (*PFP*) by the denotation of fixed point operators as *IFP*. The semantics is also defined as it is for *PFP*, except that the sequence of stages is now as follows:
$X_0^{\Im} := \emptyset$
$X_{j+1}^{\Im} := X_j^{\Im} \cup \{\overset{k}{a} \in A^k \mid \Im \models \psi[\overset{k}{a}, X_j^{\Im}]\}$.
$X_\infty^{\Im}$ is derived from this sequence as it was in the case of *PFP*. Obviously, we have (as for *LFP*)
$X_\infty^{\Im} = \bigcup_{j\geq 0} X_j^{\Im} = X_{n^k}^{\Im}$. □

Directly from the definitions, one concludes that

$$LFP \leq IFP \leq PFP$$

(for the first inclusion, note that the stages $X_j^{\Im}$ defined by an *LFP*–operator form an increasing sequence; hence we can interprete an *LFP*–formula also by the *IFP*–semantics). Also the inclusions

$$DTC \leq TC \leq LFP$$

are well known. The last one is due to the equivalence

$$[TC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u}, \overset{k}{v})]\,\overset{k}{t}_1\overset{k}{t}_2 \quad\leftrightarrow\quad [LFP_{\overset{k}{x},X}(\overset{k}{x} = \overset{k}{t}_1 \vee \exists \overset{k}{z}\,(X\overset{k}{z} \wedge \psi(\overset{k}{z}, \overset{k}{x})))]\,\overset{k}{t}_2\,.$$

For the above variants of fixed point logics there are normal forms known [22, 18]:

Fact 1.1.12. 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.
The same holds for LFP (that is to say, for this case χ can be chosen positive in X). □

We summarize some results on fixed point logics, underlining their important role in descriptive complexity. A more detailed exposition can be found in [12].

Theorem 1.1.13.
(i) $LFP \equiv IFP$ [20].
(ii) $IFP \equiv_{\mathcal{O}} PTIME$ [22, 39].
(iii) $PFP \equiv_{\mathcal{O}} PSPACE$ [1].
(iv) $IFP \equiv_{\mathcal{O}} PFP$ if and only if $IFP \equiv PFP$ [2].

To capture $PSPACE$ on arbitrary structures, one can interleave PFP–operators with second order quantifiers, that is to say $SO(PFP) \equiv PSPACE$ [12]. Also SO itself has a computational counterpart, the so called polynomial hierarchy, PH. This class can be defined by polynomial time machines that are allowed to alternate between existential and universal states a fixed number of times. A precise definition (not needed here) can be found in [34].

Theorem 1.1.14. [37] $SO \equiv_{\mathcal{O}} PH$.

1.2 Arity hierarchies on ordered structures

Definition 1.2.1. Let $k_0 \geq 0$.
The k_0*–ary fragments* of DTC, TC, LFP, IFP, and PFP are obtained by restricting the application of the formation rules (DTC), (TC), (LFP), (IFP), and (PFP) respectively, to the case $k \leq k_0$. These fragments are denoted DTC^{k_0}, TC^{k_0}, LFP^{k_0}, IFP^{k_0}, and PFP^{k_0} respectively.
The k_0*–ary fragment of second order logic*, denoted SO^{k_0}, is defined by imposing the upper bound k_0 to the arity of relation variables in SO–formulas. □

The inclusions between the variants of transitive closure logic and fixed point logic obviously remain true if one passes to these fragments.

Remark 1.2.2. For all $k \geq 1$,
$DTC^k \leq TC^k \leq LFP^k \leq IFP^k \leq PFP^k$.

For the well known inclusion $SO \leq_{\mathcal{O}} PFP$ (see [12]), it is also possible to give a proof which preserves arities.

Lemma 1.2.3. For all $k \geq 1$, $LFP^k \leq SO^k \leq_{\mathcal{O}} PFP^k$.

Proof: The first inclusion is derived from the fact that a least fixed point is the intersection of all fixed points of that operator.
For the inclusion $SO^k \leq_{\mathcal{O}} PFP^k$, assume that there is a formula $\theta(\overset{k}{x}, X) \in FO[<]$ such that for an ordered structure $\mathcal{A}$, the stages $X_j^{\mathcal{A}}$ of $[PFP_{\overset{k}{x},X}\theta(X)]$ run through all subsets $M_0, M_1, M_2, \ldots$ of A^k, starting from $M_0 = \emptyset$, and ending (stationary) with $M_{2^{n^k}-1} = A^k$ (for $n := |A|$). Then $\exists X \psi(X)$ can be restated as

$$\varphi := \neg\exists \overset{k}{z}\, [PFP_{\overset{k}{x},X}(\neg\psi(X) \wedge \theta(X))]\, \overset{k}{z}\,.$$

To see this, assume first that $\mathcal{A} \models \neg\exists X \psi(X)$. Then, in $\mathcal{A}$, φ is equivalent to $\neg\exists \overset{k}{z}\, [PFP_{\overset{k}{x},X}\theta(X)]\, \overset{k}{z}$, which is false because its fixed point is A^k. If $\mathcal{A} \models \psi[M_j]$ for some $j < 2^{n^k}$, and j is the smallest number with this property, then $\mathcal{A} \models \neg\psi[M_i]$ for all $i < j$, and hence M_j coincides with the stage $X_j^{\mathcal{A}}$ of φ. Obviously, $X_{j+1}^{\mathcal{A}} = \emptyset = X_{\infty}^{\mathcal{A}}$, and so $\mathcal{A} \models \varphi$.
For the existence of θ, identify A with $\{0, \ldots, n-1\}$, enumerate A^k in the lexicographical order, $A^k = \{\overset{k}{a}_0, \ldots, \overset{k}{a}_{n^k-1}\}$, and identify each $M \subset A^k$ with a string $w \in \{0,1\}^{n^k}$, the ν-th symbol of which indicates whether or not $\overset{k}{a}_\nu \in M$. It is relatively easy to write down a formula $\chi(\overset{k}{x}, X) \in FO$ which says that $w' := \{\overset{k}{x} \mid \chi(\overset{k}{x}, X)\}$ is the successor (again in the lexicographical order) of $w := X$, and we can set $\theta(\overset{k}{x}, X) := \forall \overset{k}{x}\, X \overset{k}{x} \vee \chi(\overset{k}{x}, X)$. □

Arity hierarchies have been thoroughly investigated by Grohe, who showed the following:

Theorem 1.2.4. [18]
(i) For all $k \geq 1$, $DTC^k < DTC^{k+1}$.
(ii) For all $k \geq 1$, $TC^{k+1} \not\leq PFP^k$. Hence:
$TC^k < TC^{k+1}$, $LFP^k < LFP^{k+1}$, $IFP^k < IFP^{k+1}$, and $PFP^k < PFP^{k+1}$.
(iii) $DTC \leq_{\mathcal{O}} LFP^2$. □

For ordered structures, we are going to show the following (a proof of the next theorem will be given at the end of this section, when we will have provided a series of statements which imply it).

Theorem 1.2.5.
(i) $TC <_{\mathcal{O}} PFP^1$.
(ii) $PFP^k <_{\mathcal{O}} PFP^{k+1}$.
For the arity hierarchies on ordered structures, we have the following (*collapse* denotes the case that a certain k_0–ary fragment is equivalent to the full logic, *strictness* the negation of collapse):
(iii) For IFP,
collapse implies $PTIME < PSPACE$,
strictness implies $LOGSPACE < PTIME$.

(iv) For LFP,
collapse is equivalent with collapse for IFP.
(v) For SO,
collapse implies $PH < PSPACE$,
strictness implies $NLOGSPACE < PH$.

Part (i) implies that the general method for the proof on arbitrary structures [18] — a generalized graph reachability problem on k–tuples of a given structure — cannot be carried over to ordered structures. Consequently, part (ii) is based on a different approach. But it yields an alternative proof for arbitrary structures. Whereas this proof works for PFP, the arity hierarchies for other variants of fixed point logic and for second order logic are all linked to prominent open problems of complexity theory.

For SO, it does not matter whether we consider arbitrary structures or ordered ones, since it is possible to quantify over a binary symbol for an ordering. The same holds for its existential fragment Σ_1^1. Again, $\Sigma_1^{1,k}$ denotes its k–ary fragment. Ajtai showed this hierarchy to be strict:

Theorem 1.2.6. **[3]** For all $k \geq 1$, $\Sigma_1^{1,k} < \Sigma_1^{1,k+1}$. □

Conventions: For the rest of this section, we confine ourselves to ordered structures $\mathcal{A}$. n will be the cardinality of A and, according to the order, we will identify A with $\{0, \ldots, n-1\}$. This explains the use of constants $0, 1, 2, \ldots, max$. They are not regarded as part of the signature. Instead, they can be replaced in all formulas below by their first order definitions. Similarly, $\min(X)$ (and $\max(X)$) will be used to denote the smallest (largest) element of any nonempty subset X of a given ordered structure.
$\mathcal{A}_n$ will always denote the pure ordering $(\{0, \ldots, n-1\}, <)$.
A query defined on pure orderings will be called an *arithmetical predicate*. That is to say, an arithmetical predicate is either a class Q of pure orderings or (if it is r–ary, $r > 0$) a map π assigning to each $\mathcal{A}_n$ a subset of $\{0, \ldots, n-1\}^r$ (and $\emptyset$ to any non ordered structure).
Occasionally, r–tuples $\overset{r}{x} = (x_0, \ldots, x_{r-1}) \in \mathcal{A}_n$ will be used as n–ary expansions of numbers. We refer to these numbers by setting
$(\overset{r}{x})_n := x_0 n^{r-1} + \ldots + x_{r-1} n^0$.

In all logics to be investigated below it will be possible to define basic arithmetical relations between numbers (justifying the name "arithmetical predicate"). We start with DTC, where Immerman has observed the following:

Lemma 1.2.7. **[23]** The following arithmetical predicates can be defined in DTC (note that the function symbols are not part of the signature; we use them only for naming the relations; hence addition, etc. will only be partially defined).

- $x + y = z$ (to be read as $\pi : \mathcal{A}_n \mapsto \{(i, j, k) | i, j, k < n, i + j = k\}$),
- $x \cdot y = z$

- $(\overset{r}{x})_n + (\overset{r}{y})_n = (\overset{r}{z})_n$,
- $(\overset{r}{x})_n = 2^y$,
- $\mathrm{BIT}(\overset{r}{x}, w) \quad :\leftrightarrow \quad \exists\, \overset{r}{y}\overset{r}{z}\ ((\overset{r}{x})_n = 2^w + (\overset{r}{y})_n + (\overset{r}{z})_n \wedge 2^{w+1} | (\overset{r}{y})_n \wedge (\overset{r}{z})_n < 2^w)$
  (this means that the w-th bit of the binary expansion of $(\overset{r}{x})_n$ is a 1).

$\square$

The BIT–relation will allow us later to switch between binary and n–ary representations. To be parsimonious with arities, we need the following *pairing function*. As a consequence of the last lemma, it is also DTC–definable. More precisely: its graph is DTC–definable and in the formulas below, the function is only a denotation for its graph.

Definition 1.2.8. Fix a natural number $C \geq 1$. For all $n \geq 1, i < C, j \leq n/C - 1$, set $\langle i, j \rangle := Cj + i \ (< n)$. $\square$

For the admitted values of i and j, this is indeed a pairing function (i.e. injective). As it actually is a relation, it is always possible to define i and j from $\langle i, j \rangle$ in DTC. For each application, we have to specify a suitable number C.

We now turn to fixed point logics.

Definition 1.2.9. For $s, k \geq 1$, let PFP^k_s denote the class of PFP^k–formulas in which not more than s *nested* fixed point operators occur (the total number of these operators may well be greater than s). $\square$

We state two inclusions that imply $DTC \leq_{\mathcal{O}} PFP^1$. Both of them will also be important in their own right.

Theorem 1.2.10.
(i) $DTC \leq_{\mathcal{O}} IFP^1$.
(ii) $DTC \leq_{\mathcal{O}} PFP^1_1$.

Proof:
(i) First, we show IFP^1 to contain the unary versions of the arithmetical predicates in 1.2.7, which makes it possible (in the second step of the proof) to encode sequences of k-tuples (for a fixed $k \geq 1$) by a unary predicate.
To see that $x + y = z$ is in IFP^1, for instance, assume that $x < y$ and consider the sequence of intervals of natural numbers

$$[x, y], [x-1, y+1], \ldots, [0, z].$$

For $x, y \in \mathcal{A}_n$, these intervals are subsets of A_n if and only if the addition can be carried out within $\mathcal{A}_n$. Hence $x + y = z$ is equivalent to

$$[IFP_{u,P}\varphi]0 \ \wedge \ z = \max([IFP_{u,P}\varphi]), \quad \text{where } \varphi :=$$

$x \leq u \leq y \ \vee \ (\max(P) < max \ \wedge \ u+1 = \min(P)) \ \vee$
$(\min(P) > 0 \ \wedge \ u = \max(P)+1)$.
Similarly, one can express $x \cdot y = z$, $x = 2^y$ and hence $\mathrm{BIT}(x, w)$ in IFP^1.
Now, arguing in $\mathcal{A}_n$, we can encode each $a \in A_n$ by a string $s_a \in \{0,1\}^*$ of length $d := \lceil \log n \rceil$ using the binary expansion of a, i.e. we have $a = (s_a)_2$. Furthermore, if $\bar{a}$ denotes the k-tuple $(a_1, \ldots, a_k)$, we set $s_{\bar{a}} := s_{a_1} \ldots s_{a_k}$ (a string of length kd). Each $P \subset A$, when regarded as a string $p_0 \ldots p_{n-1}$ (where $p_i = 1$ iff $i \in P$), can code a sequence of k-tuples as follows: let $\bar{a}_1, \ldots, \bar{a}_l \in A^k$. Then we say

$$P \ \textit{encodes} \ \bar{a}_1, \ldots, \bar{a}_l, \text{ if}$$

$$p_0 \ldots p_{n-1} = s_{\bar{a}_1} 1 s_{\bar{a}_2} 1 \ldots s_{\bar{a}_l} 10 \ldots 0$$

(this is possible whenever $(kd+1)l \leq n$). Now, assume $\mathcal{A} \models [DTC_{\bar{u},\bar{v}}\psi]\bar{a}\bar{b}$ (where $\psi \in \mathit{IFP}^1$) via the deterministic (and thus unique) path $\bar{a} = \bar{a}_1, \ldots, \bar{a}_l = \bar{b}$ and assume

$$l \leq n/(kd+1) =: L(n).$$

Encode initial segments of this path by a sequence of subsets of A:
For $1 \leq j \leq l$, let P_j encode the sequence $\bar{a}_1, \ldots, \bar{a}_j$.
For $j > l$, let P_j encode the sequence $\bar{a}_1, \ldots, \bar{a}_l$ prolonged by $j - l$ copies of the last element $\bar{a}_l$.
(Note that the whole path can be encoded in this way due to the assumption on l).
Using the arithmetical operations which were shown to be IFP^1-definable above, it is easy to see that P_{j+1} is IFP^1-definable from P_j.
Summing up, we can express in IFP^1 that there exists a deterministic path from $\bar{a}$ to $\bar{b}$ of a length l with $l \leq L(n)$. In order to quantify over paths of length $l \leq L(n)^2$, regard such a path as a sequence of up to $L(n)$ paths of a length $l' \leq L(n)$. Thus, we can iterate the construction (note that the beginning and end points of partial paths are again uniquely determined at each step provided that we require these partial paths to be of maximal encodable length). After $k+1$ steps we are finally able to speak about paths of length $l \leq L(n)^{k+1}$. The observation that $L(n)^{k+1} \geq n^k$ (for big enough n) concludes the proof.
(ii) We sketch how a k–ary DTC–operator can be translated. Then we are done, because each DTC–formula is equivalent to one of the form

$$[DTC_{\overset{k}{u},\overset{k}{v}}\psi(\overset{k}{u},\overset{k}{v})]\overline{0}\,\overline{max},$$

where $\overline{0}$ and $\overline{max}$ are k–ary repetitions of 0 and max, respectively, and ψ is first order [23]. As for (i), we use a fixed point variable P and let its stages P_j encode the k–tuples of a path. The differences are the following:

- Whereas for (i) we encoded complete initial segments of a path, now we let P_j represent only the j-th tuple, $\overset{k}{a}_j$, of the path. Hence there will be no restriction on the length of a path, and we can make do with a single PFP–operator.

- In (i), the components of k–tuples were encoded in binary. This was possible because of the definability of the BIT–predicate. Here, we cannot make use of this technique as we must avoid fixed point operators in the formula to which our single unary *PFP*–operator is applied (and which defines the tuple $\overset{k}{a}_{j+1}$ from the encoding of $\overset{k}{a}_j$). Instead, we encode the components of the tuples by themselves, i.e. the $a_{j,\nu}$ $(1 \leq \nu \leq k)$ will all be elements of P_j. This forces us to somehow provide information in P_j as to how the components form a tuple.

We have to establish how a k–tuple $\overset{k}{a}_j$ is to be encoded, and have to show that (the encoding of) its successor is first order definable from that.

Encode a tuple $\overset{k}{a}=(a_1,\ldots,a_k)$ (to be thought of as a tuple $\overset{k}{a}_j$ in the path) as a pair (M,δ), where M is the set $\{a_1,\ldots,a_k\}$ and δ the *order type* of $\overset{k}{a}$, that is the quantifier free type of that tuple with respect to $(A,<^{\mathcal{A}})$. Let $T(k)$ be the number of order types of k-tuples, and fix any enumeration $\delta_1,\ldots,\delta_{T(k)}$ of these types.

Suppose $|A| = n \geq (k+1)(T(k)+k+2)$. Then we can encode the pair (M,δ) by means of the set P_j as follows: Choose $t \in \{1,\ldots,T(k)\}$ such that $\delta = \delta_t$. Due to the assumption on n, there is an interval I in $(A,<^{\mathcal{A}})$ of size $k+1+t$, having a distance > 1 from each element of M. For the sake of definiteness, choose I with a minimal left boundary. Set $P_j := M \cup I$, the union being disjoint.

Since I is the only interval in P_j of length $> k$, it is clear how to reconstruct $\overset{k}{a}$ from P_j and thus how to define P_{j+1} from P_j using only first order constructs and the formula ψ to which the DTC-operator had originally been applied. □

Knowing that all the arithmetical predicates from 1.2.7 as well as the pairing function from 1.2.8 are available in PFP^1, we can prove a refinement of $PFP \equiv_{\mathcal{O}}$ *PSPACE*. To this end, we must have a closer look at the levels that form *PSPACE*. For the whole class *PSPACE*, as defined in 1.1, the following details are of no interest. For the sake of simplicity, we deal only with 0–ary predicates, but as usual it is possible to pass to higher arities since constants are allowed.

Definition 1.2.11.
Let f: $\mathbb{N} \longrightarrow \mathbb{R}^+$ be such that $f(n) \geq \log n$ for all n, and let $Q \subset \mathrm{Str}[\tau]$ for some τ.
Then Q is said to be in Space$(f(n))$ if it is accepted by a deterministic Turing machine M of the following type.

- M has an *input tape* on which words over the alphabet $\{0,1,\$\}$ can be written. M will accept if and only if the input is a string of the form $\mathrm{enc}(\mathcal{A},\prec^{\mathcal{A}})$, for some structure $\mathcal{A} \in Q$ and some ordering $\prec^{\mathcal{A}}$ on $\mathcal{A}$ (not necessarily equal to the given order on $\mathcal{A}$).

- M has one *work tape* of size $\lceil f(n) \rceil$, containing symbols of a certain finite alphabet, the *work tape alphabet* (WTA) of M. Here we assume that the input encodes a τ-structure, and that n refers to the size of its domain.

Otherwise (i.e. if the input does not encode a τ-structure), M rejects it, and does not use more than $\log n$ cells of the work tape (this time, n refers to the length of the input).

- At the beginning of the computation, let each cell contain a 0, which is assumed to be a fixed element of the WTA.
- There is one *read head* for the input tape and one *read-and-write head* for the work tape. At the beginning, the heads are at their leftmost positions.
- Starting from the state q_0, M is at any time in one of a finite number of states. In each step of the computation, M scans the tapes at the actual head positions, writes a new symbol on the work tape, and moves the heads (independently) one step to the left or right (if possible). So the behaviour of M can be described by a collection of lines like

$$q_4 0 c q_7 d R L,$$

  meaning the following: in state q_4, scanning a 0 on the input tape and the symbol c on the work tape, change into state q_7, write a d on the work tape, move the input head right and the work tape head left. As M is deterministic, there is exactly one line associated with each state and pair of scanned symbols.
- An accepting run of M is one that halts in state q_1, otherwise M halts in state q_2.

□

Some consequences and remarks:

- Since the size of the WTA and the number of states can be chosen arbitrarily, we have $\text{Space}(f(n)) = \text{Space}(cf(n))$ for all $c > 0$.
- The space bound refers to the size of the structure, not to the size of the input strings encoding it. If τ contains only monadic relation symbols and the symbol $<$, the two notions coincide.
- The way we dealt with inputs that do not encode structures does not come into conflict with our earlier definitions of space bounded complexity classes (given in 1.1), because it can indeed be checked in logarithmic space, whether or not an input encodes a structure. Hence the above definition is a refinement of the earlier ones. In the following, we will tacitly assume that the input does represent a structure (otherwise relying on a logarithmic space bound that will be sufficient for all proofs).

Remark 1.2.12. Immerman has shown [24] that

$$\text{Iter}[\text{arity } k] \equiv_{\mathcal{O}} \text{Space}(n^k),$$

where the definition of Iter[arity k] is very similar to that of PFP^k. The only difference is that Iter[arity k] admits *simultaneous inductions* over a finite number of formulas, as opposed to the single formula to which a PFP–operator is applied. (To be precise, we note that the above equivalence holds for $k \geq r$, where r is the maximum arity of the vocabulary in question.) $\square$

From this result one can derive some consequences for the PFP–arity hierarchy (we will not use them later, whence we will not need a more precise definition of Iter[arity k] here).

First, observe that $PFP^k \leq$ Iter[arity k]. On the other hand, we have Iter[arity k] $\leq PFP^{k+1}$, which can be shown by the well known technique of encoding simultaneous inductions into single fixed point operators. Moreover, the above remark and the fact that Space(n^k)<Space(n^{k+1}) imply

$$\text{Iter[arity } k] <_{\mathcal{O}} \text{Iter[arity } k+1].$$

Hence we get $PFP^k <_{\mathcal{O}}$ Iter[arity $k+1$]$\leq PFP^{k+2}$, and thus

$$PFP^k <_{\mathcal{O}} PFP^{k+2}.$$

We will now relate PFP^k directly to Space(n^k) and to

$$\text{Space}(o(n^k)) := \bigcup_{f=o(n^k)} \text{Space}(f(n)).$$

We will not get an equivalence like that of Remark 1.2.12. The question of whether or not such an equivalence holds will be related to the *nesting hierarchy* $(PFP^k_s)_{s\geq 1}$ within PFP^k.

Nevertheless, we will be able to improve Immermans results (more precisely, their consequences for PFP) in two directions: first, we obtain $PFP^k <_{\mathcal{O}} PFP^{k+1}$. Second, this is witnessed by a class of pure orderings, as opposed to *word structures* of a vocabulary $\{<, R\}$ (R unary), which are required for the usually applied argument that Space(n^k)<Space(n^{k+1}). Finally, it should be mentioned that our approach will treat k and r (as given in Remark 1.2.12) independently. Hence the restriction $k \geq r$ will no longer be necessary.

Theorem 1.2.13. Let $k \geq 1$. Then
(i) $PFP^k \leq$ Space(n^k).
(ii) Space($o(n^k)$) $\leq_{\mathcal{O}} PFP^k$.

This theorem is a consequence of the following stronger statements, provided for later use:

Lemma 1.2.14. Let $k \geq 1$.
(i) If, for $s \geq 1$, the class Q is definable in PFP^k_s, then there is a machine M of WTA $\{0, 1, *\}$ that accepts Q, and that uses a work tape of size at most $(3s+1)n^k$.
(ii) Space($o(n^k)$) $\leq_{\mathcal{O}} PFP^k_2$.

Proof: (i) The claim follows directly from the proof of the fact $PFP \leq PSPACE$ (cf. 1.1.13), which is given by induction on PFP. We analyze the amount of additional space that is required for the work tape at each step:
Since FO is contained in $LOGSPACE$, and $\log n = o(n)$, we can, for $\varphi \in FO$ choose a machine M that checks φ in Space($o(n)$). We can further assume that the WTA of M for this case has at most two elements, 0 and 1, say. This is because we could encode other symbols by sequences of two symbols of a fixed length. Similarly, all the steps of existential or universal first order quantification that occur together in the derivation of a PFP–formula, increase the required space only by an amount which is $o(n)$, since the binary coding of single elements requires only logarithmic space. This term is clearly bounded by n^k for big enough n (which suffices, because we can check finitely many structures without using any tape cells).
For disjunctions, conjunctions, and negations, no additional space is necessary (that is to say, the maximum will do in the first two cases).
The step that remains is that of a k–ary PFP–operator. To check whether a formula $[PFP_{\overset{k}{x},X}\psi(\overset{k}{x},X)]\overset{k}{t}$ holds in a structure of size n, one can proceed as follows (we argue with additional work tapes; this can be realized on one work tape if one uses the symbol $*$ as a separator):

- Use two new tapes of size n^k to encode the old and the new (i.e. just being computed) value of X. The encoding can be done by the symbols $0, 1$ in the same way as relations were encoded into strings in Definition 1.1.1 (using again the input order $\prec$).
- Use a third tape of size $k \log n$ to go through all k–tuples of the structure. This can be done in the lexicographical order. For each tuple, check whether ψ holds (using the old value for X like a segment of an input tape) and, accordingly, write 0 or 1 on the tape for the new value of X.
- Use a fourth tape of size n^k as a binary counter for the stages X_j. So, in case no fixed point exists, this can be told from the fact $j = 2^{n^k}$.

The size of the third tape can be added to the term which is due to first order quantifiers and is $o(n)$. So apart from that term, the PFP–step increased the space requirements by $3sn^k$, and the claim follows.
(ii) is, in principle, also proved along the lines of a proof for the inclusion $PSPACE \leq_{\mathcal{O}} PFP$. However, it will take some effort in encoding to be parsimonious with the arities. In particular, we will make use of the pairing function defined above. This will allow us to use one individual variable for two values. Let $Q \in \mathcal{O}[\tau]$ for some τ with $<\in \tau$. Assuming $Q \in$ Space($o(n^k)$), we have to show that $Q \in PFP^k_2$. We make the following simplifications:

- $\tau = \{<, R\}$ for a symbol R of arity $r \geq 1$.
- The input order $\prec^{\mathcal{A}}$ coincides with the order $<^{\mathcal{A}}$ of the structure in question. (To see that this assumption is possible, one has to look at the line of the formula, to be given below, in which the symbol scanned by the

input head is defined. There, one would have to use a BIT–predicate that is defined with respect to $\prec$ rather than $<$.)

- The input will only consist of a $0,1$–encoding of R with respect to $<$. (By the last assumption, this amounts only to dismissing the encoding of the domain as a word $1\ldots1$ of length n, and the encoding of $<$.) Both of these strings could easily be reconstructed in Space($o(n)$), whence this assumption does not affect the proof. [4]

Now, assume that a machine M accepts Q on the basis of the above assumptions on encodings of structures, and does not require more than $f(n)$ cells on its work tape, where $f(n) = o(n^k)$. Note that there is now no restriction on the size of the WTA. Nonetheless, we will argue for a WTA of two symbols (0 and 1), below; but we will have to say how arbitrary WTAs can be dealt with.
A *configuration* of M is a quadruple (q, p_1, p_2, v), where q is the actual state, p_1 the position of the input head, p_2 the position of the work head, and v the actual content of the work tape.
Let P be a relation variable of arity k. Then a configuration can be encoded by P as follows (let the states of M be enumerated as $q_0, q_1, \ldots$, and remember the n–ary expansions of numbers):

$P\langle 0,j\rangle\bar{0}$	M is in the state q_j
$P\langle 1,w\rangle\bar{0}$	The w-th bit of the input-head position (in binary) is a 1
$P\langle 2,w\rangle\bar{0}$	The w-th bit of the work-head position (in binary) is a 1
$P\langle 3,x_0\rangle x_1\ldots x_{k-1}$	At position $(\overset{k}{x})_n (= x_0 n^{k-1} + \ldots + x_{k-1})$, there is a 1 on the work tape.

Note that for big enough n, all pairs $\langle i,j\rangle$ above can be chosen with $i < 4$ and $j \leq n/4 - 1$. In the last line, this is due to the assumption on f, which implies $f(n) \leq n^{k-1}(n/4 - 1)$ for big enough n. Hence we can choose $C := 4$ for the pairing function (see Definition 1.2.8). This is the only thing which would have to be changed for an *arbitrary* WTA. In fact, if there were more than two symbols in the WTA, we could add finitely many lines of the form $P\langle i,x_0\rangle x_1\ldots x_{k-1}$ for $i = 4,5,\ldots$, and let the Boolean combination of their truth values define the symbol at position $(\overset{k}{x})_n$.
Now, if δ is a line in M, say

$$\delta = q_j 10 q_l 1LR$$

(in state q_j, scanning a 1 (0) on the input tape (work tape), pass to state q_l, write a 1 on the work tape, move the input head (work head) left (right)),

we can write down a formula $\varphi_\delta(\overset{k}{x}, P) \in DTC$ having the following properties (for the definition of φ_δ see below):

[4] This reconstruction would have to yield a single bit at each step, not the complete strings. Therefore (and this is a standard argument) it can be carried out in space less than that required by the full strings.

- If P encodes a configuration (q, p_1, p_2, v) of M, then $\varphi_\delta(\overset{k}{x}, P)$ asserts that δ *applies* to that configuration (for δ as above this means $q = q_j$, there is a 1 at position p_1 of the input tape, and there is a 0 at position p_2 of the work tape).

- If δ applies, then $\{\overset{k}{x} |\ \varphi_\delta(\overset{k}{x}, P)\}$ encodes the next configuration of M.

So, remembering that q_0, q_1, and q_2 code the starting, accepting, and rejecting state respectively, we have

$$M \text{ accepts } \mathcal{A} \qquad \text{iff} \qquad \mathcal{A} \models \varphi,$$

where $\varphi =$

$$\forall \bar{x}[PFP_{\bar{x},P}(\forall \bar{z} \neg P\bar{z} \ \wedge \ x_0 = \langle 0,0\rangle \wedge x_1 = \ldots = x_{k-1} = 0) \quad \vee$$

$$(\forall u(P\langle 0,u\rangle \bar{0} \rightarrow u = 2) \wedge P\bar{x}) \quad \vee$$

$$P\langle 0,1\rangle \bar{0} \quad \vee$$

$$\bigvee_{\delta \text{ line in } M} \varphi_\delta(\bar{x}, P)]\bar{x}.$$

The first line initializes P as the start configuration. The second line guarantees that the current value of P remains the fixed point provided it represents a rejecting configuration. The third line guarantees $P = A^k$ for the next stage provided it represents an accepting configuration; then, it makes A^k a fixed point. The last line tells how to proceed from the configuration encoded by P to its successor.
Note that as long as the fixed point is not reached, there is exactly one line in the above formula (and exactly one disjunct φ_δ in the case of the last line) that can be true for a k–tuple $\overset{k}{x}$ (here, we tacitly assume that there are no lines δ with $q \in \{q_1, q_2\}$; they would not make any contribution to the behaviour of M). Hence the stages P_j of this formula do indeed reflect the run of M before the fixed point is reached.
Let δ be as above (any other δ can be treated similarly). For φ_δ we take the conjunction $\varphi_1 \wedge \varphi_2$, where φ_1 and φ_2 are as follows. (Text in square brackets consists only of comments; the text within formulas will either be clear from examples in earlier lines or is explained separately; we use bounded quantifiers only to indicate that the values are suitable for the pairing function.)

$$\varphi_1 :=$$

$$P\langle 0,j\rangle \bar{0} \quad \wedge$$

[the state is q_j]

$$\wedge \, \forall \overset{r}{y} \left(\forall w < n/4 (P\langle 1,w\rangle \bar{0} \leftrightarrow \mathrm{BIT}(\overset{r}{y}, w)) \rightarrow R\, \overset{r}{y}\right) \wedge$$

[the input head is scanning a 1]

$$\wedge \forall \overset{k}{y} \Big(\forall w < n/4 (P\langle 2,w\rangle \bar{0} \leftrightarrow \mathrm{BIT}(\overset{k}{y}, w)) \to \neg P\langle 3, y_0\rangle y_1 \ldots y_{k-1} \Big)$$

[the work head is scanning a 0].

$$\varphi_2 :=$$

$$\Big(x_0 = \langle 0, l\rangle \quad \wedge \quad x_1 = \ldots = x_{k-1} = 0 \Big) \quad \vee$$

[the new state is q_l]

$$\vee\ \exists w < n/4 \Big(x_0 = \langle 1, w\rangle \quad \wedge \quad x_1 = \ldots = x_{k-1} = 0 \quad \wedge$$

$$\forall \overset{r}{y}\ ((\overset{r}{y})_n \text{ is the (old) position of the input head } \to \mathrm{BIT}(\overset{r}{y} -1, w)) \Big) \quad \vee$$

[the input head moves to the left; its old position can be addressed as in φ_1]

$$\vee\ \exists w < n/4 \Big(x_0 = \langle 2, w\rangle \quad \wedge \quad x_1 = \ldots = x_{k-1} = 0 \quad \wedge$$

$$\forall \overset{r}{y}\ ((\overset{r}{y})_n \text{ is the (old) position of the work head } \to \mathrm{BIT}(\overset{r}{y} +1, w)) \Big) \quad \vee$$

[the work head moves to the right]

$$\vee\ \exists y_0 \Big(x_0 = \langle 3, y_0\rangle \quad \wedge$$

$$((y_0, x_1, \ldots, x_{k-1})_n \text{ is the (old) position of the work head} \quad \vee \quad P \overset{k}{x}) \Big)$$

[write a 1 on the work tape and keep the existing symbols 1].

φ_1 says that δ applies to the configuration given by P. Hence the x_ν do not occur there. φ_2 holds exactly for the set of k–tuples that encodes the successor configuration of P. Since the pairing function and the BIT–predicate are DTC–definable, φ_δ can be chosen in DTC, and φ can be chosen in PFP_2^k. □

Noting that $\mathrm{Space}(n^{k-1/2}) < \mathrm{Space}(n^k)$, one can already conclude from the last theorem that $PFP^k <_{\mathcal{O}} PFP^{k+1}$ via a class of word structures, say of vocabulary $\{<, R\}$ for a unary R (corresponding to a language over the alphabet $\{0,1\}$ usually used to separate $\mathrm{Space}(n^k)$ from $\mathrm{Space}(n^{k-1/2})$ by diagonalization). We aim at a class of pure orderings to witness $PFP^k <_{\mathcal{O}} PFP^{k+1}$. Therefore we explicitly carry out a diagonalization. Since we reuse the argument later on, we first recall a general method, which, in various forms, is often applied in mathematical logic.

Remark 1.2.15. (Diagonalization principle) Let $\mathcal{L}$ be a logic for which there exist only countably many τ–sentences for $\tau := \{<\}$. Then the class

$$Q_{\mathcal{L}} := \{\mathcal{A}_n \mid \mathcal{A}_n \models \neg\varphi_n\},$$

formed with respect to a fixed enumeration $(\varphi_n)_{n\geq 1}$ of the τ-sentences of $\mathcal{L}$, is not $\mathcal{L}$–definable.

(To get hierarchy results, we have to choose this enumeration in a way asserting that $Q_{\mathcal{L}}$ becomes definable in a certain other logic, $\mathcal{L}'$.)

Proof: This is clear because the assumption $Q_{\mathcal{L}}$ =Mod(φ_n), for some n, leads to the equivalence

$$\mathcal{A}_n \models \varphi_n \text{ iff } \mathcal{A}_n \models \neg\varphi_n.$$

$\square$

Theorem 1.2.16. For each $k \geq 1$, there is a class of orderings, Q_k, such that $Q_k \in \mathit{PFP}^{k+1} \setminus \mathit{PFP}^k$.

Proof: We apply the diagonalization principle to $\mathcal{L} := \mathit{PFP}^k$. All we have to do is to find an enumeration of the set of $\mathcal{L}[<]$–sentences, such that $Q_{\mathcal{L}} \in \mathit{PFP}^{k+1}$ holds. First, some notation:

- We will define Turing machines T, T_1 and T_2, which, on a given input, produce an output. This output is assumed to be the content of the work tape once the machine has halted. In particular, space bounds for the machines are upper bounds for the length of the output.
- For a sentence $\varphi \in \mathcal{L}[<]$, let M_φ denote a machine that accepts Mod(φ) according to Lemma 1.2.14(i). So M_φ has $\{0, 1, *\}$ as WTA and, on an input structure of size n, does not require more than $(3s_\varphi + 1)n^k$ cells on the work tape, where s_φ is the PFP–nesting depth of φ. In case $\varphi = \varphi_n$, we write M_n for M_φ and s_n for s_φ.
- $|\varphi|$ denotes the number of symbols in φ; $|M|$ denotes the length of a canonical encoding of M (the concatenation of its lines δ, say).

Claim: There is a machine T with the following properties:

- On unary input n, T produces an output (φ_n, M_n), where the sequence $(\varphi_n)_{n\geq 1}$ enumerates all $\mathcal{L}[<]$–sentences.
- T has n as a space bound.
- $|\varphi_n| \leq n^{1/2}$. Hence $s_n \leq n^{1/2}$.

Based on this sequence $(\varphi_n)_{n\geq 1}$, we can show that $Q_{\mathcal{L}} \in \mathit{PFP}^{k+1}$ and thus conclude the proof. To see this, we show

$$Q_{\mathcal{L}} \in \text{ Space}(n^{k+1/2}),$$

and apply Lemma 1.2.14(ii) (for $k+1$ in place of k). For the last line, assume we are given an input structure $\mathcal{A}_n$. Using T, we generate M_n. Next, we simulate M_n on $\mathcal{A}_n$. This can be done in a fixed alphabet, since the WTA of M_n did not depend on n. Only the number of states can vary for different n. So we use a logarithmic number of additional tape cells to encode the current state of

M_n in binary (M_n cannot have more than n states, because $|M_n| \leq n$). Hence the whole simulation can be carried out using $(3s_n + 2)n^k$ tape cells. Since $s_n \leq n^{1/2}$, this procedure checks in Space$(n^{k+1/2})$ whether or not $\mathcal{A}_n \models \varphi_n$.
Now for the above claim: There are machines T_1, T_2 performing the following tasks:
T_1, on input $\varphi \in \mathcal{L}[<]$, produces the output M_φ and has the space bound of $|\varphi|^m$ for some fixed $m \geq 1$ (for instance, the inductive proof of 1.2.14(i) can clearly be turned into a polynomial space algorithm).
T_2, on unary input n, produces a sentence $\varphi_n \in \mathcal{L}[<]$ thereby using not more than $\log n$ cells of the work tape (hence $|\varphi_n| \leq \log n$). All sentences of $\mathcal{L}[<]$ occur at least once in $(\varphi_n)_{n \geq 1}$.
This can be achieved as follows: On the one hand, one can apply the *PFP* formation rules in a systematic way so as to get a machine T_2' like T_2 but only having a polynomial space bound. On the other hand, the function $\log n$ is computable in logarithmic space, and hence so is $\log(\log n)$ (to be read as the smallest integer grater than this value). Let T_2 transform n into $\log(\log n)$ (both in unary representation) and then apply T_2' to this string.
Finally, let T, on input n, produce φ_n by simulation of T_2 (with logarithmic space requirements). Then, applying T_1 to this formula, M_n can be generated. The space needed to this end is bounded by $(\log n)^m$, and hence by n (for big enough n). Assuming that T deals separately with the finitely many exceptions, it is easily verified that this machine has the properties listed above. □

Theorem 1.2.17. For all $s, k \geq 1$, $PFP^k_s <_{\mathcal{O}}$ Space(n^k).

Proof: We can argue in exactly the same way as in the last proof. There we showed

$$Q_{\mathcal{L}} \in \text{ Space}(n^{k+1/2}),$$

for $\mathcal{L} := PFP^k$. An inspection of the proof shows that we can, in that line, replace Space$(n^{k+1/2})$ with Space(n^k) except for one point: when we carried out the simulation of M_n on $\mathcal{A}_n$, we only had the space bound $(3s_n + 2)n^k$ available. We had the upper bound $s_n \leq n^{1/2}$, and that is why we could take Space$(n^{k+1/2})$ for our argument.
But now, if we apply the diagonalization principle to $\mathcal{L} := PFP^k_s$, we can perform the above simulation with a space bound of $(3s+2)n^k$. Hence the whole procedure which checks whether or not $\mathcal{A}_n \models \varphi_n$ can be done in Space(n^k). □

Corollary 1.2.18. If, for $k \geq 1$,

$$PFP^k \equiv_{\mathcal{O}} \text{ Space}(n^k),$$

then the nesting hierarchy $(PFP^k_s)_{s \geq 1}$ within PFP^k is strict. □

Little is known about such nesting hierarchies for a fixed arity. The case of *LFP* has been investigated by Bosse [6].

Leaving PFP, we show some inclusions for other logics.

Lemma 1.2.19. For all $k \geq 1$, $IFP^k \leq_{\mathcal{O}} LFP^{2k}$.

Proof: Our proof is similar to the argument in [20], for showing that $IFP \equiv LFP$ on arbitrary finite structures. The situation simplifies for ordered structures since we can use counters for the levels of the fixed point process. We start with the case $k = 1$.
Consider a formula $\psi(x, X)$ (not necessarily positive in X). We can assume that the operator $X \mapsto \{x|\psi(x, X)\}$ is inflationary. That is to say, we always have $X \subset \{x|\psi(x, X)\}$ (otherwise replace ψ by $Xx \vee \psi$). Furthermore, we assume that the fixed point $[IFP_{x,X}\psi]$ is, in each structure of size n, already reached after $\lfloor n/2 \rfloor - 1$ steps of the iteration (otherwise replace ψ by $\psi(x, \psi(\_ , X))^5$, and repeat this replacement, if necessary). We simultaneously get those levels and their complements by positive formulas: Choose $\theta(x, X, Y)$ positive in both X and Y, and $\chi(x, X, Y)$ negative in both X and Y, so that (up to logical equivalence):

$$\psi(x, X) \quad = \quad \theta(x, X, \neg X) \quad = \quad \chi(x, X, \neg X).$$

We choose a binary relation variable X and encode the levels X_j of the fixed point process by the sets

$$X\langle 1, j\rangle_{\_} := \{x|X\langle 1, j\rangle x\}.$$

Their complements are encoded by the sets $X\langle 0, j\rangle_{\_}$. Here, $\langle \_ , \_ \rangle$ is the pairing function for $C := 2$, as defined in 1.2.8. It can be used because it is DTC– and hence LFP^2–definable (1.2.4), and its arguments are bounded by 2 and $n/2 - 1$ respectively (due to our assumption on ψ).
For any term t, the formula $[IFP_{x,X}\psi]t$ is equivalent to

$$\exists u \Big[LFP_{w,x,X} \Big(w = \langle 0, 0\rangle \quad \vee$$
$$(\exists 0 < v < n/2 \quad (w = \langle 0, v\rangle \quad \wedge \quad \neg\chi(x, X\langle 1, v-1\rangle_{\_}, X\langle 0, v-1\rangle_{\_}))) \quad \vee$$
$$(\exists 0 < v < n/2 \quad (w = \langle 1, v\rangle \quad \wedge \quad \theta(x, X\langle 1, v-1\rangle_{\_}, X\langle 0, v-1\rangle_{\_}))) \Big) \Big] \langle 1, u\rangle t.$$

For the case $k > 1$, one can argue similarly, splitting a counter $\overset{k}{v}$ into v_0 and $v_1, \ldots, v_{k-1}$, and using the predicates $X\langle b, v_0\rangle v_1, \ldots, v_{k-1}\ \_$ for $b \in \{0, 1\}$ and a $2k$–ary X. Again, v_0 is smaller than $n/2$ provided that $(\overset{k}{v})_n < n^k/2$. $\square$

Lemma 1.2.20. $TC \leq SO^2$.

Proof: We argue for ordered structures, which is possible because in SO^2, one can quantify over an ordering. Thus we can assume we have a TC–formula

[5] This notation describes the substitution of subformulas Xt with $\psi(t, X)$. We will go on using this kind of notation.

of the normal form (due to [23])

$$\varphi = [TC_{\overset{k}{u},\overset{k}{v}}\theta(\overset{k}{u},\overset{k}{v})]\bar{0}\overline{max}.$$

As in the proof of 1.2.10(i), we restrict ourselves to paths of length $l \leq n/k$. As for 1.2.10, this assumption can be removed by iterating the construction $k+1$ times.
A path of k-tuples can be transformed into a path of pairs of elements if one imposes a condition on every $2k$ subsequent elements, instead of on every two subsequent k–tuples (as the TC–operator does). We explain this procedure for $k = 3$:
A path $\overset{3}{a}, \overset{3}{b}, \overset{3}{c}, \ldots$ corresponds to the sequence of pairs
$(a_0, 0), (a_1, 1), (a_2, 2), (b_0, 3), (b_1, 4), (b_2, 5), (c_0, 6), \ldots$
together with the conditions:

- for any two subsequent pairs $(e_0, i_0), (e_1, i_1)$, we have $i_1 = i_0 + 1$, and
- for any six subsequent pairs $(e_0, i_0), \ldots (e_5, i_5)$, we have that if $3|i_0$ then $\theta(e_0, \ldots, e_5)$.

Thus the existence of a path of length $l \leq n/k$ from $\bar{0}$ to $\overline{max}$ is equivalent to (with unary Y and binary X)

$$\exists Y \exists X \Big(\chi(Y) \quad \wedge \quad \forall v \exists^{=1} u Xuv \quad \wedge \quad X00 \quad \wedge \ldots \wedge \quad X0(k-1) \quad \wedge$$

$$\exists v (Yv \quad \wedge \quad Xmax\ v \ \wedge \ldots \ \wedge \ Xmax(v+k-1)) \quad \wedge$$

$$\forall \overset{k}{u}\overset{k}{v}\, w \Big((Yw \quad \wedge \quad Xu_0w \quad \wedge \ldots \wedge \quad Xv_{k-1}(w{+}2k{-}1)) \rightarrow (\theta(\overset{k}{u}, \overset{k}{v}) \ \vee \ \overset{k}{u} {=} \overline{max}) \Big) \Big).$$

Here $\chi(Y)$ is a first order formula expressing that Y contains exactly the multiples of k, say

$$\chi(Y) = \quad Y0 \quad \wedge \quad \neg Y1 \quad \wedge \ldots \wedge \neg Y(k-1) \quad \wedge \quad \forall v (Yv \leftrightarrow Y(v+k)).$$

□

Although the SO^2–formula which occured in the last proof was existential, this property gets lost if we iterate the construction as required for paths of length $l > n/k$. In the next step, for instance, we would have to express a statement of the type
"there is a path of k–tuples such that for any two subsequent k–tuples there is another path ..."
But the phrase "for any two" is expressed by *universal* first order quantifiers.

Hence our proof does not yield $TC \leq \Sigma_1^{1,2}$.

Proof of Theorem 1.2.5:
(i) We have $TC \equiv_{\mathcal{O}} NLOGSPACE$, which, by Savitch's theorem is contained in Space$((\log n)^2)$. This is strictly contained in Space$(n^{1/2})$, and by 1.2.13(ii), we have Space$(n^{1/2}) \leq_{\mathcal{O}} PFP^1$.
(ii) was established in 1.2.16.

(iii) First assume collapse to a level k. Then
$PTIME \equiv_{\mathcal{O}} IFP^k \leq PFP^k <_{\mathcal{O}} PFP \equiv_{\mathcal{O}} PSPACE$.
Now, assume strictness. Then,
$LOGSPACE \equiv_{\mathcal{O}} DTC \leq IFP^1 <_{\mathcal{O}} IFP \equiv_{\mathcal{O}} PTIME$.
The last case can also be proved from the fact that $DTC \leq_{\mathcal{O}} LFP^2$ (Theorem 1.2.4(iii)).
(iv) follows from $LFP^k \leq IFP^k$ and Lemma 1.2.19.
(v) In case of collapse to level k, we get
$PH \equiv_{\mathcal{O}} SO^k \leq_{\mathcal{O}} PFP^k <_{\mathcal{O}} PFP \equiv_{\mathcal{O}} PSPACE$.
Now, assume strictness. Then
$NLOGSPACE \equiv_{\mathcal{O}} TC \leq SO^2 <_{\mathcal{O}} SO \equiv_{\mathcal{O}} PH$. □

The main reason why we could apply a diagonal argument to the *PFP*–arity hierarchy was the slow increase of space needed to decide model classes of these formulas. That is to say, an additional *PFP*–operator does not increase the required space too drastically. In contrast to this, it is not clear how arity hierarchies could be connected with *time* hierarchies, because a straight forward evaluation of a first order quantifier would already increase the degree of a polynomial bound by 1. There are, however, results in this direction. They are either based on additional restrictions (apart from arities) or on different models of computation.

As for the first approach: A hierarchy for second order logic, based on a bounded number of alternations and bounded arity, has been established by Makowsky and Pnueli [32].

An example of the second approach is the close connection between existential second order logic of fixed arity k, and the class NRAM(n^k), denoting the predicates computable by a *nondeterministic random access machine* in time n^k (in this result, the first order part of formulas is universal with quantifier depth k). This work is due to Grandjean [17].

The case of polynomially time-bounded nondeterministic Turing machines has been investigated by Lynch [31].

1.3 Generalized quantifiers and oracles

The logics introduced in 1.1 augmented first order logic by various mechanisms of recursion. Another way to increase the expressive power of logics is by means of generalized quantifiers. This approach plays an important role in descriptive complexity, and in mathematical logic in general. For the following definition, assume that $\mathcal{L}$ is a logic the syntax of which is given by a set of formation rules. Remember also our conventions on variables (given before Definition 1.1.2). Throughout this section, σ will denote a constant free vocabulary, that is to say

$$\sigma = \{R_1, \ldots, R_s\}.$$

For $1 \leq i \leq s$, let r_i denote the arity of R_i.

Definition 1.3.1. Let Q be a class of σ–structures. Regarding Q as a quantifier, we extend a logic $\mathcal{L}$ to $\mathcal{L}(Q)$ in the following way.

For each vocabulary τ, $\mathcal{L}(Q)[\tau]$ is the smallest set of formulas that is closed under the formation rules for $\mathcal{L}[\tau]$, 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 of rr_i–tuples over A. Thus we have indeed defined a σ–structure above.
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, or ψ_i, but does not occur in $\overline{x}$, or $\overline{x}_i$ respectively.
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 is A^r. $\square$

The formation rules of $\mathcal{L}$ may be subject to certain restrictions like the positivity condition for *LFP*. The last definition does not say how these restrictions are to be transferred to the syntax of $\mathcal{L}(Q)$. Since *LFP* is the only critical case to be encountered in this thesis, we state the definition of *LFP*(Q) separately.

Definition 1.3.2. Let Q be a *monotone* class of σ–structures. Monotonicity means that for $\mathcal{A} \in Q$ and $\mathcal{A}' = (A, R'_1, \ldots, R'_s) \in$Str$[\sigma]$ with relations $R_i \subset R'_i$ ($1 \leq i \leq s$), we always have $\mathcal{A}' \in Q$. Then we extend the notion of positivity in a relation variable X to the Q–clause as follows:
If all formulas in the list $\psi, \psi_1, \ldots, \psi_s$ are positive in X, then so is $Q\overline{x}, \overline{x}_1, \ldots, \overline{x}_s; \psi, \psi_1, \ldots, \psi_s$
(strictly spoken, one would have to define negativity as well in order to give an inductive definition; the Q–clause would also leave negativity unchanged).
With this notion of positivity, let *LFP*(Q) be the logic obtained from augmenting the *LFP* formation rules by the Q formation rule. Note that the monotonicity properties associated with positive formulas (see Definition 1.1.10) remain valid for *LFP*(Q)–formulas because of the monotonicity of Q. $\square$

We confined ourselves to classes Q of constant free vocabularies. We did this to avoid uniqueness problems for formulas that would have to define the constants. For practical purposes this is no restriction as the following (popular) example shows.

Example 1.3.3. Let $\sigma := \{E, S, T\}$ with unary S, T, and binary E, and take for Q the class of models of

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

Obviously, each formula $[TC_{\overline{x},\overline{y}}\chi]\overline{u}\overline{v}$ is equivalent to

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

Therefore, we write Q_{TC} for Q. Inductions on either of the logics yield[6]:

$$FO(Q_{TC})\equiv TC.$$

The syntax of the Q–clause enabled us to assert connectivity of a graph the vertices of which are tuples rather than single elements. In the sense of Lindström [29], the above Q–clause is restricted to the case where $r=1$, and where ψ is dropped. So, in that sense, 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 1.3.4. 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, set

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

$\square$

The cases where domain relativization is dropped correspond then to the adjunction (in the sense of [29]) of pure vectorizations. For $r\geq 1$, the r–vectorization of Q is defined as

$$Q^r:=\{(A;S_1,\ldots,S_s)\mid(A^r;S_1,\ldots,S_s)\in Q\}.$$

It is well known that extending first order logic with quantifiers leads to a logic that is minimal under the requirements of making these quantifiers expressible and of satisfying some natural closure properties. We will often make use of this argument and make it precise now.

The following definition refers to a logic as the class of queries that are definable in it. It should be obvious what, e.g., the negation or existential quantification of a query is. For a thorough treatment of logical systems the reader is referred to [11].

Definition 1.3.5. A logic $\mathcal{L}$ is called *semi–regular*, if it contains all (queries defined by) atomic and negated atomic formulas , and is closed under the following operations:

- Existential and universal first order quantification.
- Substitution of relation symbols with definable predicates. This means: In any $\mathcal{L}$–formula ψ that contains a relation symbol R, one can replace R by a relation (of matching arity) which is given by any other $\mathcal{L}$–formula $\varphi(\overline{x})$. The resulting $\mathcal{L}$–formula is usually denoted $\psi(R/\lambda\overline{x}\varphi(\overline{x}))$.

[6] See the proof of Lemma 1.3.6 below as to how this induction can be carried out.

- Relativization: For an $\mathcal{L}[\tau]$–formula φ, and a monadic $P \in \tau$ we can form φ^P in $\mathcal{L}$, a formula that is true in a τ–structure $\mathcal{A}$ if and only if $P^{\mathcal{A}} \neq \emptyset$ and φ holds in the substructure generated by $P^{\mathcal{A}}$.

$\mathcal{L}$ is called *regular*, if it is *semi–regular* and closed under negation. □

All logics studied in this thesis are at least semi–regular.

Lemma 1.3.6. Minimality Lemma: Let $\mathcal{L}$ be regular and assume $Q^{r,P} \in \mathcal{L}$ for all $r \geq 1$. Then $FO(Q) \leq \mathcal{L}$. Also, if several quantifiers are adjoined 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, assume that $\psi, \psi_1, \ldots, \psi_s$ are $FO(Q)$–formulas, which have $\mathcal{L}$–translations (we will use the same notation for the translations). The formula
$Q \overset{r}{x}, \overline{x}_1, \ldots, \overline{x}_s; \psi, \psi_1, \ldots, \psi_s$
is equivalent to
$\varphi_r(P/\lambda \overset{r}{x}\, \psi,\, S_1/\lambda\overline{x}_1\psi_1, \ldots, S_s/\lambda\overline{x}_s\psi_s)$,
where $\varphi_r \in \mathcal{L}$ is a definition of $Q^{r,P}$. □

Note that we did not use relativization in the last proof (but we will need this property later). Instead, we assumed the relativized vectorizations of Q to be $\mathcal{L}$–definable. Note that for many familiar logics this amounts to saying that Q itself is $\mathcal{L}$–definable. This is because constructs like transitive closure or fixed point operators can be vectorized and relativized. We will refer to logics of the type $FO(Q)$ as Lindström logics.

Complexity theory, too, provides a method of enhancing a class of queries with a further query: *oracles*. Let σ be as in Definition 1.3.1, and $Q \subset \mathrm{Str}[\sigma]$. In the same way as we defined Turing machines that accept queries we are now going to extend these machines with an oracle tape that can ask the query Q in one step. So an oracle can be regarded as a subroutine that does not require any resources (or only space resources, depending on the oracle model that is considered).

Definition 1.3.7. Let Q be a class of σ–structures and $k \geq 0$. An *oracle Turing* machine M is defined to accept k–ary queries in the same way as described in Definition 1.1.1 but it has, in addition to its input and work tapes, an *oracle tape*. Using Q as an oracle, M operates as follows:

- M is allowed to write strings $w \in \{0, 1, \$\}^*$ on the oracle tape. This tape is a write-only tape, that is the behaviour of M may not depend on the symbol scanned by the oracle head, except for the following case:

- M has three states $s_{\text{query}}, s_{\text{yes}}$, and s_{no}, designated to query the oracle Q as follows:

- Assume M is in state s_{query}. Let w be the content of the oracle tape.

If w =enc($\mathcal{A}, \prec^{A}$) for some structure $\mathcal{A} \in Q$ and some ordering $\prec^{\mathcal{A}}$ on A, then M changes into state s_{yes}, otherwise it changes into state s_{no}. In either case, M erases the oracle tape and moves the oracle head back to the leftmost position.

□

Subjecting M to resource bounds leads to relativized versions of complexity classes. We restrict ourselves to the case where M is deterministic.

Definition 1.3.8. A query $\tilde{Q}$ is in $LOGSPACE^{Q}$, if it is accepted by a deterministic oracle Turing machine M which has access to the oracle Q and, on input of size n, does not use more than $\mathrm{O}(\log n)$ cells of its work tape.
Similarly, $PTIME^{Q}$ is defined by imposing a polynomial bound on the number of steps. □

So far, these definitions are in line with the so called Ladner-Lynch oracle model [27] since they do not restrict the size of the oracle tape. There is, however, an implicit *polynomial space bound* for this tape. For $PTIME^{Q}$ this is clear. For $LOGSPACE^{Q}$ it is also well known and can be shown by counting the number of possible configurations of a machine (where a configuration does not include the content of the oracle tape). For the relativization of *PSPACE* we deviate from the Ladner-Lynch model in that we explicitly impose a polynomial space bound for the oracle tape.

Definition 1.3.9. A query $\tilde{Q}$ is in $PSPACE^{Q}$, if it is accepted by a deterministic oracle Turing machine M which has access to the oracle Q and does not use more than a polynomial number of cells neither of its work tape nor of its oracle tape. This oracle model is sometimes called *bounded*, as opposed to the unbounded model of Ladner and Lynch.[7] □

Definition 1.3.10.
For a complexity class $\mathcal{D}$ and $\mathcal{C} \in \{LOGSPACE, PTIME, PSPACE\}$, we write $\mathcal{C}^{\mathcal{D}}$ for $\bigcup_{Q \in \mathcal{D}} \mathcal{C}^{Q}$. □

Oracle complexity classes can give some evidence into the difficulty of proving the (in-)equality of two complexity classes. For instance, showing $\mathcal{C}_1 \equiv \mathcal{C}_2$ for two classes $\mathcal{C}_1, \mathcal{C}_2$ for which a result of the form $\mathcal{C}_1^{Q} \not\equiv \mathcal{C}_2^{Q}$ is known requires a method that does not relativize. A survey of results of this kind can be found in [7].

Like in complexity theory, where $\mathcal{C}^{Q}$ may depend on the specific representation of $\mathcal{C}$ and not only on the queries computable in $\mathcal{C}$, the same situation holds for logics: The expressive power of $\mathcal{L}(Q)$ may very well depend on the syntax

[7]For the relations between the bounded and unbounded model, for further models, and for further references, the reader is referred to [7].

of $\mathcal{L}$ and not only on the expressive power of $\mathcal{L}$. We will verify this for fixed point logics in section 4.

The analogy between quantifiers and oracles is expressed in the following theorems, which form the basis for our further investigation of the interaction between fixed point operators and quantifiers.

Theorem 1.3.11. **[15]** For any class of structures, Q, we have
$PTIME^Q \equiv_{\mathcal{O}} IFP(Q)$.

Theorem 1.3.12. **[36]** For any monotone class of structures, Q, we have
$PTIME^Q \equiv_{\mathcal{O}} LFP(Q)$.

We close this section with an analogous result for *PSPACE*. The proof is a variation of the proofs of the previous two theorems. We can base it on the proof of Lemma 1.2.14, where most of the techniques were provided.

Theorem 1.3.13. For any class of structures, Q, we have
$PSPACE^Q \equiv_{\mathcal{O}} PFP(Q)$.

Proof: $PSPACE^Q \geq PFP(Q)$ can be shown by a straightforward induction on $PFP(Q)$. For the Q–step, consider machines that compute the queries defined by the subformulas bound by Q. Applying these machines to all tuples (of the arity in which Q occurs), one can compute the structure which is defined by the quantified formulas. This structure has to be written onto the oracle tape and checked by an oracle query.
For the inclusion $PSPACE^Q \leq_{\mathcal{O}} PFP(Q)$ we can extend the proof of Lemma 1.2.14. The lemma referred to space bounds $f(n) = o(n^k)$ for a fixed k. Let us keep this assumption, and let us assume that the oracle tape, too, obeys to that space bound (taking the union over all $k \geq 1$ will yield the claim). Furthermore, we assume that Q is of vocabulary $\sigma_Q = \{S\}$ for an s–ary S. An encoding of a σ_Q–structure $\mathcal{B}$ has the form

$$1 \ldots 1\$\alpha,$$

where the first part (of length m, say) represents B and the second part, $\alpha \in \{0,1\}^*$, represents $S^{\mathcal{B}}$ with respect to some ordering on B (note that $\mathcal{B}$ itself may be unordered). To simplify the formulas to be given below, we assume that M has two oracle tapes, a *domain tape* for the representation of B, and a *structure tape* for α. We can further assume that M, before making any oracle queries, checks whether these tapes encode any structure $\mathcal{B}$. If this is the case, we assume that M moves its domain head to the last symbol 1. So each time M is in state s_{query}, it is guaranteed that there is a string $1 \ldots 1$ of some length m on the domain tape, a string α of length m^s on the structure tape, and the domain head is at position m.
A configuration is now a tuple $(q, p_1, p_2, p_3, p_4, v, \alpha)$, where p_3, and p_4 are the positions of the domain head, and the structure head respectively, α is the content of the structure tape, and the other components are as before. The

encoding of configurations can now be extended with the following lines:[8]

$P\langle 4, w\rangle\bar{0}$	The w-th bit of the domain-head position (in binary) is a 1
$P\langle 5, w\rangle\bar{0}$	The w-th bit of the structure-head position (in binary) is a 1
$P\langle 6, x_0\rangle x_1 \ldots x_{k-1}$	At position $(\overset{k}{x})_n (= x_0 n^{k-1} + \ldots + x_{k-1})$, there is a 1 in the string α.

Remember that the numbers of positions refer to an order $<^{\mathcal{A}}$ of an input $\mathcal{A} \in \mathcal{O}[\tau]$ for some τ with $<\in \tau$, here. Now, the formula φ that (among other things) initializes P in the proof of 1.2.14 has to take the oracle tapes into account. This can be done in the same way as for the other tapes.
The formulas φ_δ have to take into account the oracle query steps. Consider a configuration $(q, p_1, p_2, p_3, p_4, v, \alpha)$, where $q = s_{\text{query}}$.
Since the domain head is at the rightmost (i.e. m–th) position, and because of $m = o(n^k)$ (for $n := |A|$), there is a formula $\psi_{\text{Dom}}(\overset{k}{z}, P) \in \mathit{PFP}$ which holds exactly for the first m tuples of length k in $\mathcal{A}$ (with respect to the lexicographical order derived from $<^{\mathcal{A}}$) provided that $P \subset A^k$ encodes the above configuration. (See the proof of 1.2.14 for how to retrieve the information of the binary encodings of head positions.)
Furthermore, there is a formula $\psi_{\text{Str}}(\overline{z}_1, \ldots, \overline{z}_s, P) \in \mathit{PFP}$ (all tuples $\overline{z}_i$ of length k) such that the following holds: If P codes for the above configuration, $\mathcal{B}$ is the structure on the oracle tapes, and $\overline{a}_1, \ldots, \overline{a}_s$ are tuples over A with $\mathcal{A} \models \psi_{\text{Dom}}[\overline{a}_i, P]$ for each i, then $\mathcal{A} \models \psi_{\text{Str}}[\overline{a}_1, \ldots, \overline{a}_s]$ holds just in case $\mathcal{B} \models Sb_1, \ldots, b_s$, where b_i is the following tuple in B^k: Let $\overline{a}_i$ be the ν–th tuple in A^k. Then b_i is the ν–th element of B with respect to the order by which $\mathcal{B}$ is presented on the oracle tapes.
In fact, we can set

$$\psi_{\text{Str}}(\overline{z}_1, \ldots, \overline{z}_s, P) := \forall \overset{k}{x} \left(((\overline{z}_1)_n, \ldots, (\overline{z}_s)_n)_m = (\overset{k}{x})_n \rightarrow P\langle 6, x_0\rangle x_1 \ldots x_{k-1} \right).$$

Here, again, we used the notation for n– and m–ary expansions given before Lemma 1.2.7 (note that if we have $\psi_{\text{Dom}}(\overline{z}_i)$, then $(\overline{z}_i)_n < m$). Due to the restricted length of the structure tape, the value of $((\overline{z}_1)_n, \ldots, (\overline{z}_s)_n)_m$ is $o(n^k)$ and, therefore, can be described as $(\overset{k}{x})_n$, with restricted values for x_0 (so that the pairing function is defined). Using the arithmetical predicates of Lemma 1.2.7 and a similar technique to encode m–ary representations for varying m, one finds that ψ_{Str} can be chosen as a *PFP*–formula.
Now, the pair $\Phi := (\psi_{\text{Dom}}(\overline{z}, P), \psi_{\text{Str}}(\overline{z}_1, \ldots, \overline{z}_s, P))$ defines, for P as above, on $\mathcal{A}$, the structure $\mathcal{A}^\Phi = \mathcal{B}$. Hence $Q\overline{z}, \overline{z}_1, \ldots, \overline{z}_s; \Phi$ expresses that the answer to the oracle query is positive. Using this formula, one can extend the formulas φ_δ of the proof of 1.2.14(ii) so as to assert that in the case of a positive (negative) answer the successor configuration has the state s_{yes} (s_{no}). $\square$

[8] Clearly, the pairing function could be avoided in this proof. We kept it only to base this proof on the earlier one.

2 Complete problems

2.1 Quantifier free reductions

The notion of quantifier free reductions is closely related to that of Lindström logics. Let Q be a class of structures of signature $\sigma = \{R_1, \ldots, R_s\}$, let $r \geq 1$, and let $\Phi := (\psi(\overset{r}{x}), \psi_1(\overline{x_1}), \ldots, \psi_s(\overline{x_s}))$ be a tuple of quantifier free formulas of a vocabulary τ, such that $arity(\overline{x_i}) = r \cdot arity(R_i)$ for $1 \leq i \leq s$. Recall that on each τ–structure $\mathcal{A}$, Φ defines a σ–structure denoted by $\mathcal{A}^\Phi$ (see Def. 1.3.1).

Definition 2.1.1. Let $\tilde{Q}$ be a class of τ–structures and Q a class of σ–structures. The mapping $\mathcal{A} \mapsto \mathcal{A}^\Phi$ is called a *quantifier free reduction* (of arity r).[9] It is said to reduce $\tilde{Q}$ to Q if for each τ–structure $\mathcal{A}$, we have

$$\mathcal{A} \in \tilde{Q} \quad \text{iff}$$

$$\mathcal{A}^\Phi \text{ is defined (i.e. } \psi^{\mathcal{A}} \neq \emptyset) \text{ and } \mathcal{A}^\Phi \in Q.$$

Given a logic (or complexity class) $\mathcal{L}$, we say that Q is *$\mathcal{L}$–hard under quantifier free reductions* if for each τ and each class $\tilde{Q} \in \mathcal{L}[\tau]$, there exists a quantifier free reduction from $\tilde{Q}$ to Q.
Q is said to be *$\mathcal{L}$–complete* under quantifier free reductions if, in addition, Q is itself $\mathcal{L}$–definable (or computable in $\mathcal{L}$ respectively). □

Observe that a class of τ–structures $\tilde{Q}$ is reducible to Q in the above sense just in case it is definable by a formula of $FO(Q)$ of the form $Q\overline{x}; \Phi$ for a tuple Φ as above.

The following is straightforward to check.

Remark 2.1.2. Let ρ, σ, and τ be constant free vocabularies, and let Φ: Str$[\rho]$→Str$[\sigma]$, and Ψ: Str$[\sigma]$→Str$[\tau]$ be quantifier free reductions. Then there is a quantifier free tuple $\Theta \in FO[\rho]$ such that for all $\mathcal{A} \in$Str$[\rho]$, we have

$$\mathcal{A}^\Theta = (A^\Phi)^\Psi.$$

□

The above notions of hardness and completeness carry over to ordered structures if we assume $\tilde{Q} \subset \mathcal{O}[\tau]$, and replace 'for each τ–structure $\mathcal{A}$' with 'for each ordered τ–structure $\mathcal{A}$' in the last definition. Note that for this definition it is not necessary that Q itself be ordered. But if Q is $\mathcal{L}$–hard (respectively $\mathcal{L}$–complete) on ordered structures, then so is $Q^<$, since we can extend a quantifier free reduction with a (quantifer free) definition of the lexicographical order of r–tuples. Hence if we now state completeness–results for ordered structures, they can also be read from the point of view (which is taken by some authors) that Q must be ordered as well.

[9]Remember, our notation does not imply that in $\psi_i(\overline{x_i})$ all variables of the tuple $\overline{x_i}$ occur. So, the arity r is a property of the mapping associated with Φ above, and is not determined by Φ itself.

For the class of graphs that contain a Hamiltonian path, Q_{HAM}, Stewart has shown the following.

Theorem 2.1.3. **[35]** On ordered structures with constants for the first and last element, Q_{HAM} is NP–complete under quantifier free reductions. □

Dahlhaus has established an NP–completeness–result for Q_{HAM} under reductions that come close to those of Definition 2.1.1 in that they do not require an underlying ordering. But they require two constants which are assumed to be interpreted differently [8].

Next to Fagins result on Σ_1^1 (Theorem 1.1.3), this gives another logical description of NP, namely as a fragment of $FO(Q_{HAM})$. This raises the question what the expressive power of the *full* logic $FO(Q_{HAM})$ is on ordered structures. This has been answered by Gottlob in terms of oracle complexity classes.

Theorem 2.1.4. **[14]** $FO(Q_{HAM}) \equiv_{\mathcal{O}} LOGSPACE^{NP}$. □

The last theorem has been derived in [14] as a consequence of a considerably more general statement about complexity classes $\mathcal{C}$ (in place of NP) and $\mathcal{C}$–complete classes (in place of Q_{HAM}). We will address the expressive power of $FO(Q_{HAM})$ in Section 4, again. There we will also introduce other notions of reducibility.

Let us turn to arbitrary finite structures. Stewart raised the question whether there exists an analogue of Theorem 2.1.3 for arbitrary structures, i.e. a class Q that is NP–complete under quantifier free reductions that do not rely on an ordering, or on constants that are interpreted by distinct values. We will give an affirmative answer to this in 2.4 using quantifiers based on circuits. This method had already been used by Lovász and Gács, who also investigated NP–completeness, but under a slightly different type of reductions [30].

Surprisingly, modifications of this method yield complete problems for a variety of logics including not only NP, but also all other levels of the Polynomial Hierarchy, as well as fixed point logics, and their extensions with arbitrary quantifiers.

In Section 3, we will give a characterization of those logics for which there exist complete problems (under quantifier free reductions). This generalizes a theorem of Dawar (to be restated in 3.3) according to which there exists such a problem for $PTIME$ just in case $PTIME$ has any reasonable logical characterization at all.

2.2 The monotone circuit value problem

Circuits can be viewed as directed graphs the nodes of which, sometimes referred to as gates, carry out certain basic operations on the truth values TRUE and FALSE. Some nodes will have no other function than providing the fixed value TRUE as an input. Given this input of a circuit, we want to know whether or not one of the nodes, designated as an output node, produces the value TRUE. In later applications, there will either be exactly one output node, or all output nodes will yield the same result. First, we concentrate on the case where only conjunctions and disjunctions occur, that is on so called *monotone* circuits.

Definition 2.2.1. 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

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

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

$$\varphi_{CON}(x,Y) := Kx \wedge \forall y(Eyx \to Yy),\ \varphi_{DIS}(x,Y) := Dx \wedge \exists y(Eyx \wedge Yy).$$

□

Note that we did not make any restrictions on the underlying graph $(A; E)$ of a circuit $\mathcal{A}$. By monotonicity, nodes with value FALSE and not yet evaluated nodes can be treated in the same way (so FALSE does 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, in that it allows cycles. This will enable us to encode fixed point processes.

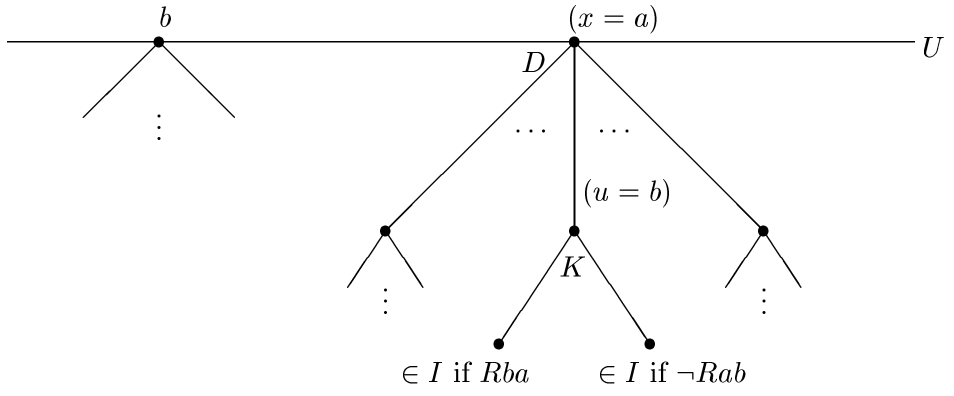
It is a well known technique to evaluate first order formulas by circuits. Here we use circuits to deal uniformly with the first order part χ of formulas of the form $[IFP_{x,X}\chi]t$. Before going into details we sketch the main ideas:

Let $\mathcal{A}$ be a τ–structure and $\chi(x) \in FO[\tau]$ be a formula in which negations (if any) occur only in front of atomic subformulas. We assume that each variable occurs only free or only bound (in the second case bound by only one quantifier).

Expanding the 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. Let $\sigma' = \sigma_{MC} \cup \{U\}$ for a unary U (it will represent a universe of an encoded structure). The following σ'–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 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 2.2.1, 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 χ are evaluated in the obvious way, and the leaves of $\mathcal{C}_a$ will correspond to atomic or negated atomic formulas.

For instance, in case $\chi = \exists u(Rux \wedge \neg Rxu)$, $\mathcal{E}$ looks as follows (disregarding the direction of arcs):



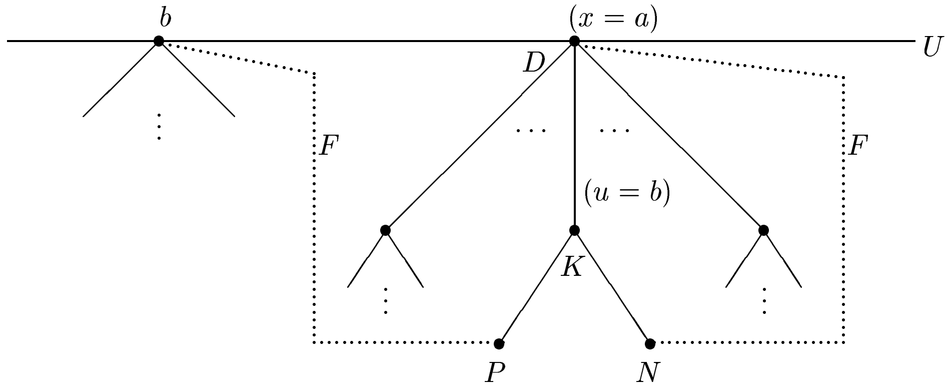
In general, the input I on the leaves is defined as follows: Following the path from a leaf λ to the root a of $\mathcal{C}_a$, one gets an assignment $\beta_\lambda : u_i \mapsto a_i$ for the variables $u_1, \ldots, u_m$ of χ (including $x \mapsto a$ if x occurs in χ). The leaf λ is being labeled with I if and only if the corresponding atomic formula is true in $(\mathcal{A}, \beta_\lambda)$.

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 finite structures, below. Hence, we would have the problem of defining levels of a circuit corresponding to the stages. This causes difficulties in unordered structures.

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 problems only for those leaves λ that correspond to a formula Xy or $\neg Xy$, where y is a variable which has been assigned a certain value $b = \beta_\lambda(y)$ (by the above procedure). In the following, we call those leaves *undefined*, since their input value is not a priori determined.

We identify the stages $X_j \subset A$ of the iteration 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 undefined leaves must be provided with information as to whether they correspond to a formula Xy or to $\neg Xy$, and, to which $b \in U$ 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 $b \in U$. As an example, let $\chi = \exists u(Xu \wedge \neg Xx)$. Then $\mathcal{E}(\chi, \mathcal{A})$ looks as follows:

$\mathcal{E}(\chi, \mathcal{A})$:



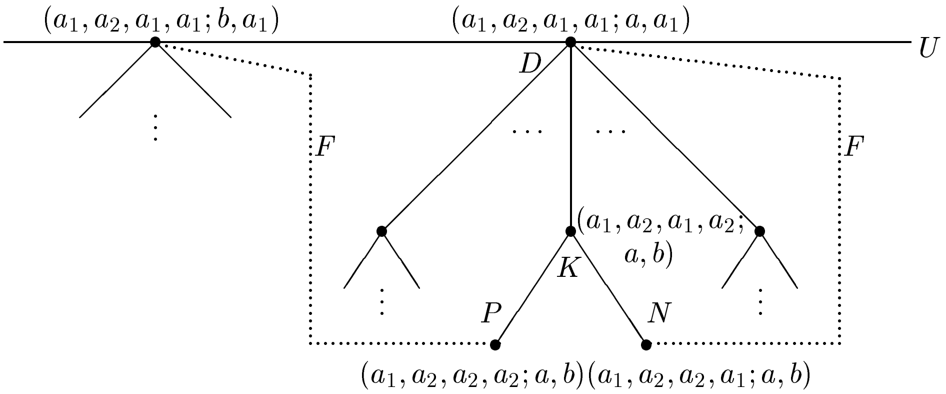
$\mathcal{E}(\chi, \mathcal{A})$ operates as follows: Starting from $X_0 := \emptyset$, it generates a sequence $(X_j)_{j\geq 0}$ of subsets of U. To compute X_{j+1} from X_j, it evaluates each tree as before, taking however for each leaf $\lambda \in P$ (resp. N) the value TRUE (resp. FALSE) if and only if $b \in X_j$ for the b with $F\lambda b$.

We will describe this evaluation in *IFP*, below. So there is a fixed *IFP*– formula that, for all χ, can describe the stages X_j. (This is possible because χ is encoded by $\mathcal{E}(\chi, \mathcal{A})$). Note that X_{j+1} is derived from X_j by *monotone* circuits (the trees in $\mathcal{E}(\chi, \mathcal{A})$). (Negation occurs only at leaves; at each round, X_j is being updated in U before the circuit computation is restarted.)

The next step is 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 1.1.12), we can transfer the question of whether an *IFP*$-$ formula is satisfied in a structure $\mathcal{A}$ to the question of whether $\mathcal{E}(\chi, \mathcal{A})$ yields output TRUE in a certain node that has to be specified.

A straightforward attempt is to obtain Φ_χ via equality types for the different isomorphism types of nodes. In the example of the above χ, let $r = 6$ be the arity of Φ_χ. The elements of $\mathcal{A}^{\Phi_\chi}$ are tuples $(a_1, \ldots, a_6)$, where we can use the equality type of (a_1, a_2, a_3, a_4) to encode the type of the node. There are four such types corresponding to the four subformulas of χ. The pair (a_5, a_6) 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 $(a_1, a_2, a_1, a_1; a, a_1)$ for the root of $\mathcal{C}_a$ in $\mathcal{E}$. For each $b \in A$ it has the predecessor $(a_1, a_2, a_1, a_2; a, b)$, which, in turn, has the predecessors $(a_1, a_2, a_2, a_2; a, b)$ and $(a_1, a_2, a_2, a_1; a, b)$. For this pair (a_1, a_2), the structure $\mathcal{E}(\chi, \mathcal{A})$, now denoted $\mathcal{E}(\chi, \mathcal{A})^{a_1, a_2}$, looks as follows:

$\mathcal{E}(\chi, \mathcal{A})^{a_1,a_2}$:



Two technical difficulties remain:

- 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 to the root, then there are many other tuples $\overline{c}'$ of the same equality type as $\overline{c}$, and one cannot distinguish between them in a quantifier free reduction. We avoided this difficulty above by fixing a_1 and a_2.

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 $\{\mathcal{E}(\chi, \mathcal{A})^{a_1,a_2} \mid a_1 \neq a_2\}$ of isomorphic copies of $\mathcal{E}(\chi, \mathcal{A})$ (and of some isolated points – corresponding to the tuples that do not match any of the specified equality types). Each copy works 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:

Theorem 2.2.2. Q_{MC} is complete for *IFP* with respect to quantifier free reductions. So every $\varphi \in \mathit{IFP}$ 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, $\mathit{IFP} \equiv FO(Q_{MC})$.

(We postpone the proof. First, we give a completeness result for the structures of type $\mathcal{E}(\chi, \mathcal{A})$.)

Q_{MC} is well known to be $PTIME$–complete under $LOGSPACE$–reductions. The above theorem establishes a completeness in a stronger sense. There are several other constructions of *IFP*–complete problems in the literature [9, 19, 21]. They are derived from strong normal form theorems for *IFP* [9, 19], or by a game theoretical approach [21], rather than from complexity theoretic examples of $PTIME$–complete problems (like Q_{MC}).

We make the above ideas precise:

Let τ be a vocabulary without constants [10] , $\chi \in FO[\tau]$, $\mathcal{A}$ a τ–structure and $\beta : V \to A$ an assignment. Here 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.

Definition 2.2.3. $\mathcal{C}(\chi, \mathcal{A}, \beta)$ is the following monotone circuit (defined by induction on χ):

- If χ is atomic or negated atomic, then $\mathcal{C}(\chi, \mathcal{A}, \beta)$ is a single node λ being both input and output, i.e. $O = \{\lambda\}, E = K = \emptyset, I = \{\lambda\}$ 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 $a \in A$ (the roots of the subtrees are no longer in O).
- Similarly, universal quantifiers, disjunctions and conjunctions can be treated.

$\square$

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 structure $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ checks whether $(\mathcal{A}, \beta) \models \chi[M]$ holds:

Definition 2.2.4. Let $\hat{\sigma}_{MC} = \sigma_{MC} \cup \{U, P, N, F\}$ for unary relation symbols U, P, N and a binary relation symbol F.
Then $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ 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 we let $\lambda \in N$ (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, by $\forall z_\nu$, say, 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 of the U–part). As in $\mathcal{C}(\chi, \mathcal{A}, \beta)$, O holds only for the root of the tree. $\square$

[10] The case of constants could easily be included but would make the definitions below more complex.

Lemma 2.2.5. Let $\chi, \mathcal{A}, \beta, M$ and $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ be as above and let

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

$$[IFP_{x,Y}\Big(Ix \vee \Big(Px \wedge \exists z(Fxz \wedge Xz)\Big) \vee \Big(Nx \wedge \exists z(Fxz \wedge \neg Xz)\Big) \vee$$

$$\Big(\exists y Eyx \wedge (\varphi_{CON}(x, Y) \vee \varphi_{DIS}(x, Y))\Big)\Big)]w.$$

Then, identifying A^s and U (and hence M with a subset of U), we have

$$(\mathcal{A}, \beta) \models \chi[M] \quad \text{iff} \quad \hat{\mathcal{C}}(\chi, \mathcal{A}, \beta) \models \exists w(Ow \wedge \hat{\varphi}_{TRUE}(w, X))[X := M].$$

Proof: This is easyily proved by induction on χ. In fact, it is a straightforward generalization of the notion of a circuit evaluating a first order formula. For the leaves that correspond to formulas in a second order variable the input values are defined by 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 2.2.6. 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 $\hat{\mathcal{C}}_{\overline{a}}$ (the relation O being dismissed), but all of them share $U^{\mathcal{E}}$ as their U–part.
- 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 $a_1 = \ldots = a_s$.

□

Note that the structure $\mathcal{E}(\chi, \mathcal{A})$ for $\chi = \exists u(Xu \wedge \neg Xx)$ as defined above is an example of a generalized circuit (for $s = 1$). 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 the next sections.

Theorem 2.2.7. Let $\hat{Q}_{MC} := Mod(\exists z(Oz \wedge [IFP_{w,X}(Uw \wedge \hat{\varphi}_{TRUE}(w, X))]z))$.
(i) For $\mathcal{A}, \chi, \beta$ as above we have

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

(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 2.2.5: By the lemma, for each $\overline{a} \in A^s$, we have
$(\mathcal{A}, \beta\frac{\overline{a}}{\overline{x}}) \models \chi[M]$ iff $\hat{\mathcal{C}}_{\overline{a}} \models \hat{\varphi}_{TRUE}[w_{\overline{a}}, M]$,
where $w_{\overline{a}}$ denotes the root of $\hat{\mathcal{C}}_{\overline{a}}$. So, 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, that is the stages $X_j \subset U$ are copies of the original stages $X_j \subset A^s$, whence the claim follows (here again we identify the elements of U with the tuples in A^s).
(ii) Given a formula in the normal form 1.1.12, $\varphi = \exists z[\mathit{IFP}_{\overline{x},X}\chi(\overline{x}, X)]z \ldots z$, where $\chi \in \mathit{FO}$ is of the form described above (negations pulled before atomic subformulas), we show that there is a tuple $\Phi_\chi = (\psi_O, \ldots, \psi_F)$ of quantifier free formulas such that $\mathcal{A}^{\Phi_\chi}$ is a collection of isomorphic copies of $\mathcal{E}(\chi, \mathcal{A}, \beta)$ and of 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 some isolated points.
Φ_χ can be obtained via equality types: Let m be the number of subformulas in χ. (So for the structures $\hat{\mathcal{C}}_{\overline{a}}$, m denotes the number of nodes in the tree part provided that we identify all inputs of the nodes which evaluate a quantifier.) Let d be the quantifier rank of χ. Choose $e \geq 1$ such that there are at least $m+1$ equality types of e–tuples over a set with two elements. 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 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 of length $s+d$ will encode the path to the root, consisting of the root itself and d choices for the quantified variables.
To make this specific, set

$$\psi_U(\overline{x}) := x_1 \neq x_2 \wedge x_1 = x_3 \wedge x_1 = x_4 \wedge \ldots \wedge x_1 = x_e$$

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

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

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

where the first tuple $w_1, w_2, \ldots, w_1$ of length e is to be understood as the equality type coding U. The last d repetitions of w_1 simply fill up $\overset{s}{\overline{a}}$ to an $s+d$–tuple. Similarly, one can assign equality types to the other nodes in the trees. 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 bound variables of χ.
We illustrate this procedure by the above example (at each step we explain how to deal with the general case). We studied $\chi = \exists u(Xu \wedge \neg Xx)$. 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. Again, 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. 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

$$\psi_D := \psi_U.$$

(In general, ψ_D is a disjunction over the equality types chosen to encode the nodes in D.)
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 each $b \in A$). They represent the subformula $Xu \wedge \neg Xx$ of χ, with values $x = a, u = b$. Since they are the only conjunctions, we set

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

(In general, ψ_K is a disjunction over the equality types chosen to encode the nodes in K.)
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 (in the general case, take a disjunction over all types of edges). 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, that is which subformula they represent. In this example, this amounts to saying that the first one is a conjunction, the second a disjunction, as this determines the types 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.
Observe that we did not need a domain relativizing formula $\psi(\overline{x})$ (compare Definition 2.1.1). We have isolated points in $\mathcal{A}^{\Phi_\chi}$. They do not belong to any copy of $\mathcal{E}(\chi, \mathcal{A}, \beta)$ but have no influence on the circuits computation, and hence are irrelevant for the membership of $\mathcal{A}^{\Phi_\chi}$ in $\hat{Q}_{MC}$.
(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 2.2.2: We make use of the fact that *IFP* $\equiv$ *LFP* [20], and of the stronger normal form associated to *LFP* (see 1.1.12). 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 will also dispense with U and replace F–arcs by usual E–arcs. The function of these new E–arcs is to pass the value TRUE from roots to (undefined) leaves. Hence we let the undefined leaves operate as disjunctions (with only one input). More formally:

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 Φ_χ^* to the class Q_{MC} by setting

$$\Phi_\chi^* := (\psi_O, \psi_E^*, \psi_K, \psi_D^*, \psi_I), \text{ where}$$

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

$$\psi_D^* := \psi_D \vee \psi_P.$$

(Note that ψ_P defines exactly the undefined leaves of $\mathcal{E}(\chi, \mathcal{A})$ due to the choice of χ.)

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 $\mathcal{E} \models Fvu$, and with D extended to all (former) leaves λ with $\mathcal{E} \models P\lambda$, then $\mathcal{A}^{\Phi_\chi^*}$ consists of a collection of copies of $\mathcal{E}^*$, and of some isolated points. As before, we have

$$\mathcal{A}^{\Phi_\chi^*} \in Q_{MC} \text{ just in case } \mathcal{E}^* \in Q_{MC}.$$

We claim that

$$\mathcal{E}^* \in Q_{MC} \text{ if and only if } \mathcal{E} \in \hat{Q}_{MC},$$

which concludes the proof.

To see this, consider the formulas defining Q_{MC} (Definition 2.2.1) and $\hat{Q}_{MC}$ (Theorem 2.2.7). Write $Y_l (l \geq 0)$ for the stages of

$$[\mathit{IFP}_{x,Y}(Ix \vee (\exists y Eyx \wedge (\varphi_{CON}(x, Y) \vee \varphi_{DIS}(x, Y))))]$$

in $\mathcal{E}^*$, and $X_j (j \geq 0)$ for those of

$$[\mathit{IFP}_{w,X}(Uw \wedge \hat{\varphi}_{TRUE}(w, X))]$$

in $\mathcal{E}$. Remember, the X_j can be identified with the (s–ary) stages induced by χ in $\mathcal{A}$. We show that X_∞ and Y_∞ agree on U (for $\mathcal{E}^*$ this means the copy of $U^{\mathcal{E}}$). Then the definitions of Q_{MC} and $\hat{Q}_{MC}$ (saying that $O \cap X_\infty \neq \emptyset$, and $O \cap Y_\infty \neq \emptyset$ respectively) yield the equivalence we aim at, namely $\mathcal{E}^* \in Q_{MC}$ iff $\mathcal{E} \in \hat{Q}_{MC}$. It suffices to show the following:
Claim 1: $X_\infty \subset Y_\infty$.
Claim 2: Let $l \geq 0$ and, for a node x, assume $x \in Y_l$. Then, the (sub–)tree with root x evaluates to TRUE if X_l defines the values of the undefined leaves. (For $x \in U$ this conclusion amounts to $x \in X_{l+1}$.)
Proof of Claim 1: Let h denote the height of the tree-like circuits constructed from χ (here we start counting from the leaves which we assume to be of height 1). By induction on j we show:

$$X_j \subset Y_{hj} \text{ for all } j \geq 0$$

(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). In fact, $w \in X_{j+1}$ means that w is a root to which TRUE is assigned (in one evaluation round of the tree with root w) if the truth values of undefined leaves in P are defined by X_j (which means, a leaf $\lambda \in P$ is assigned TRUE just in case the root w' with $F\lambda w'$ is an element of X_j). Assume $X_j \subset Y_{hj}$. Let now $Y^*_{hj} := Y_{hj} \cap U$ define the input of the tree with root w. So this time, a leaf $\lambda \in P$ is assigned TRUE just in case $w' \in Y^*_{hj}$ for the root w' with $F\lambda w'$. Let us write TRUE(Y^*_{hj}) for the set of theses leaves.

We find that w is still evaluated to TRUE, and we have

$$X_j \subset Y^*_{hj} \subset Y_{hj}.$$

Now, apply h steps of the Y–iteration to Y^*_{hj}. This results in a subset Y^* of Y_{hj+h} since $\mathcal{E}^*$ is a monotone circuit.

The first step increases Y^*_{hj} by TRUE(Y^*_{hj}), because each of the nodes in TRUE(Y^*_{hj}) is E–successor of a node in Y^*_{hj}, and operates as a disjunction. The fact that w evaluates to TRUE means that in one of the following $h-1$ steps, w comes into the fixed point predicate, that is $w \in Y^*$. Hence $w \in Y_{h(j+1)}$.
Proof of Claim 2: By induction on l. Assume $x \in Y_{l+1}$. The interesting case is $x \in D$ (the other cases are left to the reader).

If x is an undefined leaf in $\mathcal{E}$, then there is a unique x' with $\mathcal{E}^* \models Ex'x$, and we have $x' \in Y_l$. By induction, $x' \in X_{l+1}$. So, under X_{l+1}, the leaf x is assigned the input value TRUE (remember we have $\mathcal{E} \models Fxx'$).

The case of x having input nodes (in $\mathcal{E}$) is straightforward, because for this case, the transition from Y_l to Y_{l+1} amounts to the usual evaluation of a disjunction node. □

We made essential use of the fact *IFP* $\equiv$ *LFP*. In turn, this fact follows from Theorem 2.2.2, as one can replace *IFP* in Definition 2.2.1 by *LFP* and hence, rewrite the $FO(Q_{MC})$–formulas as *LFP*–formulas.

Note however that if one does not use *IFP* $\equiv$ *LFP*, the proof of Theorem 2.2.2 can still be interpreted as a proof[11] of the completeness of Q_{MC} for *LFP*.

We close with an observation on the nesting of quantifiers. Our completeness result (or those of the literature cited above) implies that in $FO(Q_{MC})$ no nesting of Q_{MC} is necessary.

As we now show, the phenomenon that a fixed nesting suffices, does not depend on the choice of the quantifier. Let $FO(Q_k)$ denote the formulas of $FO(Q)$ with up to k nested quantifiers Q (of any arity).

Theorem 2.2.8. 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 2.2.1, and let $\rho \in FO(Q)$ be an equivalent formula. Choose k with $\rho \in FO(Q_k)$. If ψ is

[11] The choice of *IFP* and the intermediate construction of $\hat{Q}_{MC}$ made the proof of 2.2.2 more complicated than a direct approach via *LFP* and Q_{MC}. But we will reuse the provided techniques later.

an arbitrary $FO(Q)$–formula, then ψ is equivalent to $Q_{MC}\bar{v};\Phi$ 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 of all first order quantifiers ($\exists u$ becomes $\exists \overset{r}{u}$ etc.), and of the IFP–operator, and by substitution of the circuit symbols by the formulas in Φ. Applying the same procedure to ρ yields a $\rho\prime \in FO(Q_k)$ which is equivalent to $\varphi\prime$ and thus to ψ. $\square$

Remark 2.2.9. The proof of Theorem 2.2.8 also applies to all logics $\mathcal{L}$ of the form $FO(Q)$, that is to say, 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 $FO(Q')$. $\square$

2.3 Enhancing circuits with oracles

Having defined the generalized circuits (Definition 2.2.6), we are going to enrich them by a new sort of nodes for additional quantifiers. These so called Q–nodes evaluate an additional quantifier Q of some signature σ_Q. They will check whether their input nodes encode 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 let $\hat{\sigma}_C^Q := \hat{\sigma}_{MC} \cup \{R, R_Q\}$. R_Q will denote the set of Q–nodes, and R will be used to distinguish between nodes that determine the *domain* of an encoded structure, 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 Definition 2.2.4 of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ to the case of $\chi \in FO(Q)$. We consider only the Q–step, and the negations.

Definition 2.3.1. For $\chi \in FO(Q)$, the structure $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ is inductively defined as follows:

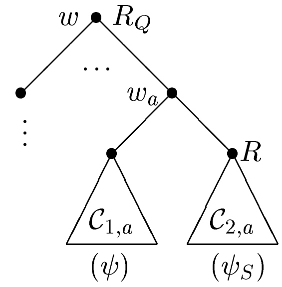
- If $\chi = Qx, y; \psi(x), \psi_S(y)$, we take for each $a \in A$ a circuit

$$\mathcal{C}_{1,a} := \hat{\mathcal{C}}(\psi, \mathcal{A}, \beta\frac{a}{x})$$

  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, finally, let all the w_a be predecessors of a new node $w \in R_Q$, which is the root of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$.



This circuit can check χ in the following way: 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 $\mathcal{D} \in Q$ (and, by construction, that is the case, if and only if $(\mathcal{A}, \beta) \models \chi$). Similarly, if $\chi = Q\ \overset{r}{x}, \overset{r}{y}; \psi(\overset{r}{x}), \psi_S(\overset{r}{y})$, we take $|A|^r$ many predecessors of w, each representing an r–tuple.

- Negations are represented by N–nodes within the circuits. They flip their (unique) input bit. (Hence we no longer have leaves marked with N; instead, negated fixed point atoms are represented by a P–node connected to an N–node.)

□

We aim at 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. Therefore, we have another look at the class of satisfied circuits.

In Definition 2.2.1 we could deal with nodes of value FALSE and not yet evaluated nodes in the same way. The situation changes drastically if negation nodes (in the sense of the last definition) are admitted: for arbitrary graphs it is not even clear how to evaluate the circuits as cycles may occur.

In the case of tree-like circuits, 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 $\{z \mid Y_j vz\}$ are intended to contain the nodes z with value TRUE, and FALSE respectively, after j steps of the evaluation. We rewrite the formulas for circuit evaluation in this framework. In addition to the earlier case, we include formulas for the negations, and a formula φ_{EVAL} that checks whether all predecessors of a given node have already been evaluated. If u and v are considered as parameters, the following formulas depend on x and Y.

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

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

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

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

The following formula says that w is evaluated to TRUE.

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

A tree-like circuit (the root marked as output) is satisfied if it is in the following class:

Definition 2.3.2. $Q_C := \mathrm{Mod}(\exists uvw(u \neq v \wedge Ow \wedge \varphi^*_{TRUE}(u,v,w)))$. □

We will not make use of the class Q_C itself. Rather, we extend its definition with the following clauses for Q:

$$\varphi_{Q+}(x) := R_Q x \wedge Q y_1, y_2; \psi_1, \psi_2$$

$$\varphi_{Q-}(x) := R_Q x \wedge \neg Q y_1, y_2; \psi_1, \psi_2, \text{ where}$$

$$\psi_1(y_1) := Ey_1 x \wedge \exists y(Eyy_1 \wedge \neg Ry \wedge Yuy),$$

$$\psi_2(y_2) := \exists y(Eyy_2 \wedge Ry \wedge Yuy).$$

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

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

$$[\mathit{IFP}_{z,x,Y}\Big(z = u \wedge Ix\Big) \vee \Big(z = v \wedge \neg Px \wedge \neg\exists y Eyx \wedge \neg Ix\Big) \vee$$

$$\Big(z = u \wedge Px \wedge \exists y(Fxy \wedge Xy)\Big) \vee \Big(z = v \wedge Px \wedge \exists y(Fxy \wedge \neg Xy)\Big)\vee$$

$$\Big(\varphi_{EVAL}(x) \wedge \Big(z = u \wedge (\varphi_{CON+} \vee \varphi_{DIS+} \vee \varphi_{NEG+} \vee \varphi_{Q+})\Big)\Big)\vee$$

$$\Big(\varphi_{EVAL}(x) \wedge \Big(z = v \wedge (\varphi_{CON-} \vee \varphi_{DIS-} \vee \varphi_{NEG-} \vee \varphi_{Q-})\Big)\Big)]uw.$$

This formula augments φ^*_{TRUE} by the clauses for Q. Hence it describes exactly the evaluation process in $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$ (the version for $\chi \in FO(Q)$). So we get:

Lemma 2.3.3. For $M \subset A^s$ we have $(\mathcal{A}, \beta) \models \chi[M]$ if and only if $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta) \models \exists uvw(u \neq v \wedge Ow \wedge \hat{\varphi}^Q_{TRUE}(u,v,w,X)[X := M]$.

Proof: Like Lemma 2.2.5, this is proved by induction over $\chi \in FO(Q)$. The Q–step has been treated in the definition of $\hat{\mathcal{C}}(\chi, \mathcal{A}, \beta)$, already. The negation step is trivial. □

Remember that M is understood as an s–ary relation on the left side of the last lemma, and as a unary relation 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 $\hat{\varphi}^Q_{TRUE}$, although the circuits can evaluate Q-queries of any arity. Our results on quantifiers for $IFP(Q)$, to be given below, do *not* imply collapse of the arity hierarchy connected with Q since we always admit quantifier free reductions of arbitrary arity (that is reductions to vectorized versions of the circuit quantifiers).

Definition 2.3.4. Let $\hat{Q}^Q_C$ be the class of models of

$$\exists z_0(Oz_0 \wedge \exists uv(u \neq v \wedge [IFP_{w,X}Uw \wedge \hat{\varphi}^Q_{TRUE}(u,v,w,X)]z_0)).$$

□

Observe that in $\hat{\varphi}^Q_{TRUE}$ there are no nested applications of Q. Coding the two nested IFP–operators of the last definition into a single one (cf. [22]), we will obtain a formula in a normal form.

Definition 2.3.5.
(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 2.3.6. 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 [18], the proof of Fact 1.1.12 can be carried over to $IFP(Q)$ since the well known *transitivity lemma* [22] does not depend on the logic from where occurring subformulas are taken.

Lemma 2.3.7. 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 $\models \varphi \leftrightarrow \varphi'$.
$PFP(Q) \equiv PFP_1(Q)$ (witnessed by formulas of the analogous type). □

Having the weak normal form available, we can proceed exactly as for Theorem 2.2.7(ii), using Lemma 2.3.3 in place of Lemma 2.2.5, and get:

Theorem 2.3.8.
$\hat{Q}^Q_C$ is complete for $IFP(Q)$ with respect to quantifier free reductions.
In particular, $IFP(Q) \equiv FO(\hat{Q}^Q_C)$.
Replacing IFP with PFP in the definition of $\hat{Q}^Q_C$ yields a problem complete for $PFP(Q)$. □

Observe that this provides a Lindström logic, that is one of the form $FO(Q')$, for $PTIME^Q$ (on ordered structures), as this oracle complexity class is captured by $IFP(Q)$ (similarly for $PSPACE^Q$ if the oracle tape is subject to the polynomial space bound). But it is noteworthy that Theorem 2.3.8 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 [14] 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 Section 4. 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 first order definable from these two classes.

For monotone classes Q, Hella has constructed problems complete for $LFP(Q)$ [21]. His result can also be restated by our circuit approach: working with additional nodes for the dual quantifier[12] of Q, one can eliminate negation nodes as in the proof of Theorem 2.2.2. Then one could extend Q_{MC} with quantifier nodes so as to obtain a complete problem for $LFP(Q)$. Rather than carrying out this construction, we observe that one can use the above quantifiers for $IFP(Q)$ also for $LFP(Q)$, because we have:

Remark 2.3.9. Let Q be monotone. Then $IFP(Q) \equiv LFP(Q)$.

The proof is obtained from the proof of $IFP \equiv LFP$, noting that it does not depend on the type of occuring subformulas as long as the least fixed point operators are well defined (which is the case for a monotone Q, see 1.3.2). □

As a corollary of our completeness results, we can sharpen the above weak normal forms.

Theorem 2.3.10. Normal Form.
(i) $IFP(Q) \equiv IFP_1(Q_1)$.
(ii) $PFP(Q) \equiv PFP_1(Q_1)$.
(iii) $LFP(Q) \equiv LFP_1(Q_1)$ for each monotone Q.

Proof: (i) 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 2.3.6 and vectorize all operators to the arity r of Φ, that is, replace for instance $\exists u$ by $\exists \overset{r}{u}$, multiply the arities of the fixed point variable, of the Q–operators, and of the symbols in $\hat{\sigma}_C^Q$ by r. Then replace those symbols by the defining formulas in Φ.
(ii) is obtained analogously.
(iii) can also be proved in this way. We did not exhibit a complete problem for this case, but we have argued for its existence above. □

For ordered structures, the above normal form for $IFP(Q)$ has already been

[12]For a class of graphs, for instance, the dual quantifier is defined as $\overline{Q} := \{(A, A^2 \setminus E) | (A, E) \notin Q\}$.

obtained via the equivalence to oracle classes [15].

2.4 Quantifiers for the polynomial hierarchy

Leaving the area of fixed point logics, we give another example of obtaining complete problems via circuits. By modifying the construction of 2.2, 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 essentially treated by Lovász and Gács in [30], though not with quantifier free reductions in the sense of Definition 2.1.1).

Theorem 2.4.1. 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 the previous sections. 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 negations within the circuits), or they are undefined, that is corresponding to a subformula of type $X \overset{s}{z}$.
- The roots of these circuits constitute the O–part.
- If, within one circuit, there occur two undefined leaves both of which evaluate[13] $X \overset{s}{a}$ 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 set of 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

$$\varphi_{SAT}(Y) := \forall x((Ix \rightarrow Yx) \wedge$$

[13] As earlier, a path from the leaf (for $X \overset{s}{z}$) to its root yields the values for the variables in $\overset{s}{z}$.

$$(Nx \to (Yx \leftrightarrow \exists y(Eyx \wedge \neg Yy)))\wedge$$
$$(Dx \to (Yx \leftrightarrow \exists y(Eyx \wedge Yy)))\wedge$$
$$(Kx \to (Yx \leftrightarrow \forall y(Eyx \to Yy)))).$$

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. All roots are in Y, since the circuits evaluate to TRUE.

For the converse, let Y be a subset of $\mathcal{A}^{\Phi_\chi}$ witnessing $\mathcal{A}^{\Phi_\chi} \in Q_{SAT}$. The restriction of Y to the undefined leaves defines a subset M of A^s as follows. Set

$$M := \{\overset{s}{a} |\text{ there is a leaf } v \in Y \text{ that evaluates } X\overset{s}{a}\}.$$

Let $\overset{s}{a} \in M$. The formula φ_{CONS} asserts that *all* leaves v' which evaluate $X\overset{s}{a}$ are in Y. So, under the identification of $\overset{s}{a}$ with each leaf that evaluates $X\overset{s}{a}$, we find that M and Y agree on the leaves. Consequently, $\mathcal{A}^{\Phi_\chi} \models \varphi_{SAT}(Y)$ implies $\mathcal{A} \models \psi[M]$.

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 augment σ_{SAT} by k new monadic relations $P_1, \ldots, P_k$. They correspond to the k types of undefined input nodes (evaluating atoms in $X_1, \ldots, X_k$). Then, we can define a class Q^k_{SAT} similarly to Q_{SAT}, using however a prefix $\exists Y_1 \forall Y_2 \exists Y_3 \ldots \exists (or \forall) Y_k$ of monadic variables, and a first order formula that asserts the following: All Y_i are consistent with R; Y_1 describes the circuit evaluation provided that the inputs of leaves in P_i are defined by Y_i for $1 \leq i \leq k$; the circuits evaluate to TRUE. □

Theorem 2.4.2. 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: (i)$\rightarrow$(iv) follows from the last theorem. (iv)$\rightarrow$(ii) is clear because SO is a regular logic.
(ii)$\rightarrow$(i): Assume $SO \equiv FO(Q)$. Let $Q \in \Sigma^1_i$. Then, as IFP-operators can be described by an existential second order variable, we have $IFP_1(Q_1) \leq \Sigma^1_{i+1}$ and, by the normal form theorem 2.3.10, $SO \leq IFP(Q) \leq IFP_1(Q_1) \leq \Sigma^1_{i+1}$. (For $i = 1$ this conclusion has already been observed by Enderton.)
(iv)$\rightarrow$(iii) is clear, and (iii)$\rightarrow$(i) is well known (see, for instance [34]). $\square$

2.5 Monotone quantifiers

Let Q be a quantifier. We are going to define a monotone class Q^{mon} such that Q and Q^{mon} are first order definable from each other. Hence for many investigations on quantifiers, one can restrict to the monotone case in principle. In particular in Section 4 we will be able to translate our results about extensions of IFP into results on extensions of LFP.

We get Q^{mon} by introducing new relation symbols for the complements of the relations in $\sigma = \{R_1 \dots R_s\}$ (the vocabulary of Q).

Definition 2.5.1. For each $R_i \in \sigma$ let R_i^- be a relation symbol of the same arity as R_i. Let $\sigma^- := \sigma \cup \{R_i^- \mid 1 \leq i \leq s\}$. In the following $FO[\sigma^-]$–formulas, $\overline{x}_i$ denotes a tuple of distinct variables that matches the arity of R_i.

$$\varphi_\cap := \bigvee_{1\leq i\leq s} \exists \overline{x}_i (R_i\overline{x}_i \wedge R_i^-\overline{x}_i),$$

$$\varphi_{PARTITION} := \bigwedge_{1\leq i\leq s} \forall \overline{x}_i (R_i\overline{x}_i \leftrightarrow \neg R_i^-\overline{x}_i),$$

$$Q^{mon} := Mod(\varphi_\cap \vee (\varphi_{PARTITION} \wedge Q\overline{x}_1, \dots, \overline{x}_s; \Phi_\sigma)), \text{ where}$$
$$\Phi_\sigma := R_1\overline{x}_1, \dots, R_s\overline{x}_s.$$

$\square$

Theorem 2.5.2.
(i) Q^{mon} is monotone.
(ii) Q and Q^{mon} are first order definable from each other, that is we have

$$Q \in FO(Q^{mon}) \text{ and } Q^{mon} \in FO(Q).$$

(iii) Similarly, $Q^{r,P} \in FO(Q^{mon})$, $(Q^{mon})^{r,P} \in FO(Q)$ (see Def. 1.3.4).

Proof: (i) Let $\mathcal{A} \in Q^{mon}$ and consider a σ^-–structure $\mathcal{A}'$ with $A' = A$ and $R_i \subset R_i'$ for all i. We have to show $\mathcal{A}' \in Q^{mon}$. If $\mathcal{A} \models \varphi_\cap$, then also $\mathcal{A}' \models \varphi_\cap$. If $\mathcal{A} \models \neg\varphi_\cap$ and $\mathcal{A}' \models \neg\varphi_\cap$, then we have $\mathcal{A} \models \varphi_{PARTITION}$ and the relations cannot have been properly extended. Hence $\mathcal{A}' = \mathcal{A}$ and the claim follows.
(ii) Clearly, Q^{mon} is definable in $FO(Q)$. On the other hand, we show that

$$Q = Mod(Q^{mon}\overline{x}_1, \dots, \overline{x}_s, \overline{x}_1, \dots, \overline{x}_s; (\Phi_\sigma, \neg\Phi_\sigma)),$$

where $\neg\Phi_\sigma$ is derived from Φ_σ by negating each formula in that tuple.
To see this, note first that a σ^-–structure $\mathcal{A}$ which is a model of $\varphi_{PARTITION}$ belongs to Q^{mon} if and only if its σ–reduct belongs to Q. Now, let $\mathcal{B}$ be a σ–structure, and set $\mathcal{A} := \mathcal{B}^{\Phi_\sigma, \neg\Phi_\sigma}$. Then $\mathcal{A} \models \varphi_{PARTITION}$ and $\mathcal{B}$ is the σ–reduct of $\mathcal{A}$. Hence $\mathcal{A} \in Q^{mon}$ iff $\mathcal{B} \in Q$, concluding the proof.
(iii) can be checked straightforward now. □

Corollary 2.5.3. If Q is complete for a logic $\mathcal{L}$ under quantifier free reductions, then so is Q^{mon}.

Proof: By the proof of (ii) above, a formula $Q\bar{x}; \Phi$ holds in a structure $\mathcal{A}$ just in case

$$(\mathcal{A}^{\Phi})^{\Phi_\sigma, \neg\Phi_\sigma} \in Q^{mon}.$$

Remark 2.1.2 yields a quantifier free reduction Θ such that $\mathcal{A} \models Q\bar{x}; \Phi$ iff $\mathcal{A} \models Q^{mon}\bar{v}; \Theta$. □

3 Logics that define their own semantics

Throughout this section, let $\mathcal{L}$ denote a semiregular logic. We are only interested in 0–ary queries here. So all occurring $\mathcal{L}$–formulas are sentences unless otherwise stated.

Deviating from our general conventions, we will restrict ourselves in this section to vocabularies without constants (not only for the case of quantifiers as in the last section). For all constructions it will be easy to see how the case of constants (and hence that of arbitrary k–ary queries) can be treated.

Graphs are understood as structures of the type $(V; E)$ with $E \subset V^2$. So the edges are directed and may be loops. For $u, v \in V$, an undirected edge between u and v means the pair of edges Euv, and Evu. Undirected paths are (finite) sequences of pairwise distinct vertices, so that each subsequent pair is connected by an undirected edge.

3.1 Defining semantics within a logic

From an abstract point of view, the completeness results of the previous section can be described as follows. Given a formula φ of some logic $\mathcal{L}$, we could find a quantifier free reduction Φ, which, on a structure $\mathcal{A}$, operated in the follwing way: essentially, Φ merged a description of φ in an encoding of $\mathcal{A}$. Each time, the logic $\mathcal{L}$ turned out to be powerful enough to say whether the encoded $\mathcal{A}$–part was a model of the encoded φ–part. For instance, in the case of *IFP*, the formulas were completely determined by their first order part, since they were in normal form. This first order part was then encoded as a circuit, and the structure $\mathcal{A}$ was encoded into the leaves.

Generalizing this idea, we could take any straightforward encoding of φ as a structure, as a graph $G(\varphi)$, say, and merge $\mathcal{A}$ and $G(\varphi)$ in the simplest way into a structure by forming the disjoint union $\mathcal{U}^{G(\varphi),\mathcal{A}} := G(\varphi) \cup \mathcal{A}$. Next, we would have to find an $\mathcal{L}$–formula φ_{SAT} such that for all φ,

$$\mathcal{U}^{G(\varphi),\mathcal{A}} \models \varphi_{SAT} \quad \text{iff} \quad \mathcal{A} \models \varphi.$$

Then, Q:=Mod(φ_{SAT}) is a candidate for an $\mathcal{L}$–complete problem (under quantifier free reductions). All we would have to show is $\mathcal{L}$–hardness. To this end, only the mapping $\mathcal{A} \mapsto \mathcal{U}^{G(\varphi),\mathcal{A}}$ has to be described by a quantifier free reduction (for each fixed φ).

What we have just described amounts to a *definition of the semantics of $\mathcal{L}$ within $\mathcal{L}$ itself.* In this section, we show that this idea of constructing complete problems always applies provided that $\mathcal{L}$ has any complete problems at all.

There is also a computational counterpart of self definability: if a machine M_{SAT} accepts the class of models of φ_{SAT}, then M_{SAT} is a *universal machine* for the class of queries in $\mathcal{L}$. Such machines have been used by Dawar for his theorem stating that there is a problem complete for $PTIME$ under quantifier free reductions provided there is any (effective) logic for $PTIME$ at all. Essentially, his proof first establishes a universal machine, from which in a second step the complete problem is derived. We will restate this approach in terms of self definability in 3.3.

In [34] it is observed that complexity classes which have a universal machine (so called *syntactical* classes) usually have complete problems (under $LOGSPACE$– or $PTIME$–reductions, say). The Theorem on self definability, to be stated in 3.2.3, can be seen as a model theoretic formulation of that observation.

In logic, self definability, or, more generally, definability of the semantics of one logic within another one, is a familiar concept. We have encountered it in the diagonalization principle 1.2.15, already. The concept exists in various forms. For instance, *truth adequacy* has been used to study the properties of logics in a set theoretical framework (for an overview, see [38]).

Those examples heavily rely on a notion of definability of semantics that is designed to admit diagonalization. Our approach is different: Since in a structure $\mathcal{U}^{G(\varphi),\mathcal{A}}$ the two parts $G(\varphi)$ and $\mathcal{A}$ are separated, there is no ambiguity between structures and formulas. So, in principle, there is nothing wrong with an $\mathcal{L}$–formula defining the class $\{\mathcal{U}^{G(\neg\varphi),G(\varphi)} | \varphi \in \mathcal{L}\}$. Hence this straightforward attempt to introduce a class of "formulas that do not satisfy themselves" fails.

Despite this lack of diagonalization there is another way to disprove self definability. As our notion of self definability will turn out to be equivalent to the existence of complete problems (3.2.3), one can interprete nesting hierarchies for quantifiers as failure of self definability. Such examples have been established in [16] and will be stated in 3.3.

We are going to put the idea of semantics definitions in a precise form. For the following reasons, the straightforward idea of taking the disjoint union of $G(\varphi)$ and $\mathcal{A}$ for $\mathcal{U}(= \mathcal{U}^{G(\varphi),\mathcal{A}})$ has to be modified slightly.

- The signature τ of $\mathcal{A}$ can vary, whereas the formula φ_{SAT} (expressing $\mathcal{A} \models \varphi$ in $\mathcal{U}$) has to be of a fixed signature. In order to deal with varying signatures τ, we will put copies of the sets $A, A^2, A^3, \ldots, A^M$ (for some $M \geq \max\{s | S \in \tau, S \text{ is of arity } s\}$) into $\mathcal{U}$ rather than a copy of $\mathcal{A}$ itself. Additionally, a graph G_τ encoding τ comes into $\mathcal{U}$. Then we can designate one symbol R to relate points $w \in G_\tau$ (representing a relation symbol of τ) to points t in the copies of the sets A^i. So Rtw will indicate that in $\mathcal{A}$ the tuple t belongs to the relation represented by w.

- The concept of self definability becomes more robust, if we do not confine the length M of tuples over A to the maximum arity of symbols in τ. Instead, we associate with each formula φ a bound $M(\varphi)$ on the length of tuples. This bound may exceed the arities of symbols occuring in φ. On the computational side, this idea corresponds to a polynomial blow up of the input, which facilitates the task of a universal machine.

Definition 3.1.1. Let G_τ be the following graph representation of τ: if $\tau = \{S_1, \ldots, S_m\}$ (the symbols S_j ordered by their arities s_j), then G_τ consists of a directed path of m elements $w_1, \ldots, w_m$, where each w_j is connected to an undirected path of length s_j. We will say that w_j *represents the symbol* S_j. We will refer to s_m as $\text{arity}(\tau)$. □

Assume there are several symbols of the same arity in τ. The definition of G_τ does not depend on the order in which the symbols in τ are enumerated. This is no longer true for the following definition where τ–structures are encoded. Without explicitly mentioning it below, we assume that for each $r \geq 1$ the symbols of arity r in τ are taken from a list $\{S_{r,1}, S_{r,2}, \ldots\}$. For the definition of G_τ above it is assumed that in the list $\{S_1, \ldots, S_m\}$ the symbol $S_{r,i}$ appears before $S_{r,j}$ if $i < j$. In this setting, the structure $\mathcal{U}$ is well defined below.

Definition 3.1.2. Let $\sigma_{SAT} := \{\leq, E, F, K, R, I, V\}$, where $\leq$,E, F, K, R, and I are binary and V is ternary. If G is a graph, τ a vocabulary, $M \geq \text{arity}(\tau)$, and $\mathcal{A}$ a τ–structure, then $\mathcal{U} := \mathcal{U}^{\tau,\mathcal{A},G,M}$ is the following σ_{SAT}–structure:
$\mathcal{U}$ consists of disjoint copies of $A, A^2, \ldots, A^M$ (so tuples over A become single elements, now!), and of

- a copy of G (with $E^{\mathcal{U}}$ as edge relation),
- a copy of G_τ (with $F^{\mathcal{U}}$ as edge relation),
- M elements ordered by $\leq$.

K, R, V are defined as follows (here we denote by i a natural number as well as the i-th element of $\leq^{\mathcal{U}}$; A^i is understood as the copy of A^i in $\mathcal{U}$).
$K := \{(t, i) \mid t \in A^i\}$,
$V := \{(t, a, t') \mid a \in A, \exists i < M : \ t \in A^i, t' \in A^{i+1}, t' = (t, a)\}$,

$I := \{(t,t') \mid t = t' \vee \exists l \geq 1, a_1 \ldots, a_l \in A, t_1 \ldots, t_{l-1}$ with
$Vta_1t_1, Vt_1a_2t_2, \ldots, Vt_{l-1}a_lt'\}$,
$R := \{(t,w) \mid w \in G_\tau, w$ represents S_j for some $j, t \in A^{s_j}, \mathcal{A} \models S_jt\}$. □

In addition to what we mentioned before, we equipped $\mathcal{U}$ with relations V and I, which tell how tuples are extended by single elements, and whether a tuple is an initial segment of another tuple, respectively. This is necessary in order to identify the composition of elements $t \in A^i$.

For the following definition, note that in our framework each formula $\varphi \in \mathcal{L}$ is in $\mathcal{L}[\tau_0]$ for a minimal finite τ_0. So we can speak of the maximum of arities of symbols occurring in φ. We denote it arity(φ).

Definition 3.1.3. (i) A *coding* for $\mathcal{L}$ is a pair of mappings (G, M) both defined on $\bigcup_\tau \mathcal{L}[\tau]$ such that $G(\varphi)$ is a finite graph, and $M(\varphi) \geq$arity(φ) is a natural number for each $\varphi \in \mathcal{L}$.
(ii) Given a coding (G, M) we set
$\mathcal{C}^{G,M} := \{\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)} \mid \tau$ a vocabulary, $\mathcal{A} \in \mathrm{Str}[\tau], \varphi \in \mathcal{L}[\tau]\}$. □

Now, we can define the property of $\mathcal{L}$ being capable of describing its own semantics. As we are not primarily interested in defining the class $\mathcal{C}^{G,M}$ itself in $\mathcal{L}$, we take an approach relative to this class. This will enable us to include the case where $\mathcal{L}$ cannot recognize the set of graphs G of the form $G = G(\varphi)$ for some φ, but where $\mathcal{L}$ can recognize the satisfaction relation for graphs G that do have the form $G(\varphi)$. (However, we will later pass to a canonical encoding which makes it possible to define $\mathcal{C}^{G,M}$ in $\mathcal{L}$.)

Definition 3.1.4. We say that $\mathcal{L}$ *can define its own semantics* if there is a coding (G, M) for $\mathcal{L}$ and a formula $\varphi_{SAT} \in \mathcal{L}[\sigma_{SAT}]$ with

$$\mathrm{Mod}(\varphi_{SAT}) \cap \mathcal{C}^{G,M} =$$

$$\{\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)} \mid \tau \text{ a vocabulary}, \mathcal{A} \in \mathrm{Str}[\tau], \varphi \in \mathcal{L}[\tau], \mathcal{A} \models \varphi\}.$$

□

It should be clear how this can be generalized to a definition of a logic $\mathcal{L}_1$ being capable of defining the semantics of a logic $\mathcal{L}_2$.

We will see later that for many important cases it is possible to drop the intersection with $\mathcal{C}^{G,M}$ above. Hence self definability is indeed a formalization of the idea sketched at the introduction to this section.

3.2 Self definability and complete problems

We want to characterize the logics for which a complete problem exists by means of self definability.

To this end, it is useful to have such complete problems Q where the reductions are easy to handle. Assume Q is complete for $\mathcal{L}$ under quantifier free reductions. First, we can assume that the formula $\psi(\overline{x})$ (see Definition 1.3.1) is dropped, that is the domain of $\mathcal{A}^\Phi$ is A^r. For obvious reasons we call such reductions *non relativizing.* To see that such reductions suffice, pass from Q to the class

$$Q' := \{(A; P, R_1', \ldots, R_s') | \emptyset \neq P \subset A, (P; R_1, \ldots, R_s) \in Q, \text{where } R_i := R_i'|_P\}.$$

The main reason why domain relativization had been introduced at all is to have the correspondence between quantifiers and oracles stated in Theorem 1.3.11.

Furthermore, we can restrict to the case where Q is *simple*, i.e. where the vocabulary of Q, σ, consists of a single relation symbol T. Assume, for instance, $\sigma = \{R_1, R_2\}$, and that R_1 and R_2 are of the same arity k. Let T be of arity $k+2$ and set

$$\tilde{Q} := \{(A; T) | \forall x \neq y Q\overline{x}_1, \overline{x}_2; Txx\overline{x}_1, Txy\overline{x}_2\}.$$

On structures with at least two elements, the formula $Q\overline{x}_1, \overline{x}_2; \psi_1(\overline{x}_1), \psi_2(\overline{x}_2)$ is equivalent to
$\tilde{Q}xy\overline{x}_1; (x = y \wedge \psi_1(\overline{x}_1)) \vee (x \neq y \wedge \psi_2(\overline{x}_1))$.
Summing up, we have:

Lemma 3.2.1. Let Q be $\mathcal{L}$–complete under quantifier free reductions. Then there is also a problem that is $\mathcal{L}$–complete under non relativizing reductions and has a simple vocabulary. □

For later use we give a typical example of a quantifier free reduction that relies only on equality types. It is the basis for a reduction that will produce the structures $\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)}$ from Section 3.1, more precisely modified versions of those.

Example 3.2.2. Let $\sigma = \{E, R\}$ with binary E and unary R. Let G be the graph with two nodes u, v and edges Euv, Evv. We want to map any set A via a quantifier free reduction to the σ–structure $\mathcal{U}$ that consists of a copy of G and of the sets A and A^2. All these copies are disjoint, and the union of A and A^2 is marked as the R–part. Note that all pairs over A are now represented as single elements in R.
As a first attempt, we fix two distinct elements $c, d \in A$. We let the reduction be of arity $r = 5$ and let $\mathcal{U}$ look as follows: The set A is represented by the tuples (c, c, c, d, a) where a runs over A. Similarly for A^2 and the tuples (c, c, d, a, b) where a, b run over A. The node u of G is represented by (c, d, d, c, c), and v by (c, d, d, d, d). It is now easy to define the edges of G on these tuples via equality types.
As our selection of c and d does not match the definition of quantifier free reductions, we make the following modification: Set $\sigma^* := \sigma \cup \{P\}$ with binary P. Now, we map A to the following σ^*–structure $\mathcal{U}^*$: The domain of $\mathcal{U}^*$ is A^5. For distinct $c, d \in A$, the tuples specified above form a copy of $\mathcal{U}$. P is an equivalence relation on the set of these tuples, such that each copy of $\mathcal{U}$ forms an equivalence class. Clearly, there are elements in $\mathcal{U}^*$ that do not belong to

any copy of $\mathcal{U}$, for instance those of the form $(c, d, c, \ldots)$. They are not in the field of P.
This modification will be of no harm later, as in a regular logic one can speak about $\mathcal{U}$ in $\mathcal{U}^*$. What we have gained is that there is a quantifier free (non relativizing) tuple $\Phi = (\psi_E(\overset{5}{x}, \overset{5}{y}), \psi_R(\overset{5}{x}), \psi_P(\overset{5}{x}, \overset{5}{y}))$ such that $A^\Phi = \mathcal{U}^*$ for all sets A. We can take
$\psi_E := x_5 = x_4 = x_1 = y_1 \neq x_2 = x_3 = y_2 = y_3 = y_4 = y_5 \lor$
$x_1 = y_1 \neq x_2 = y_2 = x_3 = y_3 = x_4 = y_4 = x_5 = y_5$,
$\psi_R := x_1 = x_2 = x_3 \neq x_4 \lor x_1 = x_2 \neq x_3$,
$\psi_P := x_1 = y_1 \land$
$\big((x_1 = x_2 = x_3 \neq x_4 \land (y_1 = y_2 = y_3 \neq y_4 = x_4 \lor y_1 = y_2 \neq y_3 = x_4 \lor \ldots)) \lor$
$(x_1 = x_2 \neq x_3 \land (y_1 = y_2 = y_3 \neq y_4 = x_3 \lor y_1 = y_2 \neq y_3 = x_3 \lor \ldots)) \lor \ldots\big)$.
The '…' denote all possible combinations of specified equality types. Note that we introduced no symbol to distinguish between the copies of A and A^2. If we want to make identifiable both A and A^2 in this example, we could introduce a new binary relation and connect all of A to the node u (in the same copy of $\mathcal{U}$), and all of A^2 to v. Similarly, when we define $\mathcal{U}^{\tau, \mathcal{A}, G(\varphi), M(\varphi)}$ on a structure $\mathcal{A}$ later, we have no problems to define the arity relation K which distinguishes between the sets A^j for $1 \leq j \leq M$.

Theorem 3.2.3. Theorem on self definability.
(i) If $\mathcal{L}$ can define its own semantics, then there is a class Q that is complete for $\mathcal{L}$ with respect to quantifier free reductions.
(ii) If there is a class Q that is complete for $\mathcal{L}$ with respect to quantifier free reductions, and if $DTC \leq \mathcal{L}$, then $\mathcal{L}$ can define its own semantics.

Note that almost all logics that are of interest in finite model theory are semi–regular and contain DTC, whence the above theorem can be seen as an equivalence statement.
Proof: (i) Assume that $\mathcal{L}$ can define its own semantics. Set $Q_0 := Mod(\varphi_{SAT})$. This is a candidate for a complete problem. Given τ and a $\varphi \in \mathcal{L}[\tau]$, let us try to define $\mathcal{U}^{\tau, \mathcal{A}, G(\varphi), M(\varphi)}$ on each τ–structure $\mathcal{A}$. That is, we are looking for a tuple Φ_0 of quantifier free formulas (over τ) such that $\mathcal{A}^{\Phi_0} \simeq \mathcal{U}^{\tau, \mathcal{A}, G(\varphi), M(\varphi)}$ for each $\mathcal{A}$. Then we are done, since $\mathcal{A}^{\Phi_0} \in Q_0$ implies $\mathcal{A} \models \varphi$ and vice versa. As in the above example this idea will be modified as follows.
As $G(\varphi)$ and $M(\varphi)$ are finite, it is not hard to find – along the lines of the example (or of Section 2) – a tuple Φ_1 such that $\mathcal{A}^{\Phi_1}$ is a collection of isomorphic copies of $\mathcal{U} = \mathcal{U}^{\tau, \mathcal{A}, G(\varphi), M(\varphi)}$, and of isolated points.[14] As in the example, we augment Φ_1 by a quantifier free formula that defines an equivalence relation on $\mathcal{A}^{\Phi_1}$ such that each copy of $\mathcal{U}$ forms an equivalence class. Let Φ be the augmented tuple. Set $\sigma^*_{SAT} := \sigma_{SAT} \cup \{P\}$ for a new binary relation symbol P.

[14] Perhaps, it is not obvious how to define the initial segment relation, I. Note, however, that for this reduction Φ_1 there is a fixed bound $M(\varphi)$ on the exponents i of powers A^i copied into $\mathcal{U}$. Hence we need only a finite disjunction to define I. This disjunction runs over all $i < j \leq M(\varphi)$.

Then Φ is a reduction from τ–structures to σ^*_{SAT}–structures.
Let Q be the class of σ^*_{SAT}–structures for which P forms an equivalence relation on its field such that there is an equivalence class generating a σ_{SAT}–substructure in Q_0. As Q_0 was $\mathcal{L}$–definable, and by semi–regularity of $\mathcal{L}$, Q is also $\mathcal{L}$–definable. Furthermore we have

$$\mathcal{A}^\Phi \in Q \;\text{ iff }\; \mathcal{U} \in Q_0,$$

since all equivalence classes in $\mathcal{A}^\Phi$ generate a copy of $\mathcal{U}$. So,

$$\mathcal{A}^\Phi \in Q \;\text{ iff }\; \mathcal{A} \models \varphi,$$

which completes this part of the proof.
(ii) Assume that Q is complete for $\mathcal{L}$ with respect to quantifier free reductions. By Lemma 3.2.1, we can assume $\sigma_Q = \{T\}$ and that no relativization is needed. Hence, each $\varphi \in \mathcal{L}$ is equivalent to a formula $Q\overline{x}; \psi_\varphi(\overline{x})$, where $\psi_\varphi(\overline{x})$ is quantifier free in disjunctive normal form. We can fix an assignment $\varphi \mapsto \psi_\varphi$. For simplicity, we assume that T is monadic. It will become clear how to modify the proof for a higher (but fixed!) arity of T. Note however that by the definition of completeness, the arity $r (=: r_\varphi)$ of the reduction $\psi_\varphi(\overset{r}{x})$ may vary as well as those of the symbols occurring in φ (and hence in ψ_φ).
The coding for $\mathcal{L}$: Let $\varphi \in \mathcal{L}[\tau]$. We set $M(\varphi) := max\{r_\varphi, \text{arity}(\tau)\}$. The graph $G(\varphi)$ will be a straightforward encoding of ψ_φ, which has the form

$$\psi_\varphi = \bigvee_{\alpha \in D_\varphi} \bigwedge_{\beta \in C_{\varphi,\alpha}} \theta_{\alpha,\beta}.$$

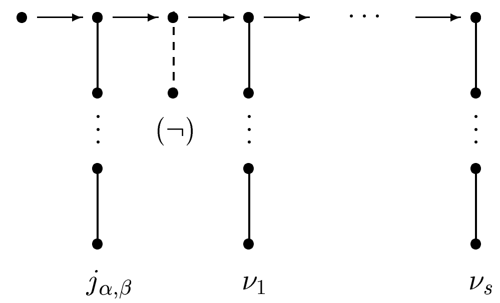
Here $D_\varphi, C_{\varphi,\alpha}$ are finite sets of indices, and $\theta_{\alpha,\beta}$ is an atomic or negated atomic τ–formula. With $\tau = \{S_1, \dots, S_m\}$, each $\theta_{\alpha,\beta}$ is of the form[15]

$$\theta_{\alpha,\beta} = (\neg) S_{j_{\alpha,\beta}} x_{\nu_1} \dots x_{\nu_s},$$

where $s := s_{j_{\alpha,\beta}}$ is the arity of $S_{j_{\alpha,\beta}}$, and the indices ν_k run over $1, \dots, r_\varphi$. We represent $\theta_{\alpha,\beta}$ by the following graph (again writing s for $s_{j_{\alpha,\beta}}$): take a directed path of $s+3$ nodes. We call the first node in this path the *root*. At each node of the path, except for the root, an undirected path of a certain length starts. These lengths are the following numbers: for the successor of the root we take $j_{\alpha,\beta}$ encoding the symbol in question. The next point is connected to a path of length one ore zero depending on whether a negation sign occurs in $\theta_{\alpha,\beta}$ or not. Then follow $\nu_1, \dots, \nu_s$.

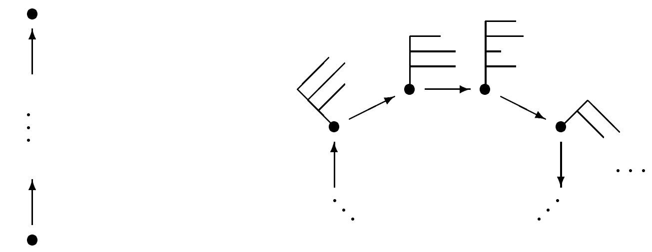
[15]Clearly, $\theta_{\alpha,\beta}$ can also be an (in-)equality statement. We can subsume this case under our graph encoding by treating equality as the binary symbol S_0.

graph encoding of $\theta_{\alpha,\beta}$:



Finally, $G(\varphi)$ looks as follows: for each α, take all the roots of paths representing a $\theta_{\alpha,\beta}$ and connect them to a cycle. The order in which the subgraphs are arranged into a cycle can be taken from the order in which the corresponding formulas are written down in ψ_φ. Disjoint to these points, take a directed path of length r_φ.

$G(\varphi)$:



r_φ (a cycle for a certain α)

Now, we show that $\mathcal{L}$ can define its own semantics via the coding (G, M) defined above. To this end, we show that there exist $DTC[\sigma_{SAT}]$–formulas $\chi_{Rel}(x), \chi_T(x)$ such that

$$\varphi_{SAT} := Qx, x; \chi_{Rel}(x), \chi_T(x)$$

fulfils the requirements of Definition 3.1.4. Note that φ_{SAT} is equivalent to an $\mathcal{L}$–formula, since $\mathcal{L}$ is semi–regular, contains DTC, and Q is $\mathcal{L}$–definable[16].
First, we set $\chi_{Rel}(x) :=$

> $\exists i(Kxi \wedge$ "there is a maximal E–path of lenght i (not connected to any other elements)").

In a structure $\mathcal{U} = \mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)}$, the predicate $\chi_{Rel}(t)$ holds exactly for those $t \in \mathcal{U}$ that represent an r_φ–tuple over A, because the path of length r_φ in $G(\varphi)$ is the only E–path in $\mathcal{U}$ that is not connected to other elements. Comparing i to the length of this path can be done in DTC, because successors are uniquely determined both in $\leq^{\mathcal{U}}$ and in the path.
The intended meaning of $\chi_T(x)$ is the following (in a model $\mathcal{U}$):

[16] It might be astonishing that domain relativization occurs here in contrast to our choice of Q. But it is convenient here to use it, although it is not necessary.

Let $(a_1, \ldots, a_{r_\varphi})$ be the tuple (of elements of A) represented by x (by $\chi_{Rel}(x)$ there is such a tuple). Let ψ_φ be the quantifier free formula in DNF encoded by the E–part of $\mathcal{U}$. Then we have $\mathcal{A} \models \psi_\varphi[\overset{r_\varphi}{a}]$.

Suppose we can find $\chi_T(x) \in DTC$ with this meaning. Then we are done because we find: $\mathcal{U} \models \varphi_{SAT}$ iff $(A^{r_\varphi}; \{\overset{r_\varphi}{a} \mid \mathcal{A} \models \psi_\varphi[\overset{r_\varphi}{a}]\}) \in Q$. By the choice of ψ_φ this is the case if and only if $\mathcal{A} \models \varphi$.
We show that there is a $\chi_T(x) \in DTC$ as claimed above. To simplify the proof, we only take care of a subgraph encoding a subformula $\theta_{\alpha,\beta}$ (it is easy to pass to the disjunction of conjunctions of such formulas).
So, consider a node u which is the root of a path of length $s+3$ encoding an s–ary (perhaps negated) atomic formula via additional paths connected to each node. Let S_j be the relation symbol occuring in that formula. The node $w_j \in G_\tau$ (G_τ is the F–part of the structure) that represents the symbol S_j is DTC–definable from u because a path of length j is attached to the successor of u. Having checked whether the formula is negated or not, we turn to its variables $x_{\nu_1}, \ldots, x_{\nu_s}$ encoded by the last s elements of the path. From x and u we can define an element t that represents the tuple $(a_{\nu_1}, \ldots, a_{\nu_s})$. This can be done as follows (remember we assumed $(a_1, \ldots, a_{r_\varphi})$ to be the tuple represented by x).
Making use of the relation V we build up t successively. V tells us how tuples are extended with single elements. At each step k (for $1 \leq k \leq s$), we have to define the element a_{ν_k} coming into t next. a_{ν_k} is the ν_k–th component of x. It can be characterized by deterministic paths as follows (note in the following that undirected paths cause no difficulties; if one wants to go through their elements, one can use two variables, one for the current position and one for the direction one is heading for):
Go simultaneously through the undirected path of length ν_k and through the sequence $a_1, a_2, \ldots$ until the end in the undirected path is reached. The second path $(a_1, a_2, \ldots)$ becomes deterministic if one considers a third path, the path of initial segments $a_1, (a_1, a_2), \ldots$, and imposes the following conditions:
$Va_1a_2(a_1, a_2),\ V(a_1, a_2)a_3(a_1, a_2, a_3), \ldots$
$Ia_1x,\ I(a_1, a_2)x, \ldots$
Now, in all three paths the succesors are uniquely determined and a_{ν_k}, the last point of the second path, is indeed DTC–definable from x, u, and the actual position k in the directed E–path. Therefore, t is DTC–definable from x and u.
Having t available, the truth of $\theta_{\alpha,\beta}$ in $\mathcal{A}$ is equivalent to $R^{\mathcal{U}}tw$ (in the non negated case; it is clear how to deal with the negated case and the (in-)equality case). This concludes the proof. $\square$

The notion of self definability becomes more natural if one can convince oneself of the fact that any reasonable encoding of formulas (other than the graph encoding) yields the same notion. But this is true (up to the vagueness in "reasonable encoding"): Take, for instance, strings (i.e. word structures). Clearly, self definability by strings implies self definability by graphs, because word structures can easily be translated into graphs. On the other hand, if self definability by graphs holds, then there is a complete problem (by part (i) of the Theorem on self definability). From a complete problem, in turn, one can conclude to self definability by strings. To see this, imitate part (ii) of the last proof for an appropriate encoding by strings.

We promised to improve Definition 3.1.4 by eliminating the intersection with $\mathcal{C}^{G,M}$. We do this by means of the *canonical* encoding (G, M) obtained in part (ii) of the last proof. We make an additional assumption on $\mathcal{L}$, which holds for most of the logics studied in finite model theory.

Theorem 3.2.4. (Self definability revisited)
Assume $DTC \leq \mathcal{L}$ and that for each τ and each $\varphi \in \mathcal{L}[\tau]$ there are infinitely many sentences $\varphi' \in \mathcal{L}[\tau]$ that are equivalent to φ. Then, $\mathcal{L}$ can define its own semantics if and only if the following holds:
There is a coding (G, M) for $\mathcal{L}$, and a formula $\varphi_{SAT} \in \mathcal{L}[\sigma_{SAT}]$ with

$$\mathrm{Mod}(\varphi_{SAT}) = \{\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)} \mid \tau \text{ a vocabulary}, \mathcal{A} \in \mathrm{Str}[\tau], \varphi \in \mathcal{L}[\tau], \mathcal{A} \models \varphi\}.$$

Proof: We write $\mathcal{C}_{SAT}$ for the right side of the equation in the theorem. (This class also occurs on the right side in Definition 3.1.4.)
Clearly, if $\mathrm{Mod}(\varphi_{SAT}) = \mathcal{C}_{SAT}$ for some encoding, then also $\mathrm{Mod}(\varphi_{SAT}) \cap \mathcal{C}^{G,M} = \mathcal{C}_{SAT}$, because $\mathcal{C}_{SAT} \subset \mathcal{C}^{G,M}$. So in this case we have self definability.
For the converse, assume $\mathcal{L}$ is self definable via a coding (G, M). By the last theorem, we can assume (G, M) to be the canonical encoding constructed for part (ii) of the theorem. There was a free choice of an assignment $\varphi \mapsto \psi_\varphi$ ($\varphi \in \mathcal{L}$, ψ_φ in DNF such that $Q\bar{x}; \psi_\varphi$ is equivalent to φ). We arrange this mapping so as to make sure that *all quantifier free formulas in DNF, ψ, occur.* In fact, by semi–regularity and by our assumption on the syntax of $\mathcal{L}$, there are infinitely many $\mathcal{L}$–formulas equivalent to $Q\bar{x}; \psi$ (for ψ as above and all variables bound). Hence we can map one of those formulas to ψ, and we can do this for all ψ.
We show $\mathcal{C}^{G,M}$ =Mod(φ_0) for a $\varphi_0 \in DTC$. Obviously, this is sufficient.
For a structure $\mathcal{U}$ ∈Str$[\sigma_{SAT}]$ we have $\mathcal{U} \in \mathcal{C}^{G,M}$ just in case $\mathcal{U}$ is of the form $\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)}$ for some τ, $\mathcal{A}$, and φ. This amounts to the following list of conditions (it is easily checked that they all hold in case $\mathcal{U}= \mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)}$; but also the converse holds, as we will see below; the conditions will obviously be DTC–expressible):

- (field$(\leq), \leq$) is an ordering of some length $M \geq 1$.
  We write 1, and M, for the first, and last element respectively.
  Level(v) denotes $\{x \mid Kxv\}$ and its usage indicates v ∈field$(\leq)$.
- The domain, U, is the disjoint union of $H := \{x \mid \exists v Kxv\}$, field$(E)$, field$(F)$, and field$(\leq)$.
  H is the disjoint union of Level(1),. . . ,Level(M).
- D :=Level(1)$\neq \emptyset$.

- $$\begin{aligned}\forall xuy(Vxuy \rightarrow \quad & (u \in D \wedge x, y \in H \wedge \\ & \exists v < M(x \in \text{Level}(v) \wedge \\ & y \in \text{Level}(v+1)) \wedge \\ & \forall u'(Vxu'y \rightarrow u' = u) \wedge \\ & \forall y'(Vxuy' \rightarrow y' = y) \wedge \\ & \forall x'(Vx'uy \rightarrow x' = x)))\end{aligned}$$

- $\forall xuv(v < M \wedge u \in D \wedge x \in \text{Level}(v) \rightarrow \exists y Vxuy)$
- $\forall tt'(Itt' \leftrightarrow (\exists uVtut' \vee [DTC_{x,y}(Iyt' \wedge \exists uVxuy)]tt'))$.
- The F–part is a graph of type G_τ for some τ.
- The E–part is a graph consisting of a directed path of some length $r \geq 1$, and of a cycle in which to each point there is a subgraph attached. These subgraphs are of the earlier described form (i.e. encodings of atomic or negated atomic formulas). The occuring arities (i.e. the undirected paths) are all present as artities of symbols in τ. Similarly for the indices of relation symbols.
- $\forall tw(Rtw \rightarrow t \in H \wedge w$ lies on the directed path of $G_\tau)$.
- R and K are consistent, that is to say, $Rtw \wedge Ktv$ implies that the position of v (w.r.t. $\leq$) equals the arity of the symbol encoded by w (present as an undirected path).
- All numbers listed so far are consistent, i.e. satisfy the (in-)equalities as in a structure $\mathcal{U}$ of type $\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)}$. For instance, $M = \max\{r, \text{arity}(\tau)\}$.

If these conditions hold for a structure $\mathcal{U}$, one can derive a structure $\mathcal{A} \in \text{Str}[\tau]$ for the encoded τ. It has domain D, and the relations are given by R. To show that $I^U tt'$ describes the initial segment relation, one can proceed by induction on the difference in length between t and t'.
There must also be a $\varphi \in \mathcal{L}[\tau]$ such that $\mathcal{U} = \mathcal{U}^{\tau,\mathcal{A},G(\varphi),M(\varphi)}$. To see this, let $\psi(\overset{r}{x})$ be the formula in DNF encoded by $E^{\mathcal{U}}$. By assumption on (G, M), there is a $\varphi \in \mathcal{L}[\tau]$ with $\psi = \psi_\varphi$. So $M(\varphi) = \max\{r, \text{arity}(\tau)\}$, and, as stated in the above list, this number is equal to the number M encoded by $\leq^{\mathcal{U}}$. Hence $\mathcal{U} \in \mathcal{C}^{G,M}$. $\square$

3.3 Examples

The fixed point logics *IFP* and *PFP* can define their semantics as there are complete problems for these logics (see the previous section). One can also check directly that there is a description of the semantics via the encoding of formulas in normal form in a straightforward way (this yields complete problems, in turn!). We sketch the idea:

Assume, the first order part ψ of a fixed point predicate $[\mathit{IFP}_{\overline{x},X}\psi]$ is represented by a syntax tree reflecting its inductive composition (a disjunction has two succesors, a node $\exists x$ has one succesor and so on; each node is marked with a subgraph encoding the sort of the node and the involved variables). Given a structure $\mathcal{A}$ and a set X of tuples over A, one can inductively define the set S consisting of the pairs $(u;\overline{a})$, where u is a node in the syntax tree, and $\overline{a}$ a tuple that makes the subformula represented by u true in $(\mathcal{A}, X)$. Provided that $M \geq$ arity$(\overline{a})$, this induction can be described in $\mathcal{U}^{\tau,\mathcal{A},G(\varphi),M}$ by means of an *IFP*–formula, since in this case, we can regard S as a subset of U^2.

How about the less expressive stratified[17] fixed point logic, *SFP*, or transitive closure logic, *TC*? Here, we have a strict hierarchy for the nesting of *SFP*– (resp. *TC*–) operators [16]. The attempt of defining the semantics within these logics in a straightforward way fails now, as it is not clear how to deal with a varying nesting of these operators in the encoded formulas. But there might still be a more elegant encoding of the class of *SFP*– (or *TC*–) definable queries. However, the Theorem on self definability 3.2.3 rules out this possibility and shows that the nesting hierarchy is a principal obstacle to defining the semantics.

Theorem 3.3.1. The logics *TC* and *SFP* cannot define their own semantics.

Proof: We argue for *TC*. Remember that $FO(Q_{TC}) \equiv TC$ for the class Q_{TC} given in 1.3.3. Moreover, the nesting of Q_{TC}–quantification equals that of *TC*–operators in the given translations and vice versa. So we can apply Remark 2.2.9 and the nesting hierarchy for the *TC*–operator and get: For each quantifier Q with $FO(Q) \equiv TC$ there must be a strict nesting hierarchy for Q. Hence there cannot be a complete problem for *TC*. □

The well known strict inclusion *SFP* < *IFP* can be interpreted in the sense that *IFP* can define the semantics of *SFP* (because the latter is a fragment of *IFP* and *IFP* is self definable), whereas *SFP* cannot.

Let us turn to logics where it is unknown whether they are equivalent to a logic of the form $FO(Q)$. For second order logic, *SO*, this is the case *iff* there is a complete problem[18] for it *iff* the polynomial hierarchy collapses (Theorem 2.4.2). By Theorem 3.2.3, we get the further equivalent condition that *SO* can define its own semantics.

Another example of this kind is *PTIME*, which can be described either by a logic $FO(Q)$ or by no recursive logic at all, as was shown by Dawar [10]. Using Theorem 3.2.3 we can restate Dawar's result.

Definition 3.3.2. We call a set $\mathcal{L}$ of words over the alphabet $\{0, 1\}$ a *recursive*

[17]See [12] for a definition.

[18]under either quantifier free or *LOGSPACE* or*PTIME* reductions

logic for $PTIME$ if the following conditions hold.

(i) $\mathcal{L}$ is recursive.

(ii) there is a machine T with the following properties:

   (a) Whenever $\varphi \in \{0,1\}^*$ is an element of $\mathcal{L}$, then T, started with input φ, yields an output (T_φ, r_φ), where T_φ is a (canonical encoding of a) deterministic machine accepting a query[19] in time n^{r_φ}.

   (b) For $\varphi \notin \mathcal{L}$, T halts with output $T_\varphi = \emptyset$, $r_\varphi = 0$.

   (c) All queries in $PTIME$ are accepted by a machine T_φ for some $\varphi \in \mathcal{L}$.

$\square$

It is not known whether such a logic exists. Assuming $PTIME \equiv NP$, one can use logical characterizations of NP to establish a recursive logic in the above sense. There is however a critical point in this argument: Whereas Σ_1^1 can clearly be regarded as a logical characterization of NP, and *nondeterministic* machines together with time bounds can effectively be derived from Σ_1^1–formulas, it is not clear, how the bare knowledge of $PTIME \equiv \Sigma_1^1$ could be used to effectively derive *deterministic* machines and time bounds from the formulas.

The class Q_{SAT}, shown to be NP–complete under quantifier free reductions in 2.4, helps out of this dilemma: Assuming $PTIME \equiv NP$, we can conclude $PTIME \equiv NP \equiv FO(Q_{SAT})$. So Q_{SAT} is computable in time n^r for a fixed exponent r, and one can effectively assign an equivalent machine and a time bound to each $FO(Q_{SAT})$–formula. Thus, after passing to an encoding over $\{0,1\}$, one obtains a recursive logic in the strict sense of Definition 3.3.2.

Theorem 3.3.3. If there is a recursive logic for $PTIME$, then $PTIME$ can define its own semantics.
Hence, under this assumption, there is a problem Q that is $PTIME$–complete under quantifier free reductions (the last conclusion restates a result of Dawar [10]).

Proof: Let $\mathcal{L}$ be a recursive logic for $PTIME$, and let T be a machine as indicated in Definition 3.3.2. Let $t(\varphi)$ be the number of steps required by T on input φ.
We define a coding (G, M) for $\mathcal{L}$: Let $G(\varphi)$ be a canonical graph encoding for the word φ (canonical here means that φ can be retrieved from $G(\varphi)$ in polynomial time). Set $M(\varphi) := t(\varphi) + r_\varphi$.
Let φ_{SAT} be an $\mathcal{L}$–representation of the following σ_{SAT}–query (which will turn out to be computable in polynomial time):

> The input is a σ_{SAT}–structure $\mathcal{U}$. First, retrieve a string $\varphi \in \{0,1\}^*$ from the E–part and the length M of the $\leq$–part, whenever possible.

[19] in the sense of Definition 1.1.1. Remember however that in this section a query is a class of structures. As usual, this restriction could be removed.

From this, simulating T, compute (T_φ, r_φ).
If $T_\varphi \neq \emptyset$ and $r_\varphi \leq M$, then simulate T_φ on the structure $\mathcal{A}$ encoded in $\mathcal{U}$ (if there is no structure encoded, stop). For this purpose, use additional tapes for the actual state of T_φ and for the presentation of $\mathcal{A}$ (the input ordering for $\mathcal{A}$ can be derived from the ordering in which $\mathcal{U}$ had been presentated).
Accept if and only if T_φ accepts $\mathcal{A}$ in up to $|A|^M$ steps.

Computing the encoding for $\mathcal{A}$ can be done in polynomial time. Simulating T_φ takes $O(t(\varphi)|A|^M) = O(|U|)$ steps, where the first factor is an upper bound for the number of states in T_φ and hence for the time for reading and writing the actual state in the simulation.
In fact, this program defines a query, i.e. its output does not depend on the order in which the input $\mathcal{U}$ is presented: If $M \geq r_\varphi$, this follows from the order invariance of the machine T_φ. The only critical case is that $\mathcal{U}$ encodes an $\mathcal{L}$–formula φ and a number $M < r_\varphi$. But in this case the program rejects $\mathcal{U}$, hence maintaining order independence. □

Note that this is by no means a new proof for Dawar's result. Rather the original proof has been split up in two parts: a purely computational part (the last proof) followed by an argument which holds in a more general context, part (i) of Theorem 3.2.3. That direction of Theorem 3.2.3 can be regarded as a logical pendant to Dawar's general theorem in [10] which concludes from certain computational assumptions to the existence of complete problems.

One could weaken Definition 3.3.2 by abandoning the condition of having explicit time bounds (the numbers r_φ in (ii)(a)). But there is no analogue of Theorem 3.3.3 known under this weaker assumption.

Another way of weakening Definition 3.3.2 is to replace the time bounds by a strong condition of order invariance. Knowing the time bounds allows one to build in clocks into the machines T_φ. Hence, the existence of a recursive logic for *PTIME* implies the existence of a *weak* recursive logic for *PTIME*:

Definition 3.3.4. A set $\mathcal{L}$ of words over the alphabet $\{0,1\}$ is called a *weak recursive logic* for *PTIME*, if the following conditions hold.

(i) $\mathcal{L}$ is recursive.

(ii) there is a machine T with the following properties:

  (a) Whenever $\varphi \in \{0,1\}^*$ is an element of $\mathcal{L}$, then T, started with input φ, yields an output T_φ, where T_φ is a (canonical encoding of a) deterministic machine T_φ accepting a query in polynomial time. For each input structure, $\mathcal{A}$, the *number of steps* in the computation of T_φ *depends only on* $\mathcal{A}$, not on the order $\prec$ by which $\mathcal{A}$ is encoded as an input string (see Def. 1.1.1).

  (b) For $\varphi \notin \mathcal{L}$, T halts with output $T_\varphi = \emptyset$.

  (c) as in 3.3.2.

□

Theorem 3.3.5. There is a recursive logic for $PTIME$ if and only if there is a weak recursive logic for $PTIME$.

Proof: In Theorem 3.3.3, one can replace a recursive logic by a weak recursive logic. To see this, note that the invariance condition also leads to a program the output of which does not depend on the input order and hence defines a σ_{SAT}–query. One can use the same program as in the last proof, cancelling only the test of $r_\varphi \leq M$. So we can conclude from a weak recursive logic to a complete problem.
The complete problem, in turn, yields a logic $FO(Q)$ for $PTIME$. Such a logic yields a recursive logic (as indicated above where we studied the case $PTIME = NP$). □

4 Quantifiers versus fixed point operators

Throughout this section, let Q_{IFP} denote a class of structures for which $FO(Q_{IFP}) \equiv IFP$ holds. We can take the monotone circuit value problem Q_{MC} for Q_{IFP}, or the class derived from the self definability of IFP in the previous section, or any of the examples in the literature (see 2.2). It will always become clear that our particular choice has no influence on the results. Similarly, Q_{PFP} will denote a class with $FO(Q_{PFP}) \equiv PFP$.

4.1 Turing- and many-one-reducibility

Definition 4.1.1.
(i) Given a quantifier Q and another class of structures $\tilde{Q}$ such that $\tilde{Q} \in PTIME^Q$, we call $\tilde{Q}$ *PTIME-Turing-reducible to* Q.
(ii) We call $\tilde{Q}$ *IFP-Turing-reducible to* Q, if $\tilde{Q} \in IFP(Q)$. □

As a consequence of Theorem 1.3.11, $PTIME$-Turing-reducibility and IFP-Turing-reducibility coincide for the case of $\tilde{Q}$ consisting only of ordered structures. If Q, too, is ordered, this amounts to the notion of $PTIME$-Turing-reducibility between the sets of strings encoding the structures.

We are going to determine the expressive power of $IFP(Q)$ for both the ordered and the general case. Looking at the class of queries definable in this logic, the question arises, whether this class is minimal under the requirements to contain all IFP–definable queries as well as Q itself, and to be closed under some natural operations such as negation, conjunction etc. In Definition 1.3.5 we have subsumed these closure properties under the notion of regularity.

By the Minimality Lemma 1.3.6 the minimal regular logic containing IFP and (the vectorizations of) Q is $FO(Q_{IFP}, Q)$. In particular, its expressive power does not depend on the choice of Q_{IFP}.

We will now establish some relations between Lindström logics[20] such as

[20] So far we used this expression only for extensions of FO with a single quantifier and its

$FO(Q_{IFP}, Q)$ and another type of reduction, the so called *many–one–reductions*.

In recursion theory, a many–one–reduction to a problem Q is understood as a mapping π subject to certain resource bounds. π is said to reduce a problem $\tilde{Q}$ to Q, if, for every instance A of $\tilde{Q}$ (here, A is a string, and $Q, \tilde{Q}$ are sets of strings), we have

$$A \in \tilde{Q} \quad \text{iff} \quad \pi(A) \in Q.$$

Demanding π to be computable in a number of steps that is polynomial in the size of A leads to $PTIME$–many–one–reducibility. It is clear that $PTIME$–many–one–reducibility implies Turing–many–one–reducibility, and it is well known that the converse does not always hold [28].

The quantifier free reductions introduced in 2.1 can be seen as a logical version of many–one–reductions, taking for Q and $\tilde{Q}$ classes of structures, and setting $\pi(\mathcal{A}) := \mathcal{A}^{\Phi}$, where $\mathcal{A}$ is a structure and Φ a tuple of quantifier free formulas of the signature of $\mathcal{A}$ and of appropriate arities (as exhibited in 2.1). We can generalize this to any given logic $\mathcal{L}$ involving free variables.

Definition 4.1.2. A mapping π of the form $\pi(\mathcal{A}) = \mathcal{A}^{\Phi}$, where $\mathcal{A}$ is a structure and Φ a tuple of $\mathcal{L}$–formulas (of the signature of $\mathcal{A}$ and of appropriate arities, see 2.1), is called an $\mathcal{L}$*–many-one-reduction.* □

Remark 4.1.3. If a class of structures, $\tilde{Q}$ is *IFP*–many–one–reducible to another class Q, then $\tilde{Q}$ is definable in $FO(Q_{IFP}, Q)$. □

Proof: To say that a tuple Φ of *IFP*–formulas reduces $\tilde{Q}$ to Q amounts to saying that $\tilde{Q}$ is defined by the $IFP(Q)$–formula $Q\overline{x}; \Phi$ (here, $\overline{x}$ is a tuple of tuples of variables, the arities being chosen according to Φ). Replacing the formulas in Φ by equivalent $FO(Q_{IFP})$–formulas yields the desired characterization of $\tilde{Q}$ in $FO(Q_{IFP}, Q)$. □

Now, assume Q and $\tilde{Q}$ to be ordered. It is known that Immerman's and Vardi's results that relate deterministic complexity classes to logics on ordered structures (see 1.1) carry over to the case of reductions.[21] Hence, we get:

Remark 4.1.4. Let Q and $\tilde{Q}$ be classes of ordered structures. Then $\tilde{Q}$ is *IFP*–many–one–reducible to Q if and only if it is $PTIME$–many–one–reducibel to Q. □

We turn to arbitrary structures again, and summarize our observations.

Remark 4.1.5. For each fixed Q, we have the inclusions

$$\{\tilde{Q} | \tilde{Q} \text{ is } IFP\text{–many–one–reducible to } Q\} \leq FO(Q_{IFP}, Q) \leq IFP(Q) \equiv$$

vectorizations. However, finitely many quantifiers can be represented by a single one, cf. the proof of Theorem 4.5.2 below.

[21] For the $LOGSPACE \equiv DTC$–case an explicit proof can be found in [12]. It can directly be adapted to the equivalence of $PTIME$ with IFP.

$\{\tilde{Q} | \tilde{Q}$ is *IFP*–Turing–reducible to $Q\}$.

Moreover, we can replace *IFP* in the first and last class with *PTIME* if we pass to ordered[22] structures Q and $\tilde{Q}$. □

Hence when we show in 4.3 that there is a class Q with $FO(Q_{IFP}, Q) < IFP(Q)$, then this also separates the two notions of *IFP*–reducibility, Turing– and many–one–reducibility.

For the corresponding pair of reductions that are linked to *PTIME*, the difference is well known [28], and we will restate this fact in 4.4 by constructing a class Q with $FO(Q_{IFP}, Q) <_{\mathcal{O}} IFP(Q)$. The proof in [28] that separates the two notions of *PTIME* reducibility is based on a class Q that is not many–one–reducible to its complement. This argument cannot be used for separating $FO(Q_{IFP}, Q)$ from $IFP(Q)$, since both logics are closed under complementation.

It is noteworthy that the proof of $FO(Q_{IFP}, Q) < IFP(Q)$ for not necessarily ordered structures will work with a class Q of very low complexity, namely the even cardinality quantifier, whereas that for ordered structures clearly requires a Q which is not *PTIME*–computable. However, both of our proofs will have one thing in common. They make essentially use of the fixed finite nesting depth of the quantifier Q in any $FO(Q_{IFP}, Q)$–formula. Of course, such a depth can also be defined for $IFP(Q)$–formulas, but there it does not reflect the maximum number of Q–evaluations that depend on each other. Intuitively, this is because the second order variables can be used to store information that is repeatedly made available to Q. Hence a polynomial nesting depth can be simulated (it will not be necessary to formalize this idea, but examples can be found in 4.3 and 4.4). Depriving $IFP(Q)$ from exactly this feature yields another characterization of $FO(Q_{IFP}, Q)$:

Theorem 4.1.6. For any class of finite structures, 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 1.3.6. 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 $IFP(Q)^*$ consist 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 free 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 Q.

[22]Now, also Q has to be a class of ordered structures. Then, Remark 4.1.4 applies. In principle, it is possible to consider also *PTIME* many–one–reductions to an unordered class Q. Remark 4.1.4 remains true but one has to modify the cited proof slightly. We will not need this general type of reduction.

For each subformula $\theta(\overline{z}) = Q\overline{x}; \psi_P(\overline{x}, \overline{z})$ of χ that is not in the scope of a further 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 free in the formulas ψ_P, we can make the following substitutions: Replace each of the above formulas $\theta(\overline{z})$ 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 a formula in $FO(Q_{IFP}, Q)$ which is equivalent to φ. □

4.2 Infinitary languages and games

Given a quantifier Q, we are going to embed the logic $FO(Q_{IFP}, Q)$ into an infinitary language and define an Ehrenfeucht-Fraïssé game which respects the nesting of Q. This will enable us to establish queries that are not definable in this logic, but can be expressed in $IFP(Q)$, where bounded nesting of Q can be overcome by storing information in the second order variable.

Let $\mathcal{L}_{\infty\omega}$ denote the extension of first order logic that allows for conjunctions and disjunctions over arbitrary sets of formulas. As each finite structure can be characterized up to isomorphism by a single first order formula, $\mathcal{L}_{\infty\omega}$ can express all queries on finite structures, and, hence, is of no interest. However, the restriction $\mathcal{L}^\omega_{\infty\omega}$, consisting of all formulas that involve only a finite number of distinct variables, has interesting model theoretic properties, and is still strong enough to simulate fixed point operations [26], i.e. we have $PFP \leq \mathcal{L}^\omega_{\infty\omega}$. This inclusion has widely been used to prove non–expressibility results for fixed point logic by means of Ehrenfeucht-Fraïssé games for $\mathcal{L}^\omega_{\infty\omega}$. We modify this technique by combining it with games for generalized quantifiers (as studied in [40]), and by imposing a boundary $l \geq 0$ to the number of moves linked to those quantifiers. The resulting game will be a means to prove the equivalence of two structures with respect to a fragment of $\mathcal{L}^\omega_{\infty\omega}(Q)$, in which the nesting depth of Q is bounded by the number l. This fragment will be strong enough to contain all $FO(Q_{IFP}, Q)$–formulas of Q–rank up to l.

Definition 4.2.1. Let $k, r, l \geq 0$. A formula $\varphi \in \mathcal{L}^\omega_{\infty\omega}(Q)$ belongs to $\mathcal{L}^k_{\infty\omega}(Q^r_l)$, if at most k variables occur in φ, the number of nested Q–quantifiers in φ is bounded by l, and their arity by r. We set

$$\mathcal{L}^\omega_{\infty\omega}(Q^\omega_\omega) := \bigcup_{k,r,l\geq 0} \mathcal{L}^k_{\infty\omega}(Q^r_l)$$ □

Note that $\mathcal{L}^\omega_{\infty\omega}(Q^\omega_\omega)$ is a regular logic, as a substitution of a relation symbol with a predicate defined by a subformula from $\mathcal{L}^k_{\infty\omega}(Q^r_l)$ does not increase the nesting of Q by more than l.

Since for all $r \geq 1$ we have $Q^{r,P}_{IFP} \in IFP \leq \mathcal{L}^\omega_{\infty\omega}$, $Q^{r,P} \in \mathcal{L}^\omega_{\infty\omega}(Q^r_1)$, and by regularity of $\mathcal{L}^\omega_{\infty\omega}(Q^\omega_\omega)$, the Minimality Lemma 1.3.6 yields

Lemma 4.2.2. $FO(Q_{IFP}, Q) \leq \mathcal{L}^\omega_{\infty\omega}(Q^\omega_\omega)$. □

This inclusion can also be obtained by extending the proof of $IFP \leq \mathcal{L}^\omega_{\infty\omega}$

to the logic $IFP(Q)^*$ from Theorem 4.1.6, thereby showing that the nesting l is being preserved.

For the rest of this section let $\sigma_Q = \{R\}$, where R is unary. Moreover, we will restrict ourselves to formulas without relativization in the Q-clause. This will not affect the expressive power for the even cardinality quantifier to be studied later on.

Definition 4.2.3. Let $l \geq 0, k \geq r \geq 0$. The *$k, r, l$–game* on two structures $\mathcal{A}$ and $\mathcal{B}$ is the following extension of the infinite k–pebble game:
There are two players I and II, and k pairs $(u_1, v_1), \ldots, (u_k, v_k)$ of pebbles being off the board before the game starts. After each move some of the u_ν are on elements of $\mathcal{A}$, and the corresponding v_ν are on elements of $\mathcal{B}$. The game is being played indefinitely, and if the mapping defined by the pebbled elements is a partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$ after each move, player II wins. Otherwise the game ends after a finite number of rounds and player I wins.
A move is either a *point move*, a *Q–move* or a *$\neg Q$–move.* Player I is free to choose the type of move, but he must not make more than l $(\neg)Q$–moves in the whole game.
In a point move he picks up a pebble u_ν (or v_ν) and puts it on an element of $\mathcal{A}$ (resp. $\mathcal{B}$). Player II puts v_ν (resp. u_ν) on an element of $\mathcal{B}$ (resp. $\mathcal{A}$).
In a Q–move player I first chooses a number $q \in \{1, \ldots, r\}$ (if $r = 0$, no Q–move is allowed). Then he chooses a set $S \subset A^q$ such that $(A^q, S) \in Q$. Player II answers with a set $T \subset B^q$ such that $(B^q, T) \in Q$. We will later refer to S and T as sets of *red coloured* tuples, and to their complements as sets of *green coloured* tuples.
Then, player I picks up q pebbles v_ν and places them on a tuple $\overline{b} \in B^q$ (he may place several pebbles on single elements of B). Player II places the corresponding pebbles u_ν on a tuple $\overline{a} \in A^q$. He has to respect the colouring, that is he must make sure that $\overline{a} \in S$ iff $\overline{b} \in T$. This ends the move.
Similarly, a $\neg Q$–move is carried out with the roles of $\mathcal{A}$ and $\mathcal{B}$ exchanged.
The sets S and T are of no matter for the rest of the game. □

There are several games for infinitary logics and Lindström quantifiers in the literature [5, 25, 40]. The variant we defined above suits to the analysis of nesting hierarchies for these quantifiers. The game can easily be adapted to arbitrary signatures σ_Q, and to domain relativization, by colouring several sets independently.

Definition 4.2.4. We write $\mathcal{A} \equiv^*_{k,r,l} \mathcal{B}$ to express that $\mathcal{A}$ and $\mathcal{B}$ satisfy the same $\mathcal{L}^k_{\infty\omega}(Q^r_l)$–formulas without domain relativization in the Q–clause.
$\mathcal{A} \equiv_{k,r,l} \mathcal{B}$ denotes full $\mathcal{L}^k_{\infty\omega}(Q^r_l)$–equivalence of the structures. □

Theorem 4.2.5. If player II has a winning strategy in the k, r, l–game, then $\mathcal{A} \equiv^*_{k,r,l} \mathcal{B}$.

Proof: By induction on $\varphi \in \mathcal{L}^{\omega}_{\infty\omega}(Q^{\omega}_{\omega})$ we show the following implication:

> Assume $\varphi \in \mathcal{L}^{k}_{\infty\omega}(Q^{r}_{l})$ (without relativization) and that l is minimal with this property. Let $\overset{k}{a} \in A \cup \{*\}$, $\overset{k}{b} \in B \cup \{*\}$ ($*$ stands for the position off the board). Assume $\mathcal{A} \models \varphi[\overline{a}]$ and $\mathcal{B} \models \neg\varphi[\overline{b}]$, or $\mathcal{A} \models \neg\varphi[\overline{a}]$ and $\mathcal{B} \models \varphi[\overline{b}]$ (here, the variables that are assigned the value $*$ must not occur free in φ, and it is assumed that $*$ occurs at the same positions in $\overline{a}$ and $\overline{b}$).
>
> Then player I has a winning strategy in the k,r,l–game started with the pebbles on $\overline{a}$ and $\overline{b}$.

We consider only the Q–step. So let $\varphi = \varphi(\overline{y}) = Q \overset{q}{x}\ \psi_1(\overset{q}{x}, \overline{y})$ and assume, for instance, $\mathcal{A} \models \varphi[\overline{a}]$ and $\mathcal{B} \models \neg\varphi[\overline{b}]$.
Consider the game with k pairs of pebbles placed on $\overline{a}$ and $\overline{b}$. As φ has at most $k-q$ free variables, we can replace some elements in $\overline{a}$ and $\overline{b}$ by the symbol $*$ so that there are at least q pairs of that symbol, thereby preserving $\mathcal{A} \models \varphi[\overline{a}]$, $\mathcal{B} \models \neg\varphi[\overline{b}]$. We take the corresponding pebbles off the board. It is clear that player I wins the original game, if he wins this modified game with possibly more pebbles off the board (because, for the original game, he can reuse pebbles that are on the board as though they were off the board).
Now, the strategy for player I is to start a Q–move with $S := \{\overline{c} \in A^q \mid \psi_1[\overline{c}, \overline{a}]\}$. Note that $(A^q, S) \in Q$. When player II answers with a set $T \subset B^q$, then T must be different from $T_1 := \{\overline{d} \mid \psi_1[\overline{d}, \overline{b}]\}$, because $(B^q, T_1) \notin Q$. If, for instance, there is a $\overline{d} \in T \setminus T_1$, then player I will pebble $\overline{d}$ (note that he can use q pebbles which are off the board). Player II answers with some $\overline{c} \in S$, and therefore $\mathcal{A} \models \psi_1[\overline{c}, \overline{a}]$, but $\mathcal{B} \models \neg\psi_1[\overline{d}, \overline{b}]$. By induction, player I wins the remaining $k, r, (l-1)$–game with the pebbles placed on $(\overline{c}, \overline{a})$ and $(\overline{d}, \overline{b})$. □

Under certain assumptions, the converse of Theorem 4.2.5 and of its generalization to arbitrary signatures is also true. One such assumption is that Q be monotone (see [40] for how to prove the existence of winning strategies from the equivalence of structures). In Section 2.5 we saw that, in principle, one could restrict to this case for the price of a more complex signature.

4.3 A hierarchy theorem for the nesting of quantifiers

We prove a nesting hierarchy to be strict for the *even cardinality quantifier* $Q_{EVEN} := \{(A, R) \mid |R| \equiv 0\ (2)\}$. From this hierarchy we will derive a separation of the Q_{EVEN}–extensions of $FO(Q_{IFP})$ and IFP. In the following, we write Q for Q_{EVEN}. For fixed $k \geq 0$ we construct sequences of trees $(\mathcal{A}_l)_{l\geq 0}, (\mathcal{B}_l)_{l\geq 0}$ such that the following holds:

(i) $\mathcal{A}_l \equiv_{k,1,l} \mathcal{B}_l$ for all l.

(ii) For all l there is a $\varphi_l \in FO(Q^1_{l+1})$ such that $\mathcal{A}_l \models \varphi_l, \mathcal{B}_l \models \neg\varphi_l$.

(iii) There is a $\varphi \in IFP(Q)$ with $\mathcal{A}_l \models \varphi, \mathcal{B}_l \models \neg\varphi$ for all l.

The formulas φ and φ_l will not depend on k. So from (i) and (ii) one derives the following nesting hierarchy (for the meaning of upper and lower indices see 4.2.1):

Theorem 4.3.1. $\mathcal{L}^{\omega}_{\infty\omega}(Q^1_1) < \mathcal{L}^{\omega}_{\infty\omega}(Q^1_2) < \dots$ $\square$

Similarly, if we fix $r \geq 1$, the nesting hierarchy $\mathcal{L}^{\omega}_{\infty\omega}(Q^r_l)_{l\geq 1}$ does not collapse, as can be seen by reducing this problem to the unary case via the following equivalence:
$(*)$ $Qx\,{}^{r-1}x;\varphi \quad \leftrightarrow \quad Qx;(\neg Q\,{}^{r-1}x;\varphi)$

In contrast to these hierarchies, the nesting can, as far as the above trees are concerned, be simulated by storing information in a *fixed point variable*. This is expressed in (iii) above. Once we have established (i)–(iii), we get the desired result that separates the two notions of *IFP*–reducibility introduced in 4.1.

Theorem 4.3.2. $FO(Q_{IFP}, Q) <$ *IFP*(Q). In fact, *IFP*(Q) is not even contained in $\mathcal{L}^{\omega}_{\infty\omega}(Q^{\omega}_{\omega})$.

Proof: Assume φ from (iii) is equivalent to a formula $\chi \in \mathcal{L}^{k}_{\infty\omega}(Q^r_l)$ for some k, r, l. Due to the equivalence $(*)$ we can assume $r = 1$. But then, $\mathcal{A}_l \models \chi, \mathcal{B}_l \models \neg\chi$, in contradiction to $\mathcal{A}_l \equiv_{k,1,l} \mathcal{B}_l$. $\square$

Now we fix a $k \geq 1$ and construct $\mathcal{A}_l, \mathcal{B}_l$ by induction on l. Rather than (i) we show

$$\text{(i)}^* \ \mathcal{A}_l \equiv^*_{k,1,l} \mathcal{B}_l \text{ for all } l.$$

(i)^* and (i) are equivalent for the even cardinality quantifier due to the equivalence

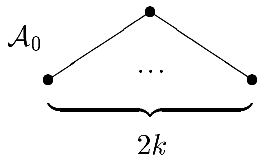
$$Qx, y; (\psi(x), \psi_R(y)) \leftrightarrow Qy; (\psi(y) \wedge \psi_R(y)).$$

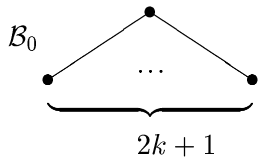
The formula on the right has the same nesting depth of quantifiers Q, and the relativization has been eliminated.

The idea to construct the trees is the following: If $\mathcal{A}_0$ and $\mathcal{B}_0$ consist of a root and an even (resp. odd) number of leaves, then these structures cannot be distinguished in infinitary logic with k variables provided that the number of leaves is bigger than k in both structures. However, one quantifier Q suffices to distinguish them. We iterate this idea by replacing leaves with subtrees. But in the following definition, the trees $\mathcal{A}_l, \mathcal{B}_l$ will not directly replace leaves. Rather they will group into intermediate subtrees, called $\mathcal{E}_l$ and $\mathcal{O}_l$, first. This will make it easier to balance the cardinalities of definable sets in the structures, for instance the number of leaves.

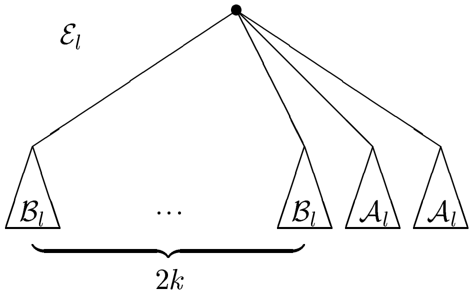
Definition 4.3.3. By induction on $l \geq 0$ we define sequences of trees, $(\mathcal{A}_l)_{l\geq 0}$, and $(\mathcal{B}_l)_{l\geq 0}$.
$\mathcal{A}_0$ is a tree having a single root and $2k$ leaves connected to the root (this is realized by a binary relation E, leading from the leaves to the root). $\mathcal{B}_0$

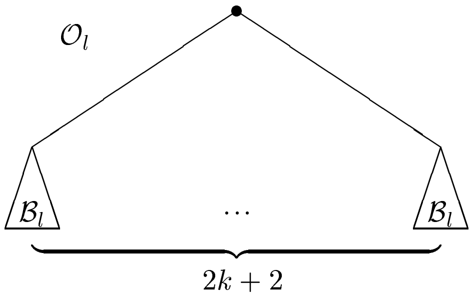
looks like $\mathcal{A}_0$, but has $2k+1$ leaves. (The following illustrations disregard the direction of edges.)



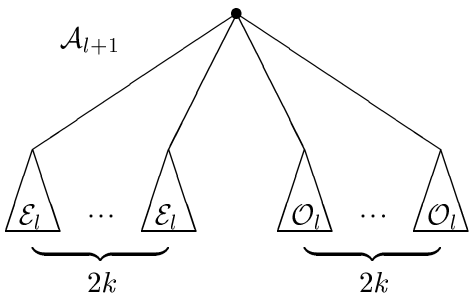


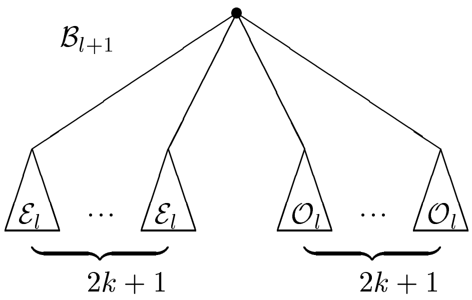
Now, assume that $\mathcal{A}_l$ and $\mathcal{B}_l$ have been constructed. We define two structures $\mathcal{E}_l$ and $\mathcal{O}_l$ ($\mathcal{E}_l$ has an *even* subtree, i.e. one of type $\mathcal{A}_l$, $\mathcal{O}_l$ has only *odd* subtrees of type $\mathcal{B}_l$). Take $2k$ copies of $\mathcal{B}_l$ and 2 copies of $\mathcal{A}_l$ and connect the root of each of these trees to a new root, thus obtaining $\mathcal{E}_l$. $\mathcal{O}_l$ looks similarly, but it consists of $2k+2$ copies of $\mathcal{B}_l$ and a new root.





Finally, let $\mathcal{A}_{l+1}$ be the tree consisting of $2k$ copies of $\mathcal{E}_l$ and $2k$ copies of $\mathcal{O}_l$, all of them connected to a new root. Similarly $\mathcal{B}_{l+1}$, but here we take $2k+1$ copies each time.





$\square$

Recall the definition of the k, r, l–game on two structures given in 4.2. For $r = 1$, a Q–move in this game is started by player I by colouring a set of even cardinality. This is answered by player II with an even set. For the purpose of an inductive proof to be given below, it is better to consider a game where player I may colour a set of any cardinality, and where player II must answer with a set of the same parity. We refer to this game as the *strong* $k, 1, l$–game. Clearly, if player II has a strategy for the strong game, then a fortiori he has a strategy for the usual one.

Lemma 4.3.4. Suppose that player II wins the strong $k, 1, l$–game on $\mathcal{A}_l$ and $\mathcal{B}_l$. Then II also wins the strong $k, 1, l$–game on $\mathcal{E}_l$ and $\mathcal{O}_l$, even if the roots (both in $\mathcal{E}_l$ and $\mathcal{O}_l$) or some elements of a certain copy of $\mathcal{B}_l$ (in $\mathcal{E}_l$ and $\mathcal{O}_l$ respectively) have already been pebbled (in such a way that the pebbled copy of $\mathcal{B}_l$ in $\mathcal{E}_l$ can be mapped isomorphically to that in $\mathcal{O}_l$, respecting the correspondence of all pebbles involved).

Proof: Identify the copies of $\mathcal{B}_l$ in $\mathcal{E}_l$ with any distinct copies of $\mathcal{B}_l$ in $\mathcal{O}_l$ (respecting the given correspondences of pebbles, if there are any) and associate with the copies of $\mathcal{A}_l$ two of the remaining copies of $\mathcal{B}_l$ in $\mathcal{O}_l$. Now, in point–moves, player II can respond to the root or to points in $\mathcal{B}_l \subset \mathcal{E}_l$ canonically via the above identification. For the two $\mathcal{A}_l \subset \mathcal{E}_l$ he answers in the two chosen $\mathcal{B}_l \subset \mathcal{O}_l$ according to his winning strategy on the pair $(\mathcal{A}_l, \mathcal{B}_l)$.
In a Q–move, player I colours some nodes in $\mathcal{E}_l$ red. Player II copies the colourings from the $\mathcal{B}_l \subset \mathcal{E}_l$ to the corresponding trees $\mathcal{B}_l \subset \mathcal{O}_l$. For the pairs $(\mathcal{A}_l, \mathcal{B}_l)$ he uses his winning strategy to transfer the colouring (here we rely on the strong game). Also, in the second part of this move, player II can play either isomorphically or according to his winning strategy. A similar procedure applies for $\neg Q$–moves. $\square$

Proof of the claims (i) – (iii): For (ii) we proceed by induction on l. We set $\varphi_0 := \exists w(\neg\exists x Ewx \wedge QxExw)$. Now assume that φ_l has been defined so that (ii) holds. $\mathcal{A}_{l+1}$ satisfies $\varphi_{l+1} :=$

$$\exists w(\neg\exists x Ewx \wedge Qu(Euw \wedge \exists v(Evu \wedge \text{ the subtree with root } v \text{ satisfies } \varphi_l))),$$

$\mathcal{B}_{l+1}$ does not. Note that this formula can be expressed in the logic required for (ii), since the subtree with root v has the fixed height $2l + 1$, which allows us to define the nodes in this subtree in FO and relativize φ_l to the subtree. (Here, we count the height starting from the leaves, which are said to be of height 0.)

In fixed point logic, we can store the roots of subtrees of type $\mathcal{A}$ up to a certain height in a monadic variable X.

The next level to be tested is then defined by

$$\vartheta(x) := \exists uv(Eux \wedge Evu \wedge Xv)$$

and (iii) holds via
$\varphi := \exists w(\neg\exists x Ewx \wedge [IFP_{x,X}(height(x) = 1 \wedge QyEyx) \vee (\vartheta(x) \wedge Qu(Eux \wedge \exists v(Evu \wedge Xv)))]w)$.

It remains to prove claim (i). Therefore we will define a winning strategy for player II in the strong $k,1,l$–game on $\mathcal{A}_l, \mathcal{B}_l$, again proceeding by induction on l. Then, (i) follows from Theorem 4.2.5, and from the above equivalence of (i) with (i)*.

Clearly, II wins the $k,1,0$–game on $\mathcal{A}_0$ and $\mathcal{B}_0$, since it reduces to the usual k–pebble game.

Assume we have defined a strategy for l. We must show that II wins the $k,1,l+1$–game on $\mathcal{A}_{l+1}, \mathcal{B}_{l+1}$. His strategy before the first Q–move is obvious: If I places, for instance, a pebble on a subtree $\mathcal{E}_l$ not pebbled so far, then II will choose such an unpebbled (free, as we will call it) $\mathcal{E}_l$ in the other structure and answer isomorphically here, now and also for the rest of the game – unless all pebbles are removed from this copy of $\mathcal{E}_l$, in which case we call $\mathcal{E}_l$ free again.

Consider the first Q–move (a $\neg Q$–move can be treated in the same way). Player I chooses $S \subset A_{l+1}$, referred to as the set of *red* elements ($A_{l+1} \setminus S$ is said to consist of *green* ones). We have to define the answer $T \subset B_{l+1}$ of player II, such that $|S| \equiv |T|$ (2).

For the *pebbled* subtrees $\mathcal{E}_l$ and $\mathcal{O}_l$ of $\mathcal{A}_{l+1}$ and for the root, we copy the colouring to the corresponding trees $\mathcal{E}_l$, and $\mathcal{O}_l$, or to the root respectively. If, in the second part of the move, player I pebbles one of these trees or the root, II will answer isomorphically, and the situation is as it was before the first Q–move.

It remains to colour the *free* subtrees $\mathcal{E}_l$ and $\mathcal{O}_l$ in $\mathcal{B}_{l+1}$. For this purpose, we consider *only the free* copies of $\mathcal{E}_l$ and $\mathcal{O}_l$ in $\mathcal{A}_{l+1}$ and their colourings, and distinguish several cases (each time we also state how the second part of the move is to be treated, where we can assume that player I pebbles an element b in one of the *free* subtrees in $\mathcal{B}_{l+1}$):

Case 1: There exist $\mathcal{C}_1, \mathcal{C}_2 \in \{\mathcal{E}_l, \mathcal{O}_l\}$ in $\mathcal{A}_{l+1}$ such that the root of $\mathcal{C}_1$ is red, that of $\mathcal{C}_2$ green.
Player II chooses a free copy $\mathcal{E}_l^*$ of $\mathcal{E}_l$, and a free $\mathcal{O}_l^*$ in $\mathcal{A}_{l+1}$, and copies their colourings to all free trees $\mathcal{E}_l$, and $\mathcal{O}_l$, in $\mathcal{B}_{l+1}$. If the resulting parity of T does not equal that of S, player II may change the colour of an arbitrary root of a free $\mathcal{E}_l$ or $\mathcal{O}_l$. Let, in the second part, player I pebble this root. If it is red, player II pebbles the root of $\mathcal{C}_1$, otherwise that of $\mathcal{C}_2$. If I pebbles an element $b \in \mathcal{E}_l \subset \mathcal{B}_{l+1}$, player II can answer isomorphically in $\mathcal{E}_l^*$. Similarly for $\mathcal{O}_l$ and $\mathcal{O}_l^*$.

Case 2: All the roots of copies of $\mathcal{E}_l$ or $\mathcal{O}_l$ have the same colour, say red. Player II colours all roots of trees $\mathcal{E}_l$, and $\mathcal{O}_l$, red. For these nodes, the parity is correct since we have two more such free subtrees in $\mathcal{B}_{l+1}$. In the second part of the move, a root is answered by a root, of course. For the definition of T on the $\mathcal{A}_l$ and $\mathcal{B}_l$ in the free trees $\mathcal{E}_l$ and $\mathcal{O}_l$ we have to distinguish two cases, depending on the colouring S in $\mathcal{A}_{l+1}$:

Case 2.1: There exist distinct copies $\mathcal{D}_1, \mathcal{D}_2$ of $\mathcal{A}_l$ such that $\mathcal{D}_1$ has an even number of red nodes, $\mathcal{D}_2$ an odd one.

This case (and the analogous one for the $\mathcal{B}_l$) can be treated similarly to case 1: Fix a copy $\mathcal{B}_l^*$ of $\mathcal{B}_l$ in $\mathcal{A}_{l+1}$ (remember, we consider only the free part) and colour all $\mathcal{B}_l$ in $\mathcal{B}_{l+1}$ like $\mathcal{B}_l^*$. Colour all $\mathcal{A}_l$ like $\mathcal{D}_1$, changing one colouring to that of $\mathcal{D}_2$ if necessary for the parity. In the second part, player II can answer isomorphically with respect to the trees $\mathcal{A}_l$, and $\mathcal{B}_l$, perhaps associating an element $a \in \mathcal{B}_l \subset \mathcal{E}_l$ with a $b \in \mathcal{B}_l \subset \mathcal{O}_l$ (or vice versa).

Case 2.2: All trees $\mathcal{A}_l$ have the same number of red nodes (mod 2), and the same holds for the trees $\mathcal{B}_l$.

Here we make use of the fact that the number of trees $\mathcal{A}_l$ (and $\mathcal{B}_l$) in each $\mathcal{E}_l$ and $\mathcal{O}_l$ is even. So the total number of red elements is 0 (mod 2). Player II can fix one $\mathcal{A}_l$ and one $\mathcal{B}_l$ and copy their colourings to all $\mathcal{A}_l, \mathcal{B}_l$ in $\mathcal{B}_{l+1}$. This produces also parity 0 and realizes no further types of colouring, that is, player II can answer the second part of the move with the same result as in Case 2.1.

The result of the first Q–move: In each of the above cases, the elements $a \in \mathcal{A}_{l+1}$, $b \in \mathcal{B}_{l+1}$, pebbled in that move, satisfy the following:

- Either a, b are the roots of $\mathcal{A}_{l+1}$, or $\mathcal{B}_{l+1}$ respectively, or
- they have been chosen in subtrees $\mathcal{C}, \mathcal{D}$ of the same type $\mathcal{E}_l$, or $\mathcal{O}_l$, such that $(\mathcal{C}, a) \simeq (\mathcal{D}, b)$ via an isomorphism that maps all other pebbled elements in $\mathcal{C}$ to the corresponding ones in $\mathcal{D}$, or
- we have $a \in \mathcal{C} \simeq \mathcal{E}_l, b \in \mathcal{D} \simeq \mathcal{O}_l$ and a, b are in subtrees $\mathcal{C}' \simeq \mathcal{B}_l, \mathcal{D}' \simeq \mathcal{B}_l$ with $(\mathcal{C}', a) \simeq (\mathcal{D}', b)$.

In the last case Lemma 4.3.4 applies. Hence the following invariant holds for $\lambda = 1$:

> *Invariant after λ Q–moves:*
> Assume λ Q–moves have been carried out $(1 \leq \lambda \leq l+1)$.
> Then, there exist k subtrees $\mathcal{C}_1, \ldots, \mathcal{C}_k$ of type $\mathcal{E}_l$ or $\mathcal{O}_l$ in $\mathcal{A}_{l+1}$ and k such trees $\mathcal{D}_1, \ldots, \mathcal{D}_k$ in $\mathcal{B}_{l+1}$ with:
>
> (a) All pebbles are placed on the $\mathcal{C}_\nu, \mathcal{D}_\nu$ or on the roots of $\mathcal{A}_{l+1}, \mathcal{B}_{l+1}$.
>
> (b) For all $1 \leq \nu \leq k$: If $\overline{c}, \overline{d}$ are the tuples of elements pebbled in $\mathcal{C}_\nu, \mathcal{D}_\nu$, then II wins the $k, 1, l-\lambda+1$–game on $(\mathcal{C}_\nu, \overline{c})$ and $(\mathcal{D}_\nu, \overline{d})$.

We show that player II can, in each move, preserve this invariant, which concludes the proof. In a *point–move*, if I plays within the $\mathcal{C}_\nu$ or $\mathcal{D}_\nu$, the invariant can clearly be satisfied via the same subtrees. Otherwise one of those trees is set free and can be cancelled from the list. The root, or a free subtree, is being pebbled, and the answer of player II can be carried out on the root, or in an isomorphic copy of this subtree respectively.

In a $(\neg)Q$*–move*, proceed as follows: For the $\mathcal{C}_\nu, \mathcal{D}_\nu$, play according to the winning strategy postulated in (b). For the remaining subtrees of type $\mathcal{E}_l$ or $\mathcal{O}_l$, use the $k, 1, l$–winning strategy from Lemma 4.3.4. □

Let us turn to other variants of fixed point logic. Let Q^{mon} denote the monotone equivalent of Q, as defined in 2.5. This quantifier can be adjoined to LFP. Note that $FO(Q_{IFP}) \equiv IFP \equiv LFP$, whence we can write Q_{LFP} for Q_{IFP}.

Theorem 4.3.5.
(i) $FO(Q_{LFP}, Q^{mon}) \; < \; LFP(Q^{mon})$.
(ii) $FO(Q_{PFP}, Q) \; < \; PFP(Q)$.

Proof: (i) follows from $LFP(Q^{mon}) \equiv IFP(Q^{mon})$ (Remark 2.3.9) and $IFP(Q^{mon}) \equiv IFP(Q)$ (by Theorem 2.5.2). (ii) follows from Theorem 4.3.2 above because Lemma 4.2.2 holds also with Q_{PFP} in place of Q_{IFP} (it can be argued in the same way, because $PFP \leq \mathcal{L}^{\omega}_{\infty\omega}$). □

4.4 Constructing oracles by diagonalization

The results of the previous section cannot be carried over to ordered structures, since the class Q we used to separate $FO(Q_{IFP}, Q)$ from $IFP(Q)$ was $LOGSPACE$–computable. To separate these logics on ordered structures, and hence $PTIME$-many-one-reducibility from $PTIME$-Turing-reducibility (cf. 4.1), requires a class Q that is not even $PTIME$–computable. This is, because for any $PTIME$–computable class Q we have $PTIME^Q \equiv PTIME$, and hence, as the left side coincides with $IFP(Q)$ on ordered structures, $IFP(Q) \equiv_{\mathcal{O}} IFP$.

Under certain complexity theoretical assumptions, it is not hard to find a class Q that satisfies $FO(Q_{IFP}, Q) \; <_{\mathcal{O}} \; IFP(Q)$. Take, for instance Q_{HAM}, the class of graphs that have a Hamiltonian cycle. Then, by results of Stewart [35] and Gottlob [14], we have $FO(Q_{HAM}) \equiv_{\mathcal{O}} LOGSPACE^{NP}$ (see 2.1.4). On the other hand, we have

$$FO(Q_{IFP}, Q_{HAM}) \equiv_{\mathcal{O}} FO(Q_{HAM}).$$

This follows from the $PTIME$–computability of Q_{IFP} and from $PTIME \leq_{\mathcal{O}} FO(Q_{HAM})$. Indeed, the last inclusion guarantees that $Q^{<}_{IFP} \in FO(Q_{HAM})$. But on ordered structures, each formula $Q_{IFP}\overline{x}; \Phi$ (with Φ of an arity $r \geq 1$) is equivalent to $Q^{<}_{IFP}\overline{x}, \overline{y}, \overline{z}; (\Phi, \theta(\overline{y}, \overline{z}))$, where θ is a definition of the lexicographical order of r-tuples based on the given order. [23] Hence we get:

[23] By the results of Dahlhaus [8], the inclusion $NP \; \leq \; FO(Q_{HAM})$ and hence $FO(Q_{IFP}, Q_{HAM}) \equiv FO(Q_{HAM})$ holds even for arbitrary finite structures.

Remark 4.4.1.
$FO(Q_{IFP}, Q_{HAM}) \equiv_{\mathcal{O}} IFP(Q_{HAM})$ if and only if $LOGSPACE^{NP} \equiv PTIME^{NP}$.
$\square$

To avoid any assumptions on complexity classes, we apply a diagonal argument, which is essentially due to J.F. Buss [7]. It was pointed out to the author by Y. Pnueli that this method is useful in the context of fixed point logic. A similar argument had already been used by Pnueli and Makowsky [33] to show that $PTIME^Q$ cannot be captured *uniformly* by logics based on generalized quantifiers (we will state their result at the end of this section).

In the following, let $\tau = \{<\}$, and consider only τ–structures $\mathcal{A}$ ordered by $<^{\mathcal{A}}$. $\mathcal{A}_n$ will denote the ordering $(A_n, <)$, where A_n has n elements. The oracle Q will be a class of σ_Q–structures for $\sigma_Q := \{\prec, R\}$ (R unary) and will be ordered by $\prec$. Note, however, that our restriction to ordered structures refers only to the τ–structures, not to the quantifiers. So, it may very well happen that in $\varphi = Q\overline{x}, \overline{y}_1, \overline{y}_2, \overline{z}; \psi(\overline{x}), \psi_{\prec}(\overline{y}_1, \overline{y}_2), \psi_R(\overline{z})$ the formula $\psi_{\prec}$ does not define an ordering on the domain $\{\overline{a} \mid \mathcal{A} \models \psi[\overline{a}]\}$ for some (ordered) $\mathcal{A}$, in which case φ will simply be false.

Each σ_Q–structure which is ordered by $\prec$ can be regarded as a word $w \in \{0,1\}^*$ (identify Rx with 1, and $\neg Rx$ with 0 at position x).

Theorem 4.4.2. There is a class Q such that $FO(Q_{IFP}, Q) <_{\mathcal{O}} IFP(Q)$ holds. Hence $FO(Q_{IFP}) \equiv_{\mathcal{O}} PTIME$ does not 'relativize' to $FO(Q_{IFP}, Q) \equiv_{\mathcal{O}} PTIME^Q$.

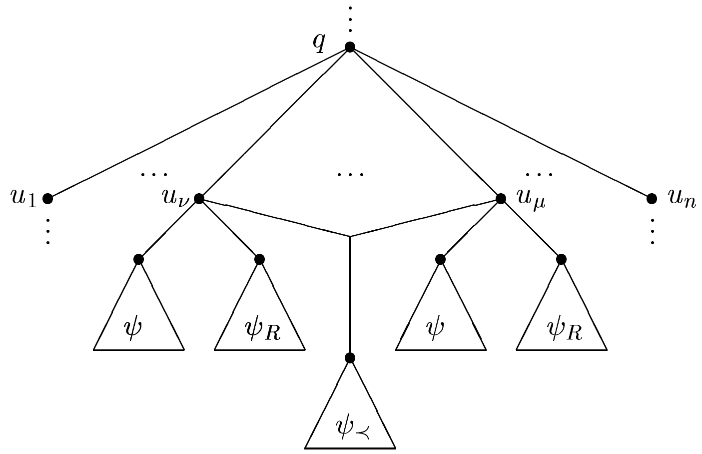
Proof: We are going to construct a class Q of strings which gives rise to a class $\mathcal{K}(Q)$ of pure orderings such that $\mathcal{K}(Q) \in IFP(Q)[\tau] \setminus FO(Q_{IFP}, Q)[\tau]$. $\mathcal{K}(Q)$ is obtained from Q by feeding the oracle successively to itself, starting from the word 0 and 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 $s(Q)|_\nu := s(Q)_1 \dots s(Q)_\nu \in Q$, $s(Q)_{\nu+1} = 0$ 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, for $\mathcal{A}_n$, subsequently check the initial segments of $s(Q)$ up to length $n-1$. (Alternatively, one can easily write down a $PTIME^Q$–machine accepting $\mathcal{K}(Q)$ and make use of the equivalence $PTIME^Q \equiv_{\mathcal{O}} IFP(Q)$.) 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 $\varphi_1, \varphi_2, \dots$ be an enumeration of $FO(Q_{IFP}, Q)[\tau]$. We aim for a Q and a sequence $n_1 < n_2 < \dots$ 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:

$$\mathcal{A}_{n_i} \models \varphi_i \text{ iff } \mathcal{A}_{n_i} \in \mathcal{K}(Q) \text{ iff } s(Q)_{n_i} = 1 \text{ iff } \mathcal{A}_{n_i} \not\models \varphi_i.$$

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 Section 2.2. 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 $Qx, y_1, y_2, z; \psi(x), \psi_{\prec}(y_1, y_2), \psi_R(z)$, can 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 $\mathcal{A} \models \psi_{\prec}[u_\nu, u_\mu]$ (a Q–node can also be of higher arity r, in which case we would 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 $\psi_{\prec}$– 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. We call it therefore the *Q–output* of $\mathcal{C}_n^i$. The particular 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 on 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

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

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

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 $i \geq 1$:

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

Then (1) is also guaranteed, since $s(Q_i)_{n_i} = s(Q)_{n_i}$ (by consistency) and the Q–output of $\mathcal{C}^i_{n_i}$ is computed in the same way as the Q_i–output due to the choice of d_i. In fact, this output only depends on Q–queries of length up to $n_i^{d_i}$. Arguing in $\mathcal{C}^1_{n_1}$, 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 to be in Q_1, and we determine 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 $\mathcal{C}^1_{n_1}$, then the bit $s(Q_1)_{n_1}$ can be chosen freely without influencing the Q_1–output in $\mathcal{C}^1_{n_1}$.

First step: Let $\{w^1_1, \dots, w^1_{r_1}\}$ be the set of strings checked by Q–nodes on level 1 (this set of strings does not depend on Q). Since $r_1 \leq n_1^{d_1}$ (a bound for the size of $\mathcal{C}^1_{n_1}$), 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 of the strings w^1_ν (determine v_1 via majority vote: if more than half of the w^1_ν 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 proper initial segments of $0v_1$ we let $s(Q_1)|_{|v_1|+1} = 0v_1$. If, for instance, v_1 starts with $101\dots$, then we let $0 \in Q_1$,$01 \notin Q_1$, $010 \in Q_1$ and so on. Paying attention only to these decisions we determine the w^1_ν to be elements of Q_1 or not (note that the w^1_ν may very well be initial segments of $0v_1$), which gives us $\{w^2_1, \dots, w^2_{r_2}\}$, 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 of the w^2_ν, 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 $w \notin \{w^1_1, \dots, w^1_{r_1}\}$ for these w, whence we are free to choose.

Continuing this way, we have, *after* l_1 *steps*, determined the Q_1–output of $\mathcal{C}^1_{n_1}$, and we have chosen $v_1, \dots, v_{l_1}$ with $|v_\nu| \leq d_1 \log n_1$ such that $0v_1 \dots v_{l_1}$ is not a prefix of any of the tested words and $s(Q_1)$ begins with $0v_1 \dots v_{l_1}$. By (2), we have $|0v_1 \dots v_{l_1}| \leq 1 + d_1^2 \log n_1 < n_1$. Now, considering only words with prefix $0v_1 \dots v_l$, we choose Q_1 consistent with the decisions made so far. We do this in a way which makes (3) hold for $i = 1$ (note that 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 $v'_1, \dots, v'_{l_{i+1}}$ such that $s(Q_{i+1})$ begins with $s(Q_i)|_{n_i^{d_i}} v'_1 \dots v'_{l_{i+1}}$, 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. $\square$

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

By diagonal arguments of the same kind as in the last proof, one gets sharp conditions for logics to characterize oracle complexity classes.

Theorem 4.4.3. **a)** For each countable family $(Q_\nu)_{\nu \in I}$ of quantifiers, there

exists a class Q of strings such that $PTIME^Q \not\leq_{\mathcal{O}} FO((Q_\nu)_{\nu \in I}, Q)$.
b) There exists a class Q of strings such that $PTIME^Q$ is not captured by $IFP(Q)$ without domain relativization in the Q–rule.
c) The analogous statements to a) and b) hold for PFP and $PSPACE$ (no matter, whether the oracle tape is subject to a polynomial space bound or not).
d) There is a monotone quantifier Q such that $FO(Q_{LFP}, \tilde{Q}) <_{\mathcal{O}} LFP(\tilde{Q})$.

Proof: **a)** The proof of Theorem 4.4.2 did not depend on the form of the additional nodes in the $\mathcal{C}_n^i$.
b) The last proof depends only on the fact that a certain word of length $n_i - 1$ is not tested in $\mathcal{C}_{n_i}^i$. 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 has to be adapted to PFP). Hence we only make use of the fact $PTIME^Q \leq PSPACE^Q$, which holds in any reasonable oracle model.
d) Let $\tilde{Q} := Q^{mon}$ be the monotone equivalent of the class of strings, Q, constructed for the last theorem (for the monotone equivalent and its incorporation into fixed point logic, see 2.5.2 and 2.3.9). Then $FO(Q_{LFP}, \tilde{Q}) \equiv FO(Q_{LFP}, Q) <_{\mathcal{O}} IFP(Q) \equiv LFP(\tilde{Q})$. □

Summing up, in the known characterization of $PTIME^Q$ by $IFP(Q)$ (or $LFP(Q)$ for monotone Q) one can neither drop relativization of the domain, nor replace the second order constructs by countably many Lindström quantifiers. The same holds for $PSPACE^Q$ and $PFP(Q)$. Remember however that for each *fixed* oracle Q, the classes $PTIME^Q$ and $PSPACE^Q$ (in a bounded oracle model) can very well be characterized by Lindström logics, namely by $FO(\hat{Q}_C^Q)$ (see 2.2), and the analogous logic for the case of PFP.

For the following corollary of Theorem 4.4.3 a), note that the syntax of $FO(Q)$ is defined as soon as σ_Q is fixed. Thus Q can be regarded as varying, similar to its role in oracle Turing machines. Given a logic $\mathcal{L}$, we can then say that $\mathcal{L}(Q)$ *captures* $PTIME^Q$ *uniformly*, if there is a mapping f associating to each $PTIME$ oracle machine M a formula $f(M) \in \mathcal{L}(Q)$ such that $f(M)$ and M define the same query for each Q.

Corollary 4.4.4. **[33]** Let σ_Q contain a unary R and a binary $\prec$. For each countable family $(Q_\nu)_{\nu \in I}$ of quantifiers, $FO((Q_\nu)_{\nu \in I}, Q)$ does not capture $PTIME^Q$ uniformly. □

4.5 The unordered case revisited

Comparing the last two sections, one might argue that on the one hand only the case of ordered structures is of interest for the results concerning descriptive complexity, whereas on the other hand it seems difficult to apply the tool of Ehrenfeucht-Fraïssé games to the ordered case. We will now show that for some quantifiers Q the question of whether or not $FO(Q_{IFP}, Q)$ equals $IFP(Q)$ is of computational relevance even in the unordered case.

Recall Remark 4.4.1 stating the following:

$$FO(Q_{IFP}, Q_{HAM}) \equiv_{\mathcal{O}} IFP(Q_{HAM}) \text{ iff } LOGSPACE^{NP} \equiv PTIME^{NP}.$$

The same holds for any class Q in place of Q_{HAM} provided that Q is NP–complete under first order reductions (on ordered structures). For instance, it holds for the class Q_{SAT} which we proved to be NP–complete under quantifier free reductions (even for arbitrary structures; see 2.4). For Q_{SAT} we will establish a statement that comes close to the above equivalence but deals with arbitrary structures on the left side. (The statement is also true for Q_{HAM} as, by a result of Dahlhaus [8], Q_{HAM} is NP–complete under quantifier free reductions that require only two distinctly interpreted constants; since we did not study this type of reduction, we chose Q_{SAT} for our exposition.)

Theorem 4.5.1.
$FO(Q_{IFP}, Q_{SAT}) \equiv IFP(Q_{SAT})$ implies $LOGSPACE^{NP} \equiv PTIME^{NP}$.
$FO(Q_{IFP}, Q_{SAT}) < IFP(Q_{SAT})$ implies $NP < PTIME^{NP}$.

Proof: The first implication is clear as the equivalence of logics carries over to the ordered case and one can argue as for Remark 4.4.1.
For the second one, observe that we have $NP \leq FO(Q_{SAT})$ and $IFP(Q_{SAT}) \leq PTIME^{NP}$. Hence, assuming $FO(Q_{IFP}, Q_{SAT}) < IFP(Q_{SAT})$, we find that $NP < PTIME^{NP}$. $\square$

Note that in $FO(Q_{IFP}, Q_{SAT})$ we could eliminate Q_{IFP} as this class is definable in $FO(Q_{SAT})$. We only kept Q_{IFP} for uniformity reasons.

Remember that by results of Gottlob and Stewart, $FO(Q_{SAT})$ captures $LOGSPACE^{NP}$ on ordered structures. It is not known whether this remains true without the presence of an ordering. Although we have $NP \equiv \Sigma_1^1 \leq FO(Q_{SAT})$, this does not mean that $FO(Q_{SAT})$ is closed under existential second order quantification, which would allow us to quantify over an ordering. To the contrary, we have the following.

Theorem 4.5.2. There is a class Q such that
(i) $NP \leq FO(Q)$,
(ii) $FO(Q)$ is not closed under existential second order quantification, that is to say there is a sentence $\exists X \psi(X)$ (with $\psi(X) \in FO(Q)$) which is not equivalent to any $FO(Q)$–sentence.

Proof: By Theorem 4.4.3 a), applied to Q_{SAT}, there is a class $\tilde{Q}$ such that $IFP(\tilde{Q}) \setminus FO(Q_{SAT}, \tilde{Q}) \neq \emptyset$. Take for Q a quantifier with $FO(Q) \equiv FO(Q_{SAT}, \tilde{Q})$ (it is a standard technique to encode two quantifiers by a single one; a proof for the non-vectorized case can be found in [12], Chapter 10; that proof can easily be modified so as to include the relativized vectorizations). Then we obviously have $NP \leq FO(Q)$. To see that (ii) also holds, observe that $IFP(\tilde{Q}) \leq IFP(Q)$ and that $IFP(Q)$ is contained in the existential second order closure of $FO(Q)$ (for this inclusion, use the normal form where only one fixed point operator occurs, and translate this operator into an existential second order quantifier). □

Open problems:
1. Does $FO(Q_{SAT})$ strongly capture $LOGSPACE^{NP}$?
2. Does $IFP(Q_{SAT})$ strongly capture $PTIME^{NP}$?

Note that a positive answer to the first question would yield the analogous equivalence to that of Remark 4.4.1 for the case of arbitrary finite structures.

References

[1] S. Abiteboul and V. Vianu. Fixpoint extensions of first order logic and datalog-like languages. In *Proceedings of the 4th IEEE Symposium in Logic in Computer science*, pages 71–79, 1989.

[2] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proceedings of 23rd ACM symposium on theory of computing*, pages 209–219, 1991.

[3] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of pure and applied logic*, 24:1–48, 1983.

[4] D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[5] J. Barwise. On moschovakis closure ordinals. *Journal of Symbolic Logic*, 42:292–296, 1977.

[6] U. Bosse. *Zur Modelltheorie der Fixpunktlogik.* PhD thesis, Universität Freiburg, 1994.

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

[8] E. Dahlhaus. Reductions to np-complete problems by interpretations. In E. Börger et al., editor, *Logic and Machines: Decision Problems and Complexity, Lecture Notes in Computer Science 171*, pages 357–365. Springer-Verlag, 1983.

[9] 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.

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

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

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

[13] R. Fagin. Generalized first–order spectra and polynomial–time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, Vol. 7*, pages 43–73, 1974.

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

[15] 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.

[16] E. Grädel and G. McColm. Hierarchies in transitive closure logic, stratified datalog and infinitary logic. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 167–176, 1992.

[17] E. Grandjean. Universal quantifiers and time complexity of random access machines. *Mathematical System Theory*, 13:171–187, 1985.

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

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

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

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

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

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

[24] N. Immerman. Dspace$[n^k]$ = var$[k+1]$. In *Proc. 6th IEEE Symp. on Structure in Complexity Theory*, pages 334–340, 1991.

[25] Ph. G. Kolaitis and J. Väänänen. Generalized quantifiers and pebble games on finite structures. *Annals of Pure and Applied Logic*, 74:23–75, 1995.

[26] Ph. G. Kolaitis and M. Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *Proc. 7th IEEE Symp. on Logic in Computer Science*, 1992.

[27] R.E. Ladner and N.A. Lynch. Relativizations of questions about log space computability. *Mathematical Systems Theory*, 10:19–32, 1976.

[28] R.E. Ladner, N.A. Lynch, and A.L. Selman. A comparison of polynomial time reducibilities. *Theoretical Comp. Sci.*, 1:103–123, 1975.

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

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

[31] J. F. Lynch. Complexity classes and theories of finite models. *Mathematical System Theory*, 15:127–144, 1982.

[32] J.A. Makowsky and Y.B. Pnueli. Arity vs. alternation in second order logic. 1993. To appear.

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

[34] C. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

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

[36] 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.

[37] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[38] J. Väänänen. Set-theoretic definability of logics. In J. Barwise and S. Feferman, editors, *Model-Theoretic Logics*, pages 599–643. Springer-Verlag, 1985.

[39] M. Vardi. The complexity of relational query languages. In *Proceedings of 14th ACM symposium on the theory of computing*, pages 137–146, 1982.

[40] Weese. Generalized ehrenfeucht-fraïssé games. *Fundamenta Mathematicae*, 109:103–112, 1980.