The Kolmogorov Expressive Power of Boolean Query Languages*

Jerzy Tyszkiewicz†
Mathematische Grundlagen der Informatik
RWTH Aachen,
Ahornstraße 55
D-52074 Aachen
Germany.
e-mail `jurek@informatik.rwth-aachen.de`

July 2, 1996

Abstract

We address the question "How much of the information stored in a given database can be retrieved by all Boolean queries in a given query language?".

In order to answer it we develop a Kolmogorov complexity based measure of expressive power of Boolean query languages over finite structures. This turns the above informal question into a precisely defined mathematical one. This notion gives a meaningful definition of the expressive power of a Boolean query language in a single finite database.

The notion of *Kolmogorov expressive power* of a Boolean query language L in a finite database $\mathbf{A}$ is defined by considering two values: the Kolmogorov complexity of the isomorphism type of $\mathbf{A}$, equal to the length of the shortest description of this type, and the number of bits of this description that can be reconstructed from truth values of all queries from L in $\mathbf{A}$. The closer is the second value to the first, the more expressive is the query language.

After giving the definitions and proving that they are correct, we concentrate our efforts on first order logic and its powerful extensions: inflationary fixpoint logic and partial fixpoint logic. We explore some connections between the proposed Kolmogorov expressive power of Boolean queries in these languages and their standard expressive power, in particular with the definability of order. We show that, except of being of interest for its own, our notion may have important diagnostic value for database query optimization.

1991 Mathematics Subject Classification: Primary 68P15, 68P20, 68Q30; Secondary 03C13, 03C80.

Keywords and phrases: Kolmogorov expressive power, Kolmogorov complexity, first order logic, least fixpoint logic, partial fixpoint logic, average case complexity of query evaluation.

---

*Preliminary version of this paper appeared in the form of a conference paper [22].

†The author has been supported by the Polish KBN grant 2 P301 009 06 and by the German DFG.

1 Introduction

1.1 Database as a black hole

When a user of a database system stores some data in it, he hopes to be able to access it at any time. Many different tools have been invented to ensure it and protect the information stored in databases. We use UPS's as a prevention against interrupts in power supply, disk arrays allowing the recovery of data in a case of a disk crash, we do frequent backups, sophisticated security systems have been developed to protect the data from unauthorized access. Database systems are getting more and more secure. There is, however, one more place where the problems may come from. This is the query language. The lack of its expressive power can turn a database into a black hole—the information can be stored in it, it is there, but we cannot access it any more. Is it a real danger? Most of general query languages allow writing the identity query, which, although practically rather useless, from theoretical standpoint is a perfect way to get all the information back. But if our query language is Boolean, like in many expert systems? Well, then the answer is not so obvious any more.

1.2 The Aim of the Paper

Our intention in this paper is to tackle the black hole problem. The notion of *Kolmogorov expressive power* we define indicates how large portion of the collected data can be retrieved by Boolean queries. The rest, due to the limitations of expressive power of the query language, remains "invisible" to the user.

Our first result is then addressing the black hole problem in a clean mathematical way. We feel that the quantity of information retrievable from a given database by a given query language is an important expressiveness measure, and should be considered when designing query languages and databases

For a logician, like the author himself, it is also a meaningful notion of the expressive power of a logic in a *single* finite structure. Recently, the main theoretical efforts in the finite model theory are shifting from considering the expressive power of logics over classes of ordered finite structures, to the unordered ones, and from the class of all finite models to particular, restricted classes, arising from combinatorics and database theory. In this direction we reach the limit, being able to deal with one fixed structure. In this paper we restrict our attention to sentences, i.e., to Boolean query languages.

The main idea is quite simple: we measure, in the terms of Kolmogorov complexity, how much of the isomorphism type of the finite structure can be reconstructed from results of evaluation of all Boolean queries.

In the paper, apart from giving all the necessary definitions and proving that they are correct, we begin the study of the Kolmogorov expressive power of the fixpoint extensions of first-order logic. Since even the existential fragment of first-order logic with unrestricted number of variables appears to be too strong for our purposes, we deal with finite-variable fragments of these logics. We prove interesting connections of the introduced notion and the standard descriptive complexity of logics.

We describe also an example, showing that by theoretical investigations one can obtain nontrivial results involving the proposed notion. It concerns mainly the interactions of the Kolmogorov expressive power of first-order queries and the *split* technique for optimizing fixpoint and partial fixpoint queries, demonstrated in [1, 3].

From our results it follows, that any speedup of query evaluation obtained by split always indicates that the Kolmogorov expressive power of the query language is limited. In the model considered in [1], where the authors proved surprisingly low average complexity results for these query languages, the Kolmogorov expressive power is extremely low. We then extend that model in a simple way, getting a broad spectrum of behaviours of both computational complexity, and

Kolmogorov complexity of the considered query languages. Our results suggest, that when the Kolmogorov expressive power is much lower than the theoretical upper bounds implied by the speed-up achieved by split, it is possible to obtain even more effective parallel query evaluation than with split itself.

1.3 The Intuitive Basis

Suppose that we have a query language L, consisting of Boolean queries only, and a finite structure $\mathbf{A}$. Then if **it is not possible** to reconstruct the isomorphism type of $\mathbf{A}$ from results of evaluation of all queries in L, then certainly some of the information about the structure cannot be retrieved. For we could change $\mathbf{A}$ into a nonisomorphic structure $\mathbf{A}'$, preserving the results of evaluation of all queries. So at least the information corresponding to the difference between isomorphism types of $\mathbf{A}$ and $\mathbf{A}'$ is lost. On the other hand, if **it is possible** to reconstruct the finite structure from the queries evaluation results, then all of the information is accessible. Of course, nobody wants to perform such a reconstruction, but the theoretical possibility of doing so is a guarantee that the user has unlimited access to all the collected data. But so far this is only a qualitative notion.

Now we want to turn the above notion into a quantitative one, giving rise to a stratification of expressive powers of query languages with respect to a fixed finite $\mathbf{A}$: *the more of the isomorphism type of* $\mathbf{A}$ *can be reconstructed, the more expressive is the query language.* What remains to be decided, is the unit of measure for this kind of expressive power. We choose it to be a *bit* in the sense of *Kolmogorov complexity.*

However, we will not be able to express precisely the qualitative notion in terms of the quantitative one. So they should be considered as two independent (but closely related) types of inexpressibility statements. See remark at the end of Section 3.

2 Preliminary Notions and Notation

2.1 Kolmogorov Complexity

The discussion of the Kolmogorov complexity is to be found in the book [19] or in the article [18]. We sketch briefly the basic notions, following these sources. However, the exposition is slightly changed, to meet our specific needs. The notation is taken from the book.

Proviso. The set $\mathbf{N}$ of natural numbers is identified with the set $\{0,1\}^*$ of finite binary strings, in the way defined by ordering $\{0,1\}^*$ by increasing length first, and then lexicographically in each length class. In particular, $0 \in \mathbf{N}$ is the empty word, but we will use rather Λ for it, since otherwise it would be difficult to distinguish the empty word 0 and the 1-letter word 0. We will use $lh(x)$ to denote the length of the word x. We use $\langle \cdot, \cdot \rangle$ to denote a recursive pairing function on binary strings. At the moment the choice of this function is inessential; later on we will introduce a special one, particularly well-suited for our purposes.

The logarithmic function is defined as follows: $\log n$ means always the greatest natural m such that $2^m \leq n$. To denote functions we use λ-notation, i.e., $\lambda x.f(x)$ is the name of a function whose value for argument x is $f(x)$.

Let $\{0,1\}^{\leq\omega}$ stand for the union $\{0,1\}^{\mathbf{N}} \cup \{0,1\}^*$, i.e., the set of all finite and infinite binary strings.

Let $\psi(\cdot,\cdot)$ be any partial-recursive function $\{0,1\}^* \times \{0,1\}^{\leq\omega} \to \{0,1\}^*$. The second argument for ψ can be infinite. This may be understood that ψ is computed by a Turing machine, whose first input tape contains the first argument, and the second input tape the second argument; therefore it does not cause any problem, if any of them is infinite.

Definition 2.1. We define the *Kolmogorov complexity of a string* $x \in \{0,1\}^*$ *relative to a string* $y \in \{0,1\}^{\leq\omega}$ *via decoding function* ψ to be

$$C_\psi(x \mid y) = \min\{lh(z) : \psi(z, y) = x\}.$$

According to a widely used convention we assume $\min \emptyset = \infty$.

$C_\psi(x \mid y)$ says how many bits we must add to y in order to describe x uniquely, where the method of understanding descriptions is given by ψ. □

Let $\phi(\cdot,\cdot)$ be any universal partial-recursive function $\{0,1\}^* \times \{0,1\}^{\leq\omega} \to \{0,1\}^*$.

Theorem 2.2. *For every partial-recursive function* $\psi : \{0,1\}^* \times \{0,1\}^{\leq\omega} \to \{0,1\}^*$ *there exists a constant* c_ψ *such that for all* $x \in \{0,1\}^*$, $y \in \{0,1\}^{\leq\omega}$

$$C_\phi(x \mid y) \leq C_\psi(x \mid y) + c_\psi.$$

□

The above theorem, called The Invariance Theorem, justifies the following definition:

Definition 2.3. The *Kolmogorov complexity of a string* $x \in \{0,1\}^*$ *relative to a string* $y \in \{0,1\}^{\leq\omega}$ is defined as $C(x \mid y) = C_\phi(x \mid y)$.

The *Kolmogorov complexity of a string* $x \in \{0,1\}^*$ is defined as $C(x) = C(x \mid \Lambda)$. □

Definition 2.4. We extend the definition of C_ϕ for infinite recursive strings $x = x_0, x_1, x_2 \ldots \in \{0,1\}^{\mathbf{N}}$ in the following way:

$$\widehat{C}_\phi(x \mid y) = \min\{lh(z) : \forall n \in \mathbf{N}\, \phi(\langle z, n\rangle, y) = x_n\}.$$

We set for x as above

$$\widehat{C}(x \mid y) = \widehat{C}_\phi(x \mid y).$$

This means that $\widehat{C}(x \mid y)$ is equal to the length of the shortest program (for ϕ) that, given y, computes x as a function. Similarly as for finite strings we set also

$$\widehat{C}(x) = \widehat{C}(x \mid \Lambda).$$

□

The Invariance Theorem 2.2 can be easily extended for the above $\widehat{C}$, and therefore the definition is also correct. (Compare Exercise 2.20 in [19].) Note that $\widehat{C}$ depends not only on the choice of ϕ, but on the choice of the paring function $\langle\cdot,\cdot\rangle$, as well. So the invariance should be understood and proven with respect to the change of both.

Formally C for finite strings and $\widehat{C}$ for infinite strings are different functions. But we will write in the sequel C for both—their domains are disjoint, and therefore it will be always clear from the context, which function we mean.

The point here is that the Kolmogorov complexity of a string depends on that string, and *the choice of the function* ϕ. Theorem 2.2 says that for every other possible choice the value of the complexity does not change more than by an additive constant. Therefore $C(x \mid y)$ captures the intuitive notion of the shortest possible description of x, given y, up to an additive constant. This means as well, that the complexity is determined "up to an additive constant term", and at this price we can take any universal Turing machine in place of ϕ. This allows us to accept some conventions concerning proofs in the paper. Namely, nothing prevents us from assuming that ϕ is actually a Pascal (C, LISP, ...) interpreter, which runs programs on the data coming from the outside world, say stored in a file accessible for programs. So we can describe Turing machines in our proofs just by Pascal programs. To help the reader we will write programs for ϕ in `this font`.

Now we introduce our pairing function for binary strings. We have already used $\langle x, y\rangle$ as *a* pairing function, but now we need a good one, in the sense that the length of the pair made of x and y is not much greater than $lh(x) + lh(y)$. Concatenation as a pairing function doesn't work at all: if $x, y \in \{0,1\}^*$ then xy doesn't allow us to reconstruct x and y, because we don't know where x ends and y begins. To remedy this we introduce the self-delimiting version of x. First write $\bar{x}$ for the string $\underbrace{1\ldots1}_{lh(x)}0x$, from which x can be uniquely reconstructed.

$$pair(x, y) := \overline{lh(x)}xy$$

is then our pairing function. The reconstruction of x and y can be done as follows: the sequence of 1's at the beginning tells us the length of the binary encoding of the length $lh(x)$ of x (remind that numbers are binary strings as well), then we can read this encoding and learn how long x is, what allows us to separate it from y. The length of $pair(x, y)$ is $lh(x) + lh(y) + 2lh(lh(x)) + 1 = lh(x) + lh(y) + O(\log(lh(x))$.

We prove four simple but fundamental facts about complexity of finite strings: we will use them and they will help us to get used to the typical arguments in proofs about Kolmogorov complexity.

Proposition 2.5. *1. If $D \subseteq \mathbf{N}$ is a finite set, then for every $y \in \{0,1\}^{\leq\omega}$ there exists $x \in D$ such that*

$$C(x \mid y) \geq \log|D|.$$

2. For every $y \in \{0,1\}^{\leq\omega}$ and arbitrary $n \in \mathbf{N}$ there is a string $x \in \{0,1\}^n$ such that

$$C(x \mid y) \geq n.$$

3. If $a = a_0, a_1, \ldots$ is a recursive sequence of finite binary strings, then there is a constant c such that for every n and every $y \in \{0,1\}^{\leq\omega}$ holds

$$C(a_n \mid y) \leq \log n + c.$$

4. There is a constant c such that for every $y \in \{0,1\}^{\leq\omega}$ and every $x \in \{0,1\}^n$

$$C(x \mid y) \leq n + c.$$

Proof. 1. Let $|D| = n$. Since there are only $2^{\log n} - 1 \leq n - 1$ binary strings shorter than $\log n$, there are at most $n - 1$ strings $x \in D$ having complexity $C(x \mid y) < \log n$. (Namely those which are the values $\phi(\texttt{prog}, y)$ for some `prog` shorter than $\log n$.) There are n strings in D, hence at least one of them must have complexity $\geq \log n$.

2. Set $D = \{0,1\}^n$ in Item 1.

3. Let `prog` be a program which computes the function $\lambda n.a_n$. Then the Pascal program `print(p(`n`))` of length $\log n + c$ confirms $C(a_n \mid y) \leq \log n + c$.

4. Set $a = \Lambda, 0, 1, 01, \ldots$ in Item 3. □

In the above proof one can observe a typical phenomenon of Kolmogorov complexity proofs. Namely upper bounds of type $C(x \mid y) \leq s$ should be proven by presenting a binary program for ϕ of length at most s, which computes x, given y. And usually it appears to be a constant-length program with several pieces of binary data embedded in it. In Item 3. above it is n. To estimate the length of the whole one uses the *pair* function we have recently introduced.

3 The Kolmogorov Expressive Power

3.1 Relational model

We present below the version of Kolmogorov expressive power for the (flat) relational model. This can be done in the same way for any other database model we might want to consider.

A *signature* (typically σ) is a finite collection of relation symbols, each one with a fixed arity. We work exclusively with finite structures (typically $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$) over σ. The universe set A of $\mathbf{A}$ (we use the corresponding latin letter) is always some initial segment of the natural numbers. The cardinality of A is denoted $|A|$. If we want to distinguish a tuple $\bar{a} = \langle a_1, \ldots, a_k \rangle$ of elements in $\mathbf{A}$, we denote the result $\langle \mathbf{A}, \bar{a} \rangle$ and understand it as a structure over signature expanded by k constant symbols $x_1, \ldots, x_k$ (always the same), whose values are $a_1, \ldots, a_k$.

A good reference for the theory of queries is the book [2]. A *query* (typically φ) is a mapping from structures over σ into the structures over some other signature τ, preserving the universe of the structures, computable, and *generic*: for any isomorphism $\alpha : \mathbf{A} \to \mathbf{B}$, it must be the case that $\varphi(\alpha(\mathbf{A})) = \alpha(\varphi(\mathbf{A}))$. A query φ is called *Boolean* if τ consists of exactly one predicate, which is 0-ary. In this paper, we usually do not distinguish between queries and logical formulae, by which they are expressed. Thus a Boolean query is just a sentence. The complexity classes of queries are defined in the natural fashion, based on the time and space complexity of the associated functions on encodings, which are defined below.

The following definitions apply to any such language, **consisting of total queries, only.** If we allowed non-terminating queries, it would be difficult to prevent deducting properties of the database from their non-termination, which seems extremely unnatural.

Definition 3.1.

1. σ is a finite, purely relational signature.
2. Σ is the class of all isomorphism types of finite structures over σ. Although formally elements of Σ are classes $[\mathbf{A}]_{\cong}$, it is convenient to work with representatives and assume that elements of Σ are structures themselves.
3. L is a recursive Boolean query language, i.e., a recursive set of total recursive queries.
4. We fix some recursive bijective enumeration $\ell : \mathbf{N} \to L$ of all queries in L.
5. enc is any recursive encoding $\mathrm{enc} : \Sigma \to \{0,1\}^*$.
6. If $\varphi \in L$ and $\mathbf{A} \in \Sigma$, then $\varphi[\mathbf{A}] \in \{0,1\}$ is the truth value of φ in $\mathbf{A}$.
7. If $\mathbf{A} \in \Sigma$, then $L[\mathbf{A}] \in \{0,1\}^{\mathbf{N}}$ is the function (or, equivalently, an infinite binary string) $\lambda n.\ell(n)[\mathbf{A}]$.
8. We fix a *recursive query evaluation function* eval : $\mathbf{N}^2 \to \{0,1\}$ such that $\varphi[\mathbf{A}] = \mathrm{eval}(\ell^{-1}(\varphi), \mathrm{enc}(\mathbf{A}))$. □

Definition 3.2. The tuple made of all the following components: algorithm for identifying binary words and natural numbers, algorithm for enumeration ℓ, algorithm of the encoding function enc of structures as binary words we denote Rep.

For $\mathbf{A} \in \Sigma$ we set

$$C^{\mathrm{Rep}}(\mathbf{A}) = C(\mathrm{enc}(\mathbf{A}))$$

and

$$I^{\mathrm{Rep}}(\mathbf{A}) = C^{\mathrm{Rep}}(\mathbf{A}) - C(\mathrm{enc}(\mathbf{A}) \mid L[\mathbf{A}]).$$

The numbers $C^{\mathrm{Rep}}(\mathbf{A})$ and $I^{\mathrm{Rep}}(\mathbf{A})$ give the information about expressive power of L in $\mathbf{A}$, yielding both the real information content of $\mathbf{A}$ and the quantity of information that can be retrieved from $\mathbf{A}$ by queries in L. Of course, these quantities depend *a priori* on all the parameters fixed in Rep. □

Before we start investigating properties of the Kolmogorov expressive power of query languages, we should first demonstrate that the above defined quantities do not depend on the choices we have made: of the encoding function enc to represent finite structures as words, of the enumeration ℓ of queries in L, etc.

Theorem 3.3. *For any other than* Rep *system* Rep′ *representing all structures over* σ *and the query language* L *in the sense of Definition 3.2, there is a universal constant* c *such that for every structure* $\mathbf{A} \in \Sigma$

$$|I^{\mathrm{Rep}}(\mathbf{A}) - I^{\mathrm{Rep}'}(\mathbf{A})| \leq c \quad \textit{and} \quad |C^{\mathrm{Rep}}(\mathbf{A}) - C^{\mathrm{Rep}'}(\mathbf{A})| \leq c.$$

Proof. The proof is based on the obvious observation, that one can recursively translate all the encodings, enumerations, etc., forming the system Rep into ones forming the system Rep′, and *vice versa.* Hence to get e.g. programs for ϕ adequate for system Rep from those for Rep′ we need to add to the latter a finite length input/output interface performing the translations. □

Since all of Kolmogorov complexity is defined up to an additive constant term, we deduce that the choices we have made in Definition 3.1 are inessential.

This in turn justifies the following definition.

Definition 3.4 (The Kolmogorov expressive power).

The *Kolmogorov complexity of a finite structure* $\mathbf{A} \in \Sigma$ is defined as

$$C(\mathbf{A}) = C^{\mathrm{Rep}}(\mathbf{A}),$$

and the *information expressible in* L *about* $\mathbf{A}$ as

$$I_L(\mathbf{A}) = I^{\mathrm{Rep}}(\mathbf{A}).$$

By the *Kolmogorov expressive power of* L we understand these two values among all structures in Σ.

The value $C(L[\mathbf{A}])$ will be of interest for us, as well. Note, that it is always finite, since $L[\mathbf{A}]$ can be computed from $\mathbf{A}$ by the function eval.

If we want to address these notions without mentioning any particular one, we write simply KEP.

□

The theorem above expresses the fact that Kolmogorov expressive power is syntax and implementation independent to the maximal possible extent, achievable for any notion based on the Kolmogorov complexity.

Remark 3.5. Let us note, that, since all the introduced quantities are defined "up to an additive constant", some subtle inexpressibility results cannot be formulated in our framework. E.g., a result stating that there are nonisomorphic graphs $\mathbf{A}, \mathbf{B}$ which are indistinguishable by all sentences of a logic L, says that one bit of information (the difference between $\mathbf{A}$ and $\mathbf{B}$) cannot be retrieved by sentences of the logic. But this bit gets lost in the constant indeterminacy of Kolmogorov complexity. All what we can express in our language is always determined "up to an additive constant", so the extremes of $I_L(\mathbf{A})$ are $C(\mathbf{A}) - O(1)$ (the highest possible expressive power) and $O(1)$ (the lowest possible expressive power).

4 The Query Languages

The definitions form the previous section apply to almost any reasonable query language. The limitation is that all its queries should be total. In this paper we intend to deal with some specific query languages: first order logic FO, inflationary fixpoint logic LFP and partial fixpoint logic PFP. We can again recommend the book [2] as a source for most of what we need to know about them.

We assume that the reader is already familiar with first-order logic FO. LFP and PFP were introduced to remedy an important weakness of FO, namely the lack of any recursion mechanism. E.g., FO fails to express the transitive closure of a graph, or that a graph is connected.

All the notions of interest for us are syntax independent, so the choice of a syntactical representation of a particular semantical query language is immaterial. In this semantical sense two query languages we are going to speak about have been introduced by Chandra and Harel in [8, 9]. But we use a different syntax for them. As a matter of fact, each one of them allows a number of different definitions, though of the same expressive power over finite models. Both of them allow iterating a FO formula up to a fixpoint. The difference is in the form of iteration.

Definition 4.1 (LFP **and** PFP **queries).**

First we define PFP, and then by restricting the syntax we get also LFP.

Let $\varphi(R, \bar{x})$ be a k-ary PFP formula over $\sigma' = \sigma \cup \{R\}$, where R is k-ary and does not occur in σ.

Then the formula $[\mathrm{fp}\ \varphi(R, \bar{x}), R](\bar{x})$ is in PFP, where $\bar{x}$ is a k-tuple of variables; the semantics of the formula is as follows:

Let $\mathbf{A}$ be a finite structure over σ. Then let $\Phi^0 = \emptyset$, and $\Phi^{i+1} = \{\bar{a} \in A^k \ : \ \mathbf{A} \models \varphi(\Phi^i, \bar{a})\}$ for $i = 0, 1, \ldots$

The sequence Φ^i need not be convergent. **If it is,** then the limit is denoted by Φ^∞ (and is equal to $\Phi^{2^{|A|^k}}$); otherwise the default value for Φ^∞ is $\emptyset$. Finally,

$$\mathbf{A} \models [\mathrm{fp}\ \varphi(R, \bar{x}), R](\bar{a}) \quad \text{iff} \quad \bar{a} \in \Phi^\infty.$$

To get LFP we restrict the use of fp constructor: it can be applied only if the relation variable R appears *positively,* i.e., under an even number of negations, in φ. This restriction ensures that the sequence Φ^i is ascending, so it must converge to a limit Φ^∞ (and $\Phi^\infty = \Phi^{|A|^k}$).

To make the distinction between PFP and LFP formulae clear, we will use pfp instead of fp in PFP formulae, and similarly lfp in LFP formulae. $\square$

The sets of queries we have just defined are denoted by PFP and LFP, respectively. FO^k, LFP^k and PFP^k stand for the sets of those queries in FO, LFP and PFP, respectively, in which only k variables are used.

4.1 The Descriptive Expressive Power of PFP and LFP

We say that a query language L *captures* a complexity class C iff every query $\varphi \in L$ is computable in C and vice versa: every query φ which is computable in C is definable in L.

The following theorem summarizes some of the information about the descriptive power of LFP and PFP.

Theorem 4.2. 1. *Over ordered finite structures* LFP *captures* PTIME*, Immerman* [14]*, Vardi* [23].

2. *Over ordered finite structures* PFP *captures* PSPACE*, Abiteboul and Vianu* [4]*, Vardi* [23].

3. *Over arbitrary finite structures neither* LFP *nor* PFP *can express the Boolean query* the cardinality of the structure is even*, Chandra and Harel* [9].

4. *The standard (descriptive) expressive powers of* LFP *and* PFP *are equal if and only if* PTIME=PSPACE*, Abiteboul and Vianu* [5]. □

4.2 The Game and the Normal Form Theorem

First a little bit of notation: we write $\langle \mathbf{A}, \bar{a}\rangle \equiv_k \langle \mathbf{B}, \bar{b}\rangle$ iff $\langle \mathbf{A}, \bar{a}\rangle$ and $\langle \mathbf{B}, \bar{b}\rangle$ cannot be distinguished by any FO^k formula, i.e., when $\mathbf{A} \models \varphi(\bar{a}) \Leftrightarrow \mathbf{B} \models \varphi(\bar{b})$ for all formulae $\varphi \in \mathrm{FO}^k$. If in the above $\mathbf{A} = \mathbf{B}$ and is unimportant to specify, we write $\bar{a} \equiv_k \bar{b}$. Similarly $\mathbf{A} \equiv_k \mathbf{B}$ iff $\mathbf{A}$ and $\mathbf{B}$ cannot be distinguished by any FO^k sentence, i.e., when $\mathrm{FO}^k[\mathbf{A}] = \mathrm{FO}^k[\mathbf{B}]$.

The other fact we will need is a characterization of the expressive power of the logics we have introduced in terms of an infinitary Ehrenfeucht-Fraïssé-style pebble game. This will lead ultimately to the conclusion, that indistinguishabilities of structures in FO^k, LFP^k and PFP^k are actually the same relation.

Definition 4.3 (The Game).

Players, board and pebbles. The game is played by two players, Spoiler and Duplicator, on two σ-structures $\mathbf{A}, \mathbf{A}'$. There are k pairs of pebbles: $(1, 1'), \ldots, (k, k')$. Pebbles with primes are intended to be placed on elements of A', while those without primes on elements of A.

First phase. In each of the first k moves Spoiler is allowed to choose one of the structures and one of elements of that structure and place a pebble on it. Duplicator must place the other pebble from the pair on some element in the other structure, so that the partial function from $\mathbf{A}$ to $\mathbf{A}'$ defined by mapping $x \in \mathbf{A}$ on which there is placed a pebble i, onto the element $x' \in \mathbf{A}'$ on which there is placed pebble i' and constants in $\mathbf{A}$ onto constants in $\mathbf{A}'$ is a partial isomorphism. Spoiler is allowed to alternate between the structures as often as he likes, when choosing elements.

Second phase. In each of the consecutive moves $(k+1, k+2, \ldots)$ Spoiler is allowed to choose one of the structures and one of the pebbles placed on an element of that structure and move it onto some other element of the same structure. Duplicator must move corresponding pebble in the other structure so that the existence of partial isomorphism is preserved.

Who wins? Spoiler wins if Duplicator does not have any move preserving the isomorphism. We say that Duplicator has a winning strategy if he can play forever despite of the moves of Spoiler, preventing him from winning.

This game characterizes the expressive power of the logics we have introduced in the following sense:

Theorem 4.4 (Barwise [6], Immerman [12], Poizat [20]). *Let* $\mathbf{A}, \mathbf{B}$ *be any two structures and let* $\bar{a} \in A^k$, $\bar{b} \in B^k$.

1. *The Duplicator has a winning strategy in the second phase of the game when after the first one the pebbles are placed on* $\bar{a}$ *and* $\bar{b}$ *in* $\mathbf{A}$ *and* $\mathbf{B}$, *respectively, if and only if* $\langle \mathbf{A}, \bar{a} \rangle \equiv_k \langle \mathbf{B}, \bar{b} \rangle$.
2. *Duplicator has a winning strategy in the game on* $\mathbf{A}$ *and* $\mathbf{B}$ *iff* $\mathbf{A} \equiv_k \mathbf{B}$. □

We are going to present a very powerful normal form theorem for PFP queries, proved first by Abiteboul and Vianu [5]. This is the basis for an interesting optimization method, called *split,* and will be shown in this paper to lead to a result concerning limitations of the Kolmogorov expressive power of query languages.

Namely, for every signature σ and every natural k there exists a query ϖ_k in LFP^{2k+2}, with output signature $\pi_k(\sigma)$ to be defined precisely soon, which in arbitrary $\mathbf{A} \in \Sigma$ computes the equivalence relation $\equiv_k$ on k-tuples of elements of A, a pre-ordering $\preceq_k$ of A^k, and a tuple of additional binary and unary relations $\bar{R}$ over A^k. (More precisely, each of the mentioned relations separately is definable in LFP^{2k+2}.)

The intended invariant is then the quotient structure $\pi_k(\mathbf{A}) = \varpi_k(\mathbf{A})/\equiv_k$. We give now a description of $\pi_k(\mathbf{A})$ in the form presented in [10]. We don't give the arguments showing well-definedness of π_k. We assume that $\sigma = \langle R_1, \ldots, R_t \rangle$ and all the relations R_i have arity at most k. For other signatures we can replace each relation symbol R (of arity $m > k$) with a collection of relation symbols of arity k, one for each way an m-tuple can be formed from a sequence of k variables. This encoding suffices, since we are considering only formulae with at most k variables.

Definition 4.5 (Dawar, Lindell, Weinstein [10]).
Let $\mathbf{A} = \langle A, R_1, \ldots, R_t \rangle$ be any structure. Then

$$\pi_k(\mathbf{A}) = \langle A^k/\equiv_k, \preceq_k, =', R'_1, \ldots, R'_t, X_i, P_s \rangle$$

is defined as follows:

1. The universe of $\pi_k(\mathbf{A})$ is $A^k/\equiv_k$, i.e., the equivalence classes of tuples from A^k under the equivalence relation $\equiv_k$. We will write $[\bar{a}]$ to denote the equivalence class to which $\bar{a} = \langle a_1, \ldots, a_k \rangle$ belongs.
2. $\preceq_k$ is a linear ordering of the universe of $\pi_k(\mathbf{A})$.
3. $='$ is a unary relation such that $='([\bar{a}])$ holds iff $\bar{a} = \langle a_1, a_2, \ldots, a_k \rangle$ and $a_1 = a_2$.
4. To each relation R_i of arity $m \leq k$ in $\mathbf{A}$ corresponds an unary relation R'_i in $\pi_k(\mathbf{A})$, where $R'_i([a_1, \ldots, a_k])$ holds iff $R_i(a_1, \ldots, a_m)$.
5. X_i is a binary relation such that $X_i([\bar{a}], [\bar{a}'])$ holds iff $\bar{a}$ and $\bar{a}'$ differ at most on their i-th element.
6. For every sequence $s = i_1, \ldots, i_k$ of integers from $\{1, \ldots, k\}$ P_s is a binary relation, and for any tuple $\langle a_1, \ldots, a_k \rangle$, $P_s([a_1, \ldots, a_k], [a_{i_1}, \ldots, a_{i_k}])$.

It appears that $\pi_k(\mathbf{A})$ contains already all the information about truth of PFP^k formulae in $\mathbf{A}$.

Theorem 4.6 (Abiteboul and Vianu [5]). *1. For every k-ary query φ in* PFP^k *over σ there exists an unary query φ^{π_k} in* PFP *over σ_k satisfying that $\varphi(\mathbf{A})$ is equal to $\bigcup \varphi^{\pi_k}(\pi_k(\mathbf{A}))$. (Recall that $\varphi^{\pi_k}(\pi_k(\mathbf{A}))$ is a set of $\equiv_k$-equivalence classes of k-tuples over A.) φ^{π_k} can be constructed effectively from φ.*

2. Moreover, if φ is in LFP, *then φ^{π_k} is also a query from* LFP. □

Corollary 4.7. $\langle \mathbf{A}, \bar{a}\rangle \equiv_k \langle \mathbf{B}, \bar{b}\rangle$ *iff* $\langle \pi_k(\mathbf{A}), [\bar{a}]\rangle \cong \langle \pi_k(\mathbf{B}), [\bar{b}]\rangle$. □

The query π_k, reducing computations of queries over any unordered structure $\mathbf{A}$ to the computations over the ordered structure $\pi_k(\mathbf{A})$ is a very powerful technique. (Formally only ϖ_k is a query, since π_k does not preserve the universe of its argument. But it is convenient to call it so.) It serves as a basis for very important theoretical results, e.g., the proof of the point 4 of Theorem 4.2 is based on this approach. It is also useful for results much closer to the practice, concerning query optimization.

4.3 Split

The *split* optimization technique is based on the fact, that often the value $|\pi_k(\mathbf{A})|$ is much smaller than $|A|$. Then the computation of φ represented as superposition of π_k and φ^{π_k} can be much more efficient than the straightforward computation of φ, suggested by semantics. We refer the reader to papers [3, 1], where this method is presented in more detail. It should be noted that *split* presented in [3] is generally more complicated than the method of [1], since it is more practice-oriented, and allows avoiding the overhead of computing $\pi_k(\mathbf{A})$ when it is of cardinality comparable to $|A|$. Nevertheless, we use the name *split* for both, having in mind all the methods based on the low cardinality of $\pi_k(\mathbf{A})$.

Concerning split, we make here an assumption: *φ^{π_k} is evaluated according to the naive semantics.* In the next sections we are going to show some limitations of this method, and we must prevent ourselves from other, unintended optimizations, except of the plain split. But we allow clever ways of computing π_k as a source of speed-up in split.

5 KEP of First-Order Logic and its Extensions

5.1 The Choice of Query Languages

The main difference between Kolmogorov expressive power and descriptive expressive power is that in the descriptive case queries are tested one by one against all finite models, while in the Kolmogorov case models are tested one by one against all the Boolean queries. Precisely speaking, the descriptive complexity is a *data complexity* in the sense of Vardi [23], while our KEP is a (kind of) *expression complexity.* So the results are in a sense orthogonal. In this section we will show some of the differences and similarities between the descriptive complexity and Kolmogorov expressive power of fixpoint extensions of first-order logic. They will serve us as tools in proving more advanced results in the next section. We begin with a discussion of the right choice of the query languages to deal with.

The query language Σ^0_1 is the existential fragment of first order logic. It appears already to be too strong for our purposes, as the next proposition shows.

Proposition 5.1. *For arbitrary finite structure* $\mathbf{A}$ *the Kolmogorov expressive power of Σ^0_1 is already maximal, i.e.,*

$$I_{\Sigma^0_1}(\mathbf{A}) = C(\mathbf{A}) - O(1).$$

Proof. It is well known, that for every finite $\mathbf{A}$ the query 'the structure contains a substructure isomorphic to $\mathbf{A}$' is definable in Σ^0_1. Now we need only constant number of extra bits to write a program that will reconstruct the isomorphism type of a structure from its Σ^0_1 theory as follows: the program looks for the first true sentence 'the structure contains a substructure isomorphic to $\mathbf{A}$' such that all the sentences 'the structure contains a substructure isomorphic to $\mathbf{B}$' for $\mathbf{B}$ satisfying $|B| = |A| + 1$ are false. Then certainly the structure at hand must be isomorphic to $\mathbf{A}$, so what remains is to output $\mathrm{enc}(\mathbf{A})$. □

We have just seen, that even the very simple fragment Σ^0_1 of the full first-order logic appears to be already too strong in our context. Therefore we should restrict our attention somehow. It is quite natural in this context to restrict the number of variables in formulae. This is our choice in this paper.

The first result indicates the difference between Kolmogorov complexity and descriptive complexity, showing that the Kolmogorov expressive powers of PFP^k and LFP^k are the same as that of FO^k. This is untrue in the descriptive sense, since PFP and LFP are known to be stronger than FO.

Theorem 5.2. *For arbitrary structure $\mathbf{A} \in \Sigma$ the quantities: $I_{\mathrm{FO}^k}(\mathbf{A})$, $I_{\mathrm{LFP}^k}(\mathbf{A})$ and $I_{\mathrm{PFP}^k}(\mathbf{A})$ are equal up to an additive constant, independent of $\mathbf{A}$.*

Proof. Since $\mathrm{FO}^k \subseteq \mathrm{LFP}^k \subseteq \mathrm{PFP}^k$, it is enough to show, that $C(\mathbf{A} \,|\, \mathrm{PFP}^k[\mathbf{A}]) \leq C(\mathbf{A} \,|\, \mathrm{FO}^k[\mathbf{A}]) + O(1)$. We will show even more, namely that $C(\mathrm{PFP}^k[\mathbf{A}] \,|\, \mathrm{FO}^k[\mathbf{A}]) = O(1)$. This means that not only the three logics express the same quantity of information about structures: they express *the same information.*

Suppose that $\phi(\mathtt{prog}, \mathrm{PFP}^k[\mathbf{A}]) = \mathrm{enc}(\mathbf{A})$ for some program $\mathtt{prog} \in \{0,1\}^*$. According to our convention, `prog` is a Pascal program and $\mathrm{PFP}^k[\mathbf{A}]$ is stored in a (infinitely long) file. In the process of computation `prog` reads several times the values $\varphi[\mathbf{A}]$ for $\varphi \in \mathrm{PFP}^k$. It is enough to show, that each such access can be substituted by a call of a finite-length subroutine `subrt`, which computes the correct value $\varphi[\mathbf{A}]$, accessing only $\mathrm{FO}^k[\mathbf{A}]$. This will show, that $\mathbf{A}$ can be also computed from $\mathrm{FO}^k[\mathbf{A}]$ by a program whose length is only a constant greater than that of `prog`.

We describe the computation of `subrt` by means of two examples.

Assume first that `subrt` gets a sentence $\varphi([\mathrm{pfp}\ \psi(R,\bar{x}), R](\bar{x}))$ as input, where ψ is first order and second order variables which are bound by other fixpoint operators in φ do not occur in ψ. (The main role of φ is to bind the free variables of the fixpoint.)

Observe, that for $[\mathrm{pfp}\ \psi(R,\bar{x}), R](\bar{x})$ its stages $\Psi^i(\bar{x})$ are FO^k definable. Then `subrt` looks through $\mathrm{FO}^k[\mathbf{A}]$ for the first (in the listing) true sentence of the form:

$$\forall \bar{x} \left(\begin{array}{c} \Psi^i(\bar{x}) \leftrightarrow \Psi^{i+1}(\bar{x}) \\ \vee \\ \neg(\Psi^i(\bar{x}) \leftrightarrow \Psi^{i+1}(\bar{x})) \wedge \bigvee_{j \leq i} \Psi^i(\bar{x}) \leftrightarrow \Psi^{i+j}(\bar{x}) \end{array} \right). \tag{5.1}$$

for some $i \in \mathbf{N}$. There is such a sentence, namely with $i = 2^{|A|^k}$, since among that many consecutive stages of a PFP induction there must be two identical ones. If they are two consecutive ones, then the fixpoint is reached (the first condition above), otherwise the induction loops without reaching a fixpoint (the second condition above). What we learn by finding formula (5.1) is the value of i. After such i is found, all occurrences of $[\mathrm{pfp}\ \psi(R,\bar{x}), R](\bar{x})$ in φ can be replaced by the following first order formula,

$$(\Psi^i(\bar{x}) \leftrightarrow \Psi^{i+1}(\bar{x})) \wedge \Psi^i(\bar{x}), \tag{5.2}$$

which is equivalent in $\mathbf{A}$ to $[\mathrm{pfp}\ \psi(R,\bar{x}), R](\bar{x})$.

The general case, where the fixpoint subformulae overlap is a bit more complicated. For simplicity we describe the computation of `subrt` with $\forall \bar{x}[\mathrm{pfp}\ \xi([\mathrm{pfp}\ \psi(R,S,\bar{x}), R](\bar{x})), S](\bar{x})$, where ξ is first order. The problem is that the inner fixpoint doesn't compute in $\mathbf{A}$, but in its expansion by the current (i.e., depending on the current stage of the outer fixpoint) value of S, so we cannot check the number of steps it needs to fully evaluate "on side", as we have done before. We have to expand both fixpoints simultaneously, instead.

First we substitute S in ψ by a formula constantly false and compute the first order equivalent φ_0 of the inner fixpoint, using (5.1) and (5.2). Then the first order formula $\xi(\varphi_0)$ defines the first stage of the outer fixpoint. Now we substitute it in ψ for S and find again the first order definition φ_1 of the inner fixpoint, and then $\xi(\varphi_1)$ is the first order definition of the second stage of the outer fixpoint, and so on. This gives us a sequence of first order definitions of consecutive stages of the outer fixpoint, so we can use formulae (5.1) and (5.2) to find the first order definition of it.

In this way `subrt` having access to $\mathrm{FO}^k[\mathbf{A}]$ can translate any PFP^k sentence into its FO^k equivalent in $\mathbf{A}$, and thus can answer all questions of the main program concerning bits of $\mathrm{PFP}^k[\mathbf{A}]$. (Note that none of the manipulations we have done required adding variables.) □

The above theorem can be seen as an effective version of a theorem of Kolaitis and Vardi [17], stating that if two structures satisfy the same sentences of FO^k, then they satisfy the same sentences of the infinitary logic $L^k_{\infty\omega}$, which subsumes both FO^k and PFP^k. This version of their theorem has been certainly observed by other researchers too.

5.2 The Normal Form as a Data Compression

The Normal Form Theorem can be understood as a form of computing with compressed database. Namely $\pi_k(\mathbf{A})$ is the compressed version of $\mathbf{A}$. It is sufficient in order to evaluate all PFP^k sentences "as in $\mathbf{A}$". Precisely speaking, the following holds:

Proposition 5.3. *There is a partial recursive function* $e_k : \mathrm{PFP}^k \times \pi_k(\Sigma) \to \{0,1\}$ *such that*

$$e_k(\varphi, \pi_k(\mathbf{A})) = \varphi[\mathbf{A}]$$

for all $\mathbf{A} \in \Sigma$ *and all* $\varphi \in \mathrm{PFP}^k$.

Proof. The procedure to compute e_k is as follows: given input structure $\mathbf{C}$ and sentence φ, the algorithm considers consecutively all finite structures $\mathbf{B} \in \Sigma$ until it finds a one such that $\pi_k(\mathbf{B}) \cong \mathbf{C}$. Then it evaluates and outputs $\varphi[\mathbf{B}]$. Observe that if $\mathbf{C} = \pi_k(\mathbf{A})$, then the search will stop, since $\mathbf{A}$ itself is a good choice for $\mathbf{B}$. The correctness of the result follows from the Normal Form Theorem. □

In the proof we could equally well translate φ to φ^{π_k} and do the evaluation in $\pi_k(\mathbf{A})$. But we haven't defined the translation, and from our point of view the brute force search suffices.

Let us now compare this proposition with the following observation, which explains what the normal form has to do with Kolmogorov complexity.

Proposition 5.4. *Let* $\mathtt{prog}_{k,\mathbf{A}}$ *denote a shortest Pascal program which computes* $\mathrm{PFP}^k[\mathbf{A}]$, *i.e., a one with* $lh(\mathtt{prog}) = C(\mathrm{PFP}^k[\mathbf{A}])$. *Then there is a partial recursive function* $\widehat{e}_k : \mathrm{PFP}^k \times \{0,1\}^* \to \{0,1\}$ *such that*

$$\widehat{e}_k(\varphi, \mathtt{prog}_{k,\mathbf{A}}) = \varphi[\mathbf{A}]$$

for all $\mathbf{A} \in \Sigma$ *and all* $\varphi \in \mathrm{PFP}^k$.

Proof. The function we need is essentially a composition of eval and ϕ, the reference universal function we have used to define Kolmogorov complexity. Namely we can take

$$\widehat{e}_k(x, y) = \mathrm{eval}(x, \phi(y, \Lambda)).$$

$\square$

Now it is an immediate observation that $\pi_k(\mathbf{A})$ is an "approximation" of $\mathtt{prog}_{k,\mathbf{A}}$, in the sense that $lh(\mathrm{enc}(\pi_k(\mathbf{A}))$ is an upper approximation of $C(\mathrm{PFP}^k[\mathbf{A}])$.

5.3 KEP vs. Definability of Order

The very important rôle of the order for the descriptive complexity of fixpoint (and not only) languages is well-known. It seems natural to ask whether definability of order has anything to do with KEP. And it appears that it does. So this is an example of a similarity between Kolmogorov and the descriptive versions of expressive power. In the descriptive case over classes of ordered structures a query language usually achieves the maximum of expressive power (e.g., LFP the power of PTIME, and PFP of PSPACE). The same is true about Kolmogorov expressive power. This will be shown in the following.

Proposition 5.5. *Suppose that a query $\varphi(x, y)$ defines a linear ordering of a finite structure* $\mathbf{A}$ *with maximal arity of relations k. Then* FO *with $k+1$ variables and with φ added as a language primitive has the maximal Kolmogorov expressive power:*

$$I_{(\mathrm{FO}+\varphi)^{k+1}}(\mathbf{A}) \geq C(\mathbf{A}) - O(1).$$

Proof. The proof is based on a well-known fact, that in a finite ordered structure $\mathbf{A}$ every element a has a unique definition by a formula $\varphi_a(x)$ which contains only two variables. Therefore in order to uniquely determine a k-tuple, we need $k+1$ variables, since the second variable needed in the definitions of the coordinates can be reused. In this way we can write sentences determining truth of all atoms in all relations in $\mathbf{A}$. After this has been achieved, determining and outputting the isomorphism type of $\mathbf{A}$ is a trivial task. $\square$

It would be nice to have a kind of converse of the above, i.e., to know that high Kolmogorov expressive power of FO^k must be caused be definability of a long linear order on a subset of the universe. This is however difficult to expect literally, since in structures with many automorphisms it can be difficult to define any nonconstant-length order on tuples of elements, as the following example shows:

Example 5.6. Let σ consist of two binary relations R, E. Start with an n-element structure being a linear order with respect to R and in which E is the identity equivalence relation. Now replace each vertex in it by an R-clique of n elements, changing the relations as follows: introduced cliques become equivalence classes of E, and if we replace by cliques two vertices which joined by an R-edge, then we set R-edges between all elements of the two introduced cliques. Quite formally, we mean structures of the form $\langle\{0, \ldots, n-1\} \times \{0, \ldots, n-1\}, R, E\rangle$, where $R(\langle x, y\rangle, \langle x', y'\rangle)$ iff $x \leq x'$, and $E(\langle x, y\rangle, \langle x', y'\rangle)$ iff $y = y'$.

It is an easy observation, that for each such constructed $\mathbf{A}$ holds $I_{\mathrm{FO}^3}(\mathbf{A}) = C(\mathbf{A}) + O(1)$, although no formula of PFP can define a nonconstant length *ordering* on a subset of A, even with parameters. The reason is that a formula of PFP must have finitely many variables, say k. Then it is able to keep fixed at most k parameters, and if $n > k+1$ then in the n^2-element structure we have constructed every vertex can be moved by an automorphism which is identity on the fixed parameters. This excludes the possibility of defining any ordering including tuples other than ones consisting entirely of parameters. $\square$

It seems more reasonable to look for a pre-order on tuples of elements, instead. We will indeed prove some kind of such a converse for query languages that extend the fixpoint query language. Accidentally, the same result indicates also connections of Kolmogorov expressive power with the *split* technique.

We begin with a technical result, which will be helpful in the considerations.

Lemma 5.7. *Let L be any Boolean query language consisting of total queries. Then*

$$I_L(\mathbf{A}) \leq C(L[\mathbf{A}]) + O(\log C(L[\mathbf{A}])), \tag{5.3}$$

for all finite structures $\mathbf{A} \in \Sigma$.

Proof. We have to show that

$$C(\mathbf{A}) - C(\mathbf{A} \mid L[\mathbf{A}]) \leq C(L[\mathbf{A}]) + O(\log C(L[\mathbf{A}])),$$

or equivalently that

$$C(\mathbf{A}) \leq C(\mathbf{A} \mid L[\mathbf{A}]) + C(L[\mathbf{A}]) + O(\log C(L[\mathbf{A}])).$$

And indeed: in order to reconstruct $\mathbf{A}$ from scratch we need two programs for ϕ : first of length $C(\mathbf{A} \mid L[\mathbf{A}])$ which reconstructs $\mathbf{A}$ from $L[\mathbf{A}]$ and second of length $C(L[\mathbf{A}])$ which computes bits of $L[\mathbf{A}]$ requested by the first program. We also need $O(\log C(L[\mathbf{A}]))$ bits to separate these two programs and finally a finite amount of bits to describe the just presented reconstruction method. □

In contrary to the claim the author made in the conference version [22] of this paper, generally it is not the case that the converse inequality to the above (5.3) holds, i.e., in general it is not true that

$$I_L(\mathbf{A}) = C(L[\mathbf{A}]) + O(\log C(L[\mathbf{A}])).$$

We can already announce the following proposition, which however won't be used any more in this paper and whose proof will be therefore presented in the Appendix.

Proposition 5.8. *There is a constant k_0 and an infinitely many models $\mathbf{A}$ of signature consisting of one binary symbol, and such that for every $k \geq k_0$ and every sufficiently large $\mathbf{A} \in D$ the following simultaneously hold:*

$$C(\mathrm{FO}^k[\mathbf{A}]) = \Omega(\log C(\mathbf{A})),$$
$$I_{\mathrm{FO}^k}(\mathbf{A}) = O(1).$$

□

This means that $\mathrm{FO}^k[\mathbf{A}]$ can contain at least about $c \cdot \log C(\mathbf{A})$ bits of "useless" information, for some constant $c > 0$. The author does not know, if this is already the maximal possible difference. It would be easier to refute the opposite direction of (5.3) using an artificial query language, but it is instructive to know, that it is not true even for a natural and generally well-behaved one, like FO^k.

Before coming to the estimation of $I_{\mathrm{FO}^k}(\mathbf{A})$, we remove redundant fragments from $\pi_k(\mathbf{A})$.

Lemma 5.9. *For every finite structure* $\mathbf{A}$ *over* σ *and every fixed* k*, if* $|\pi_k(\mathbf{A})| = n$*, then* $C(\pi_k(\mathbf{A})) \leq 4n \log n + cn$*, where* c *is a constant depending on* σ *and* k*.*

Proof. Let us verify, that $\pi_k(\mathbf{A})$, as described in Definition 4.5, can be encoded in $4n \log n + cn$ bits, where $n = |\pi_k(\mathbf{A})|$.

First of all we can up to isomorphism assume the universe of $\pi_k(\mathbf{A})$ to be $\{0, \ldots, n-1\}$ and $\preceq_k$ to be the standard ordering.

Each of the unary relations mentioned in points 3 and 4 can be encoded as a binary string of $n + O(1)$ bits.

First note that the relations P_s mentioned in point 6 are functions on the universe of $\pi_k(\mathbf{A})$, and since there are n^n such functions, each one of them can be described by a binary string of length $n \log n + O(1)$. To determine all of them, we need only three: $P_{2,1,3,\ldots,k}$, $P_{2,3,\ldots,k,1}$ and $P_{2,2,3,4,\ldots,k}$. The first two correspond to s being a transposition and a full cycle, hence their compositions suffice to define all P_s with s being a permutation of $1, \ldots, k$. The third then gives a method to replace the first element in the sequence by a duplicate of the second. Composing this operation with permutations we can define all remaining relations P_s.

Finally we need only X_1 from all relations X_i. The reason is that to define X_i we can to permute the tuples by means of a suitable P_s to get i-th components at the first place and then apply X_1. The other observation is that X_1 is obviously an equivalence relation. The number of equivalence relations on n elements is equal to so called *Bell number* B_n, whose behaviour is rather complicated. But it is fairly easy to see that $B_n \leq n^n$. Namely every equivalence relation E on $\{0, \ldots, n-1\}$ can be represented by a function $f : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ such that $\ker f = E$, and there are only n^n such functions. Hence we can represent X_1 by a binary string of $n \log n + O(1)$ bits.

The whole $\pi_k(\mathbf{A})$ can be then represented by a tuple made of all the above strings. The total length of all strings is $4n \log n + cn + O(1)$, and the separation markers introduced by using the function *pair* require $O(\log n)$ bits, and hence their length disappears in the linear term of the complexity. □

Now the promised kind of converse of Proposition 5.5, being also an estimate of $I_{\mathrm{FO}^k}(\mathbf{A})$ by complexity of $\pi_k(\mathbf{A})$.

Theorem 5.10. *Suppose that* $\mathbf{A}$ *is finite and* $|\pi_k(\mathbf{A})| = n$*. Then*

$$I_{\mathrm{FO}^k}(\mathbf{A}) \leq 4n \log n + cn,$$

where the constant c *depends only on* k *and the signature of* $\mathbf{A}$*.*

Proof. Let $\mathbf{A} \in \Sigma$ and let $|\pi_k(\mathbf{A})| = n$. Let us see, that the result of evaluation of each FO^k query φ in $\mathbf{A}$ can be decided from a description of $4n \log n + cn$ bits. Indeed, the structure $\pi_k(\mathbf{A})$ can be described uniquely by at most that many bits, by Lemma 5.9. In addition we need a constant size algorithm computing the function e_k from Proposition 5.3.

The whole process of reconstruction of the truth value of φ requires thus $4n \log n + cn$ bits describing $\pi_k(\mathbf{A})$, plus a finite number of bits. As the reconstruction method is independent of φ, it yields an algorithm for computing consecutive bits of $\mathrm{FO}^k[\mathbf{A}]$ as a function. Therefore $C(\mathrm{FO}^k[\mathbf{A}]) \leq 4n \log n + cn$. By Lemma 5.7 we get the theorem. (Note that Lemma 5.7 possibly adds a logarithmic term to the result. It disappears in the linear term however.) □

The above theorem is a converse of Proposition 5.5 in this sense, that it estimates the minimal length of a pre-order definable in $\mathbf{A}$ by LFP^{2k+2} in terms of $I_{\mathrm{FO}^k}(\mathbf{A})$. This estimate is almost linear, as the following immediate corollary shows.

Corollary 5.11. *For every k and sufficiently large $\mathbf{A}$ holds*

$$|\pi_k(\mathbf{A})| > \frac{I_{\mathrm{FO}^k}(\mathbf{A})}{5\log(I_{\mathrm{FO}^k}(\mathbf{A}))}.$$

□

The corollary gives only a *theoretical* lower bound on the length of a definable pre-ordering. The real value of it can be much (even exponentially) larger. The main reason is that seldom the real information content of $\pi_k(\mathbf{A})$ is maximal possible, and also seldom all the information present in $\pi_k(\mathbf{A})$ can be really retrieved from $\mathbf{A}$ by queries with only k variables. Below we show the best upper bound known to the author.

Proposition 5.12. *For every $\varepsilon > 0$ there exist: a natural k, a signature σ and an infinite set D of structures over σ, such that for all $\mathbf{A} \in D$*

$$|\pi_k(\mathbf{A})| \leq (I_{\mathrm{FO}^k}(\mathbf{A}))^{1+\varepsilon}.$$

Proof. Let k be chosen such that $(1+\varepsilon)(k-2) > k$. Let σ consist of an ordering symbol $\preceq$ and one $k-1$-ary relation symbol R. Let D consist of all structures $\mathbf{A}$ in which $\preceq$ is interpreted as the true inequality on the universe of $\mathbf{A}$. For every n there are $2^{n^{k-1}}$ such nonisomorphic models. So by Proposition 2.5 for every n there is an n-element $\mathbf{A}$ in D such that $C(\mathbf{A}) \geq n^{k-1}$. In view of Theorem 5.5 in D holds $I_{\mathrm{FO}^k}(\mathbf{A}) = C(\mathbf{A}) - O(1)$. Take all finite $\mathbf{A}$ with $C(\mathbf{A}) \geq |A|^{k-1}$. There are infinitely many such $\mathbf{A}$. Therefore for each such $\mathbf{A}$ we get

$$(I_{\mathrm{FO}^k}(\mathbf{A}))^{1+\varepsilon} \geq (n^{k-1} - O(1))^{1+\varepsilon} \geq (n^{k-2})^{1+\varepsilon} > n^k = |\pi_k(\mathbf{A})|.$$

□

6 KEP and Optimization of Queries

We have argued that the value $I_{\mathrm{FO}^k}(\mathbf{A})$ is interesting for at least two reasons: first for its own sake, since it expresses how much of the information content of the database $\mathbf{A}$ can be really seen by a user querying with Boolean queries from FO^k; second since it is related to the questions concerning optimization of query evaluation. This second reason hasn't been so far discussed. Now it is time for it.

First of all, we show that a large value of $I_{\mathrm{FO}^k}(\mathbf{A})$ is a rather bad news for split optimizations, while it will appear from the example we are going to discuss, that a low value of $I_{\mathrm{FO}^k}(\mathbf{A})$ indicates the potential possibility of using optimization techniques "beyond split". We defer the precise formulation of this point until later.

In this section we define a simple family of classes of databases. For each class in the family we investigate both questions: how expressive is FO^k in the Kolmogorov sense in each class, and what we can do concerning optimization of LFP and PFP queries.

The results demonstrate that we can prove some nontrivial results concerning the Kolmogorov complexity in a slightly more practical setting. Since the Kolmogorov complexity itself is noncomputable, we prove results concerning its value on almost all structures, and its connections with average time and space complexity of query evaluation.

Definition 6.1. Let $\Sigma(n) \subseteq \Sigma$ stand for the set of all isomorphism types of n-element finite structures over σ. For counting purposes we will consider structures themselves rather than their isomorphism types. Therefore let $\overline{\Sigma}$ stand for the class of all finite structures over σ, whose universe is some finite initial segment of natural numbers.

We allow the convention, that for $\overline{E} \subseteq \overline{\Sigma}$ the set E is the set of all isomorphism types of structures in $\overline{E}$, and *vice versa:* for $E \subseteq \Sigma$ $\overline{E}$ is the set of all structures in $\overline{\Sigma}$ isomorphic to some structure in E. In particular, writing $\overline{E}$ we always assume this set to be closed under isomorphisms within $\overline{\Sigma}$.

We fix a surjective encoding $\overline{\mathrm{enc}} : \overline{\Sigma} \to \{0,1\}^*$ of structures in $\overline{\Sigma}$ as binary words, defined for graphs by choosing $\overline{\mathrm{enc}}(\mathbf{A})$ to be the concatenation of the rows of the adjacency matrix of $\mathbf{A}$, and similarly for general structures. For some technical reasons it is important, that encodings of structures of the same cardinality are of equal length. This fixed encoding is used to speak about complexity of evaluation of queries.

It can be assumed, and we do it, that $\Sigma \subseteq \overline{\Sigma}$.

Now let σ be arbitrary non-unary signature and let U be a new unary relation symbol $U \notin \sigma$, and let $\sigma' = \sigma \cup \{U\}$. The set of the isomorphism types of finite σ'-structures is then Σ', and in similar way we define $\overline{\Sigma}'$. We will use set-notation for U, i.e., we will write $x \in U$ instead of $U(x)$ in the formulae. The intended meaning is that elements $a \in U$ of the finite structure $\mathbf{A} \in \Sigma'$ are important. We limit the choice of U as follows: let i be any function $\mathbf{N} \to \mathbf{N}$ satisfying $i(n) \leq n$, and let $\overline{\Sigma}^i(n) \subseteq \overline{\Sigma}'(n)$ consist of all structures $\mathbf{A}$ satisfying $x \in U \leftrightarrow x \leq i(|A|)$. Thus the initial $i(n)$ elements of A are those in U. We will assume $i(n) \leq n/2$. For KEP it is w.l.o.g., due to symmetry: each structure is considered separately, and U and its complement play indeed a symmetrical rôle in it. If we want to address a substructure of $\mathbf{A}$ induced by elements of some $S \subseteq A$, we denote it $\mathbf{A}_{\restriction S}$. In case $S = U$ and there is no risk of ambiguity, we write $\mathbf{U}$ instead of $\mathbf{A}_{\restriction U}$.

Another way of expressing the same would be to define Σ^i as a class of two-sorted structures, with first sort of size $i(n)$ corresponding to U and second sort of size $n - i(n)$ corresponding to $A \setminus U$. In this version each relation from our model is replaced by a tuple of relations, one for each choice of the arguments to be in the first or second sort.

For simplicity of notation we assume $\overline{\Sigma}^0 = \overline{\Sigma}$. (Here 0 means the function equal constantly 0.) For a subset $D \subseteq \Sigma'$ the set $\overline{D} \subseteq \overline{\Sigma}'$ consists of all structures which are isomorphic to a one in D.

We are going on a tour, to see what happens to the Kolmogorov expressive power and complexity of query evaluation in the classes $\overline{\Sigma}^i$, depending on i. We do not intend to cover all possible functions i, but the picture already shows enough to get an impression about real relations between the two notions we are going to investigate. Since Kolmogorov expressive power is a noncomputable value, instead of the exact values we consider *the most probable* value of it. To keep things comparable, the complexity of query evaluation is also treated in probabilistic sense—namely we investigate the average complexity of queries.

6.1 Tools

In the first part we have to collect all the tools we need to be able to give proofs in the next section.

6.1.1 Probabilistic tools

First we define the notion of probability on a class $D \subseteq \overline{\Sigma}$ of finite structures. Namely we define it as a limit of probabilities on fragments $D(n)$ of D consisting of n element structures.

Definition 6.2. For $E \subseteq D$ we set

$$\mu_n^D(E) = \frac{|E \cap D(n)|}{|D(n)|}.$$

The limit probability μ is then defined by the formula

$$\mu^D(E) = \lim_{n\to\infty} \mu_n^D(E),$$

if the limit exists. We say often that E *is dense in* $\overline{D}$ as a shorthand for $\mu^D(E) = 1$. Another shorthand for this is to say, that *almost every* $\mathbf{A} \in D$ *belongs to* E *(has property* E*, etc.).* Yet another way to express it is to say that a property (of being in E) *holds almost surely in* D.

Definition 6.3. For a finite signature σ the *capacity of* σ is the polynomial

$$\mathrm{cp}_\sigma(x) = \sum_{R\in\sigma} x^{\mathrm{arity}(R)}.$$

□

The sense of capacity is that the value $\mathrm{cp}_\sigma(n)$ gives the theoretical upper bound on the Kolmogorov complexity of n-element *ordered* structure over σ.

Definition 6.4. For a finite signature σ the *capacity of* σ is the polynomial

$$\mathrm{cp}_\sigma(x) = \sum_{R\in\sigma} x^{\mathrm{arity}(R)}.$$

□

The sense of capacity is that the value $\mathrm{cp}_\sigma(n)$ is the number of atoms to be decided to be false or true to get a complete n element structure of signature σ. It gives the theoretical upper bound on the Kolmogorov complexity of n-element *ordered* structure over σ, as well.

The first result we will need is known from the work of Fagin [11] (in the case of graphs it has been shown already by Pólya [21]). All the items are tightly related to each other by Burnside's Lemma. Let us recall the standard definition, that a structure is *rigid* if it has no nontrivial automorphism.

Theorem 6.5. *1. The set $R \subseteq \Sigma'$ of isomorphism types of rigid structures is dense in Σ'.*

2. Similarly the set $\overline{R}$ of rigid structures is dense in $\overline{\Sigma}'$.

3. $|\overline{\Sigma}^i(n)| = (1 + o(1))\binom{n}{i} \cdot 2^{\mathrm{cp}_\sigma(n)}/n!.$

4. For every $D \subseteq \Sigma'$

$$\lim_{n\to\infty} \mu_n^{\overline{\Sigma}'}(\overline{D}) - \mu_n^{\Sigma'}(D) = 0. \qquad (6.1)$$

□

Remark 6.6. Particularly the equation (6.1) is important: it allows us to move between Σ' and $\overline{\Sigma}'$ when establishing results, whatever will be more convenient. It is also useful to present a different definition of the measure $\mu_n^{\overline{\Sigma}'}$. Assume for simplicity that σ consists of a single k-ary relation symbol R. Then a random structure $\mathbf{A} \in \overline{\Sigma}'$ is constructed in the following experiment: one tosses a fair coin for every tuple $\bar{a} \in A^k$. If the result is heads, then $R(\bar{a})$ holds in $\mathbf{A}$, otherwise it doesn't. After $|A|^k$ tosses the structure is defined. If there are more relations in the signature, one has to choose them at random in the same way, as well. Note that U is pre-defined and we do not toss the coin to determine it.

Definition 6.7. Let us call an k, U*-extension type* any subset of

$$At_{k,U} = \{R(x_i, x_k), R(x_k, x_i), R(u, x_k), R(x_k, u), R(x_k, x_k) \ : \ i = 1, \ldots, k-1, \ u \in U\}.$$

For an k, U-extension type t the let the following sentence be denoted $Ext_{k,U}(t)$:

$$\forall x_1 \ldots x_{k-1} \begin{pmatrix} \bigwedge_{i \neq j} x_i \neq x_k \\ \bigwedge_i x_i \notin U \end{pmatrix} \to \exists x_k \begin{pmatrix} x_k \neq x_1, \ldots x_{k-1} \\ x_k \notin U \\ (\bigwedge t) \\ \neg(\bigvee At_{k,U} \setminus t) \end{pmatrix}.$$

All such sentences are called k, U*-extension axioms.*

Lemma 6.8 (Fagin [11]). *Let $i(n)$ be the constant function 0. Then for every $k, \emptyset$-extension type t*

$$\mu_n^{\overline{\Sigma}}(\{\mathbf{A} \in \overline{\Sigma} \ : \ \mathbf{A} \models Ext_{k,\emptyset}(t)\}) \geq 1 - c^n$$

for some $0 < c < 1$. □

We will need an extension of this lemma for other not too fast growing functions i.

Lemma 6.9. *1. Let $\limsup_{n\to\infty} i(n)/\log n < 1$. Then*

$$\mu_n^{\overline{\Sigma}^i}(\{\mathbf{A} \in \overline{\Sigma}^i \ : \ \mathbf{A} \models \bigwedge_{t \subseteq At_{k,U}} Ext_{k,U}(t)\}) \geq 1 - o(n^{-m})$$

for every $m \in \mathbf{N}$.

2. Let $i(n)/\log n \to \infty$. Let $Ext_{k,U,\emptyset}$ be the conjunction of all $k, \emptyset$-extension axioms over the signature σ'. (I.e., we start from σ', add to it a new unary symbol getting σ'', and finally set it empty.) Then

$$\mu_n^{\overline{\Sigma}^i}(\{\mathbf{A} \in \overline{\Sigma}^{i,0} \ : \ \mathbf{A} \models Ext_{k,U,\emptyset}\}) \geq 1 - o(n^{-m})$$

for every $m \in \mathbf{N}$.

Proof. We prove the lemma for directed graphs, the argumentation in the general case being identical, only the coefficients become more complicated.

1. Let $c < 1$ and $i(n) < c \log n$ for some n onwards. There are $2^{2k+2i(n)-1}$ k, U-extension axioms. Therefore the fraction of structures in $\overline{\Sigma}^i(n)$ in which at least one of them is false is

$$\begin{aligned}
\mu_n^{\overline{\Sigma}}(\text{some } k, U\text{-ext. axiom fails}) &\leq \underbrace{2^{2k+2i(n)-1}}_{\substack{\text{choices of} \\ \text{ext. axiom} \\ \text{to fail}}} \underbrace{\binom{n}{k-1}}_{\substack{\text{choices of} \\ \text{nonextendable} \\ k-1\text{-tuple}}} \cdot \underbrace{(1 - 2^{-2i(n)+2k+1})}_{\substack{\text{probability that} \\ \text{a single element} \\ \text{is not an} \\ \text{extension}}} {}^{\underbrace{n-i(n)-k+1}_{\substack{\text{to the power} \\ \text{number of} \\ \text{elements}}}} \\
&\leq O(n^{k-1} + c) \left(1 - \frac{1}{2^{c \log n}}\right)^{n - c \log n} \\
&\leq O(n^k) \Big(\underbrace{\left(1 - \frac{1}{n^c}\right)^{n^c}}_{\sim e^{-1}} \Big)^{(n - c\log n)/n^c} \\
&\leq O(n^k) d^{(n - c \log n)/n^c},
\end{aligned}$$

for some $d < 1$, and the latter number is easily seen to be $o(n^{-m})$ for every $m \in \mathbf{N}$. (We won't ever more describe precisely similar computations.)

2. This time the number of extension axioms under consideration is finite. Therefore the fraction of structures in $\overline{\Sigma}^i(n)$ in which at least one of them is false does not exceed

$$c \cdot \binom{n}{k-1}(1-2^{-2k+1})^{i-k+1} = O(n^{k-1})\,(1-d)^{(m+k)\log n},$$

for every $m \in \mathbf{N}$. The reason is that the most difficult situation is when x_k is requested to come from U, so we have only i such elements to choose from. Anyhow, the number is easily seen to be $o(n^{-m})$, which concludes the argument. □

Theorem 6.10 (Kaufmann [15]). *There exists a monadic second order formula of the form $\exists X_1 \ldots \exists X_m \varphi(x, y)$, where $X_1, \ldots, X_m$ are all set variables of φ and x, y are free in φ, such that with $\mu^{\overline{\Sigma}}$-probability* 1

$$\mathbf{A} \models \exists X_1 \ldots \exists X_m \begin{pmatrix} \forall x \ \varphi(x,x) \\ \forall xy \ \varphi(x,y) \wedge \varphi(y,x) \rightarrow x = y \\ \forall xyz \ \varphi(x,y) \wedge \varphi(y,z) \rightarrow \varphi(x,z) \end{pmatrix}.$$

In other words, in almost every $\mathbf{A} \in \overline{\Sigma}$ there are choices of subsets of A witnessing existential set quantifiers such that $\varphi(x, y)$ defines a linear ordering of A. □

The original Kaufmann's proof of this theorem can be found in Appendices.

6.1.2 Logical tools

Lemma 6.11. *Suppose that $\mathbf{A}, \mathbf{B} \in \overline{\Sigma}'$ satisfy all k, U-extension axioms. Then*

1. *For $\bar{a} \in A^k$ and $\bar{b} \in B^k$ we have $\langle \mathbf{A}, \bar{a}\rangle \equiv_k \langle \mathbf{B}, \bar{b}\rangle$ iff $\langle \mathbf{A}_{\restriction U \cup \{a_1,\ldots,a_k\}}, \bar{a}\rangle \cong \langle \mathbf{B}_{\restriction U \cup \{b_1,\ldots,b_k\}}, \bar{b}\rangle$.*
2. *$\pi_k(\mathbf{A})$ is uniquely determined by the isomorphism type of $\mathbf{A}_{\restriction U}$.*
3. *$|\pi_k(\mathbf{A})| = 2^{O(|U|)}$.*

Proof. 1. We show that Duplicator has a winning strategy in the second phase of the game whenever after the first one the pebbles are placed on $\bar{a}$ and $\bar{b}$ in $\mathbf{A}$ and $\mathbf{B}$, respectively. This implies Item 1. by Theorem 4.4.

The strategy of Duplicator in this is to fix some isomorphism $\alpha : \mathbf{A}_{\restriction U} \to \mathbf{B}_{\restriction U}$ and then keep all the time the initial property of the situation:

(*) substructures $\langle \mathbf{A}_{\restriction U \cup \{a_1,\ldots,a_k\}}, \bar{a}\rangle$ and $\langle \mathbf{B}_{\restriction U \cup \{b_1,\ldots,b_k\}}, \bar{b}\rangle$ are isomorphic by an extension of α.

Certainly condition (*) suffices in order not to loose immediately, and, as we will see, guarantees also the ability to continue this strategy in the next moves. (Recall that in these substructures the symbol U is interpreted, and therefore it is still clear, which of the elements of $\bar{a}$ and $\bar{b}$ do/do not belong to it.)

Namely when Spoiler moves a pebble in one of the structures in, say $\mathbf{A}$, there are three possibilities: either the target element is in U, and then Duplicator answers according to α, or the target element is one of the already pebbles ones, but not in U—in this case Duplicator answers using (*), or the move of Spoiler determines a k, U-extension type t, namely the one which describes relations between the newly pebbled element and the remaining pebbled elements and all elements in U. Since $Ext_{k,U}(t)$ is true in $\mathbf{B}$ too, Duplicator is guaranteed to find an element there to place his pebble in a way preserving isomorphism.

In the first phase $k-1$ moves Duplicator must "imagine" some additional pebbles placed in $\mathbf{A}$ and $\mathbf{B}$ in order to respond. (Because k, U-extension axioms say how to extend $k-1$-element configurations, while Duplicator needs to extend smaller ones.) It is easy to see that Duplicator can follow this strategy forever, despite of moves of Spoiler. This gives us 1.

To get 2. and 3. note that in every structure satisfying all k, U-extension axioms there are all possible extensions of $\mathbf{U}$ by at most k elements. So if $\mathbf{U}$'s are isomorphic, so are π_k's. One easily checks that if the isomorphism type of $\mathbf{U}$ is fixed, there are $2^{O(|U|)}$ such isomorphism types of extensions, and therefore exactly that many elements in π_k. □

Lemma 6.12. *For every $k > 1$ the conjunction of all k, U-extension axioms is equivalent in $\overline{\Sigma}'$ to a first order sentence, which we denote $Ext_{k,U}$.*

Proof. For the sake of simplicity we assume that σ consists of one binary relation symbol R. We have to express in first order logic the conjunction

$$\bigwedge_{t \subseteq At_{k,U}} \forall x_1 \dots x_{k-1} \begin{pmatrix} \bigwedge_{i \neq j} x_i \neq x_k \\ \bigwedge_i x_i \notin U \end{pmatrix} \to \exists x_k \begin{pmatrix} x_k \neq x_1, \dots x_{k-1} \\ x_k \notin U \\ (\bigwedge t) \\ \neg(\bigvee At_{k,U} \setminus t) \end{pmatrix}. \tag{6.2}$$

Now consider the following monadic second order sentence:

$$\begin{matrix} \forall x_1, \dots, x_{k-1} \\ \forall X, Y \subseteq U \cup \{x_1 \dots x_{k-1}\} \end{matrix} \begin{pmatrix} \bigwedge_{i \neq j} x_i \neq x_k \\ \bigwedge_i x_i \notin U \end{pmatrix} \to \exists x_k \begin{pmatrix} & x_k \notin U \cup \{x_1, \dots, x_{k-1}\} \\ \forall u \in U & u \in X \leftrightarrow R(x_k, u) \\ \forall u \in U & u \in Y \leftrightarrow R(u, x_k)) \end{pmatrix}. \tag{6.3}$$

It can be easily seen to be logically equivalent to (6.2). What remains to be done is to express the monadic second order quantification in it by first order means.

We do it as follows. Let us analyze the following sentence:

$$\begin{matrix} \exists x \forall u \in U \ \neg R(x, u)) \\ \wedge \\ \forall x \forall v \in U \exists y \forall u \in U \ (R(x, u) \leftrightarrow (R(y, u) \vee v = u)). \end{matrix} \tag{6.4}$$

We say that $X \subseteq U$ *is coded by* x iff $R(x, u)$ for all $u \in X$ and for none $u \in U \setminus X$. The first line in (6.4) says that $\emptyset$ is coded by some x, and the second says that for every coded $X \subseteq U$ and every $u \in U$ the set $X \cup \{u\}$ is also coded. Together they say that every subset of U is coded, since U is finite. This gives us a method to speak about subsets of U in first order logic. Note also the important fact that (6.4) follows from the conjunction of all $2, U$-extension axioms.

Now we replace in (6.3) each set quantifier $\forall X \subseteq U$ by $\forall x$ and each corresponding $u \in X$ by $R(u, x)$ (and similarly for Y); since we quantify there over subsets of $U \cup \{x_1 \dots x_{k-1}\}$, we must also add explicitly all possibilities $x_i \in X$ and $x_i \notin X$ expressed in first order logic. As a result we get a first order sentence, denoted γ_k.

It is easy to see that if (6.4) is true in $\mathbf{A} \in \overline{\Sigma}'$, then this encoding gives a formula equivalent in $\mathbf{A}$ to (6.3).

We claim that the conjunction of $\gamma_k \wedge (6.4)$ is equivalent in $\overline{\Sigma}'$ to (6.2). Indeed, let $\mathbf{A} \in \overline{\Sigma}'$.

If $\gamma_k \wedge (6.4)$ it is true in $\mathbf{A} \in \overline{\Sigma}'$, then the sentence (6.4) is true in $\mathbf{A}$, which in turn means that the first-order encoding γ_k of (6.3) is equivalent in $\mathbf{A}$ to (6.2), which must be therefore also true in $\mathbf{A}$.

Other way round, if (6.2) is true in $\mathbf{A}$, then (6.4) must be true there too. This means that γ_k is indeed equivalent in $\mathbf{A}$ to (6.3), and thus to (6.2) too. Therefore the conjunction $\gamma_k \wedge (6.4)$ is true in $\mathbf{A}$, what concludes the proof. □

6.1.3 Average values

Definition 6.13. Let X be a finite set, and $f : X \to \mathbf{N}$ any function. Then *the average value of f on X* is defined as

$$\operatorname*{average}_{x \in X}(f(x)) = \frac{\sum_{x \in X} f(x)}{|X|}.$$

This average is equal to the expected value of f with respect to the probability distribution on X that makes all the instances $x \in X$ equiprobable, thus e.g.

$$\operatorname*{average}_{\mathbf{A} \in \overline{\Sigma}(n)}(f(\mathbf{A})) = \sum_{\mathbf{A} \in \overline{\Sigma}(n)} f(\mathbf{A}) \cdot \mu_n^{\overline{\Sigma}}(\{\mathbf{A}\}).$$

When the argument to f is clear, we will omit it in the notation, writing $\text{average}_X(f)$. □

The typical functions f we will consider are running times of algorithms and other computational resources.

6.1.4 Boolean circuits

Boolean circuits are nonuniform parallel computing devices. Nonuniform, since for inputs of each length there is a separate Boolean circuit, which should implement the requested function/decision procedure solely for inputs of this length. For functions the length of the output must be also fixed. Thus formally a general function/decision problem is computed not by a single Boolean circuit, but by a sequence of such circuits.

A Boolean circuit is a finite acyclic graph with labeled nodes. The in- and out-degree of nodes, for Boolean circuits called *fan-in* and *fan-out*, respectively, are generally restricted by the label. The labels can be either Boolean operations $\wedge$, $\vee$ and $\neg$, or in_i, out_i. Vertices labeled with $\wedge$ and $\vee$ are allowed to have arbitrary positive fan-in, while those labeled by $\neg$ must have fan-in 1. Fan-out is unrestricted for all kinds of nodes, with one exception mentioned below. A circuit for computing with inputs of length n has precisely n nodes $in_1, \ldots, in_n$ with fan-in 0, to which the input is given. A circuit for computing outputs of length n has precisely n nodes $out_1, \ldots, out_n$ with fan-out 0 and fan-in 1, from which the output is read.

We will always indicate the number of bits of the input in a circuit by a subscript, e.g., C_n is a circuit for inputs of length n.

The computation of a given circuit C_n is performed by assigning the consecutive bits of the input word to the input nodes. Then all the nodes in the graph are recursively assigned 0's and 1's in the obvious manner: e.g., nodes labeled with $\wedge$ get 1 if all their predecessors are assigned 1, and 0 otherwise. The binary word assigned to the sequence of output nodes is the output.

The measures of complexity associated with a Boolean circuit C_n we need here is the *depth* $d(C_n)$, i.e., the length of the longest path from some input node to some output node, and *size* $s(C_n)$, that is the cardinality of the circuit.

Roughly speaking, $s(C_n)$ is the amount of hardware necessary to compute the function/solve the problem in parallel in time $d(C_n)$ for inputs of length n. So the dependence of $d(C_n)$ and $s(C_n)$ on n give the parallel complexity of the whole function/problem. One can impose some uniformity conditions by restricting the attention to sequences of circuits which are computable in some complexity classes, like DSPACE($\log n$) or PTIME.

In this paper we will be mostly interested in parallel time complexity $d(C_n)$ with the restriction that $s(C_n)$ is polynomial. In particular, the (nonuniform) AC^0 consists of all functions which are computable by Boolean circuits of constant depth and polynomial size.

We want to speak about average complexity of problems. To accommodate this notion in the realm of Boolean circuits, in which the complexity is assigned by definition to all inputs of the same length, and not to each input separately, we need some small changes. We will assign inputs to two circuits in parallel, and assume that each of them has a special, separate output node *output-here,* which, if assigned 1, means that this circuit actually computes the output. Then the parallel time for a given input is the minimal depth of a circuit which actually computes the output. Typically one circuit has quite small depth, while the other implements the naive procedure and has very large depth, but is activated only for a very small fraction of inputs.

To get used to the circuits, we prove the following fact, which will be useful in the future as well.

Theorem 6.14 (Immerman [13]). *Every* FO^k *query is computable in* AC^0.

Proof. Let us remind that according to the definition, a structure $\mathbf{A} \in \overline{\Sigma}$ will be given to the circuit as $\overline{\mathrm{enc}}(\mathbf{A})$, and therefore for all $\mathbf{A}$'s of the same cardinality the encodings will be of the same length. This makes our circuits to work on input all structures of equal cardinalities. The consecutive bits of the encoding correspond to truth values of all the atoms $R(\bar{a})$ for $R \in \sigma$ and $\bar{a} \in A^k$ of appropriate length. We will thus use these atoms as names of the input nodes in the circuit to be constructed. Similarly the output nodes can be identified with elements of A^k.

The construction is recursive w.r.t. to the construction of a FO^k formula φ. For simplicity we assume that $\sigma = \{R\}$ and that R is binary.

If φ is an atom $R(x_i, x_j)$, $1 \leq i, j \leq k$, then we get the circuit by connecting for all $\langle a_1, \ldots, a_k \rangle \in A^k$ all input nodes $R(a_i, a_j)$ to output nodes $\langle a_1, \ldots, a_k \rangle$.

If φ is a conjunction $\varphi_1 \wedge \varphi_2$ then we get the circuit for it from the circuits for φ_1 and φ_2 as in the Figure 1.

The negation $\neg\varphi$ is handled similarly, i.e., we add negation nodes to the output nodes of the circuit for φ and then output nodes of the whole circuit.

The case of $\exists x_i \varphi$ is handled in the following way: for each $\langle a_1, \ldots, a_i, a_{i+1}, \ldots, a_k \rangle \in A^{k-1}$ the output nodes of the circuit for φ corresponding to all tuples in the Cartesian product $\{a_1\} \times \ldots \{a_{i-1}\} \times A \times \{a_{i+1}\} \times \ldots \times \{a_k\}$ are connected to a single $\vee$ node, and this one to all output nodes in the same set. Informally speaking, it corresponds to expressing $\exists x_i\ \varphi(x_i)$ by $\bigvee_{a_i \in A} \varphi(a_i)$.

This finishes the construction. It is easy to see, that the resulting circuit contains only polynomially many nodes and that its depth is at most 3 time larger than the depth of the syntax tree of φ. □

And now one application of this theorem:

Proposition 6.15. *The isomorphism query for two finite structures from* $\overline{\Sigma}$ *ordered by the natural ordering and of the same universe set is in* AC^0.

Proof. In virtue of Theorem 6.14 it is enough to show, that this query is first order. It suffices to demonstrate it for structures with single k-ary relation in addition to the ordering $\preceq$.

We can assume that the relation of the first structure is R and in the second R'. Then the sentence we need asserts that the $\preceq$-lexicographically-minimal k-tuple $\mathbf{min}$ satisfies $R(\mathbf{min}) \leftrightarrow R'(\mathbf{min})$, and further that for every $\bar{x}$, if $R(\bar{x}) \leftrightarrow R'(\bar{x})$ then for the immediate $\preceq$-lexicographic-successor $\bar{y}$ of $\bar{x}$ holds $R(\bar{y}) \leftrightarrow R'(\bar{y})$. Finiteness of structures guarantees, that this induction-based sentence indeed expresses isomorphism. □

And now a bit deeper circuit:

Proposition 6.16. *If* $\overline{D} \subseteq \overline{\Sigma}$ *then any formula* $[\mathrm{lfp}\ \xi(X), X](\bar{x})$ *can be evaluated in structures from* $\overline{D}$ *by a sequence* C_n *of circuits of polynomial size and depth linear in* $\max\{|\pi_k(\mathbf{A})| : \mathbf{A} \in \overline{D}\ \&\ |A| = n\}$.

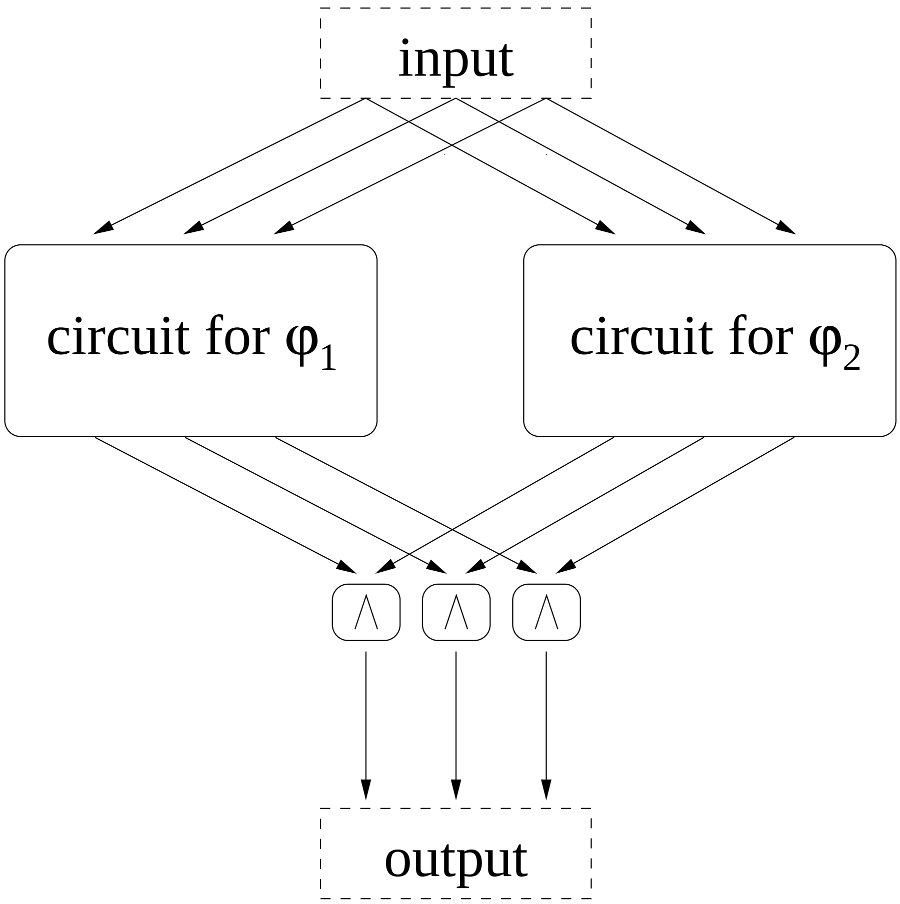

Figure 1: A circuit for conjunction

Proof. It is easy to see that the number of iterations necessary to reach the fixpoint doesn't exceed $|\pi_k(\mathbf{A})|$. Indeed, if it were greater, some two $\equiv_k$-tuples in A^k could be distinguished by a FO^k formula defining the stages of the fixpoint, in contrary to the definition of $\equiv_k$. Consequently the circuit from Figure 2 computes the fixpoint, and each of the levels has (the same) constant depth, since it implements a first order definable operation. □

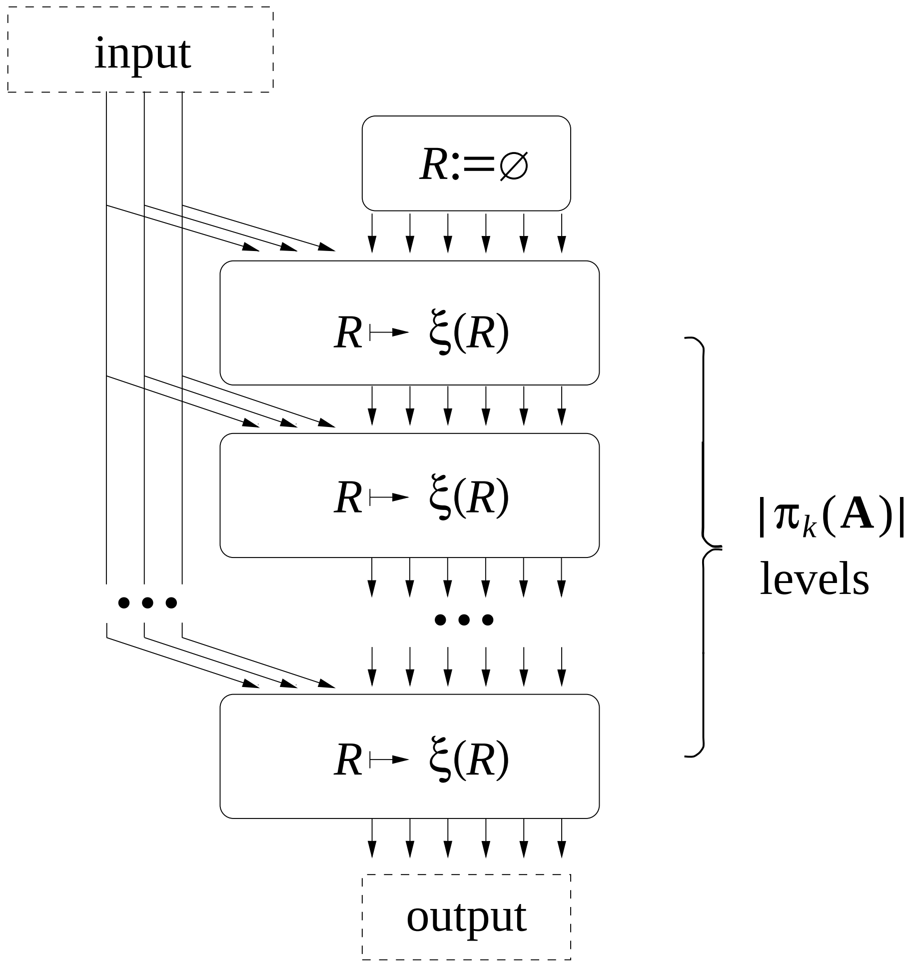

Figure 2: Circuit for computing fixpoint formula $[\mathrm{lfp}\ \xi(X), X](\bar{x})$.

6.2 The Most Probable Kolmogorov Complexity of Structures

Theorem 6.17. *For any function $\alpha(n) \to \infty$ holds*

$$\mathrm{cp}_\sigma(n) - \log n! - \alpha(n) \leq C(\mathbf{A}) \leq \mathrm{cp}_\sigma(n) - \log n! + O(1)$$

for almost every structure $\mathbf{A} \in \overline{\Sigma}(n)$.

Proof. Before we begin the proof, let us note a slight mismatch appearing in this theorem. Namely, we use isomorphism types of structures to define the information content, while we consider the "most frequent" value of complexity over all structures of given fixed universe, and not over isomorphism types of such structures. So each isomorphism type can be represented in $\overline{\Sigma}(n)$ in more than one copy. The precise number of copies depends on the number of automorphisms of it. However, in the context of databases $\mu^{\overline{\Sigma}}$ seems to be more natural choice of the probability distribution than μ^Σ (if there is any natural choice at all). But Theorem 6.5 and observation that $C(\mathbf{A})$ is a property of the isomorphism type of $\mathbf{A}$ allow us to compute the probabilities in Σ.

1. Let $\Sigma(\leq n)$ stand for $\bigcup_{i=1}^n \Sigma(i)$. By Theorem 6.5 we get $|\Sigma(n)| \geq 2^{\mathrm{cp}_\sigma(n) - n\log n}$, where $\deg(\mathrm{cp}_\sigma) \geq 2$. So $|\Sigma(n)|$ grows at least as fast as 2^{cn^2}, for some $c > 0$. Therefore

$$\frac{|\Sigma(\leq n)|}{|\Sigma(n)|} = \frac{\sum_{j=1}^n |\Sigma(j)|}{|\Sigma(n)|} = 1 + \frac{\sum_{j=1}^{n-1} |\Sigma(j)|}{|\Sigma(n)|} = 1 + o(1).$$

   Consequently almost every structure $\mathbf{A} \in \Sigma$ of at most n elements has precisely n elements.

2. Let w denote $|\Sigma(\leq n)|$. There are at most $2^{\log w - \alpha(n)} - 1$ binary words of length at most $\log w - \alpha(n) - 1$, hence (compare proof of Proposition 2.5)

$$\frac{|\{\mathbf{A} \in \Sigma(\leq n) \ : \ C(\mathbf{A}) < \log w - \alpha(n)\}|}{w} \leq 2^{-\alpha(n)},$$

   which implies that almost every structure in $\Sigma(\leq n)$ is of complexity at least $\log w - \alpha(n)$.

3. Items 1. and 2. together imply, that almost every structure $\mathbf{A} \in \Sigma(n)$ has complexity at least $\log w - \alpha(n)$.

4. Item 1. and Theorem 6.5 give us

$$w = (1 + o(1)) \cdot 2^{\mathrm{cp}_\sigma(n)}/n!,$$

   and therefore

$$\log w = \mathrm{cp}_\sigma(n) - \log n! + o(1).$$

Now one direction of the inequality follows. The other is trivial, since no structure in $\Sigma(n)$ can have complexity greater than $\mathrm{cp}_\sigma(n) - \log n! + O(1)$ — the description of this length is simply the lexicographic number of the structure in Σ (recall Proposition 2.5), and this number can be at most $(1 + o(1)) \cdot 2^{\mathrm{cp}_\sigma(n)}/n!$. □

Let us analyze an example situation, exactly the same as considered by the authors in [1]. They assumed equal probabilities on all n element input structures over σ, and showed a surprising result that the average case sequential complexity of PFP queries is DSPACE($\log n$), and that the average parallel complexity of LFP is in uniform AC^0, i.e., the queries are computable on a PRAM with polynomially many processors in constant average time. What we show now, is that the price to pay for this is very high for a user that uses only Boolean queries, in the sense that on average he can get only a constant number of bits of information about the database, out of the quantity established in the proposition above.

Theorem 6.18. *Let* 0 *be the constant function equal zero. Then for every* $k \in \mathbf{N}$

$$I_{\mathrm{FO}^k}(\mathbf{A}) = O(1)$$

for almost every structure $\mathbf{A} \in \overline{\Sigma}^0(n)$.

Proof. It will follow from the more general Theorem 6.20 below. □

Now our tour really begins, after we have seen what happens when $i(n) = 0$. First we consider the average Kolmogorov expressive power and the average query complexity in all such structures.

Theorem 6.19. *Let $i(n) \le n/2$ be as above. Then for any function $\alpha(n) \to \infty$*

$$\mathrm{cp}_\sigma(n) - \log(n-i)! - \log i! - \alpha(n) \le C(\mathbf{A}) \le \mathrm{cp}_\sigma(n) - \log(n-i)! - \log i! + O(1)$$

for almost every $\mathbf{A} \in \overline{\Sigma}^i(n)$.

Proof. After a short computation

$$\begin{aligned} |\overline{\Sigma}^i(n)| &= (1+o(1))\binom{n}{i} 2^{\mathrm{cp}_\sigma(n)}/n! \\ &= (1+o(1))\frac{1}{(n-i!)i!} 2^{\mathrm{cp}_\sigma(n)} \end{aligned}$$

the rest of the proof follows the line of the proof of Theorem 6.17. □

It appears that the Kolmogorov expressive power of the query languages depends in a strong and rather unexpected way on the growth rate of $i(n)$:

Theorem 6.20. *For arbitrary function $i(n) \le n/2$:*

1. *If $i(n)/\log n \to 0$, then for every k large enough*

   $$I_{\mathrm{FO}^k}(\mathbf{A}) = \mathrm{cp}_\sigma(i) - \log i! + O(\log i) \tag{6.5}$$

   for almost every $\mathbf{A} \in \overline{\Sigma}^i(n)$.

2. *If $i(n)/\log n \to \infty$, then for each $k \in \mathbf{N}$*

   $$I_{\mathrm{PFP}^k}(\mathbf{A}) = O(1) \tag{6.6}$$

   for almost every $\mathbf{A} \in \overline{\Sigma}^i(n)$.

It should be noted, that Knyazev [15], using the many-sorted formulation, has proven the 0-1 law for first order logic for classes in Item 2 of our theorem, and noted that this law doesn't hold for $\overline{\Sigma}^i$ with $i(n)/logn \not\to 0, \infty$. His result is related to what we do here in Item 2.

Proof. 1. First we prove the direction $\le$ of 1. Namely, if $\mathbf{A}, \mathbf{B} \in \Sigma(n)$ are such that $\mathbf{A}_{\restriction U} \cong \mathbf{B}_{\restriction U}$, and both satisfy all k, U-extension axioms, then by Lemma 6.11 Duplicator has a winning strategy in the game played on $\mathbf{A}$ and $\mathbf{B}$. Therefore $\mathrm{FO}^k[\mathbf{A}] = \mathrm{FO}^k[\mathbf{B}]$. It follows that for $\mathbf{A}$ satisfying all k, U-extension axioms the sequence $\mathrm{FO}^k[\mathbf{A}]$ can be reconstructed from the isomorphism type of $\mathbf{A}_{\restriction U}$ and finitely many additional bits, which describe the following reconstruction method: One should search all structures in Σ looking for the first one, say $\mathbf{B}$, satisfying all k, U-extension axioms and such that $\mathbf{B}_{\restriction U} \cong \mathbf{A}_{\restriction U}$. This search will stop, since such a structure exists (namely, $\mathbf{A}$ itself). When such $\mathbf{B}$ is found, it is used to reconstruct $\mathrm{FO}^k[\mathbf{A}]$, since $\mathrm{FO}^k[\mathbf{A}] = \mathrm{FO}^k[\mathbf{B}]$. We get $C(\mathrm{FO}^k[\mathbf{A}]) \le C(\mathbf{A}_{\restriction U}) + O(1)$, and then by Lemma 5.7 $I_{\mathrm{PFP}^k}(\mathbf{A}) \le C(\mathbf{A}_{\restriction U}) + O(\log C(\mathbf{A}_{\restriction U}))$. This already implies the inequality, since $C(\mathbf{A}_{\restriction U}) \le \mathrm{cp}_\sigma(|U|) - \log(|U|!) + O(1)$.

Now we turn to $\geq$ of 1. In order to establish this inequality we show, that there is a fixed k such that $\mathbf{A}_{\restriction U}$ can be reconstructed from $\mathrm{FO}^k[\mathbf{A}]$ for almost all $\mathbf{A} \in \overline{\Sigma}^i(n)$.

Let us now return to the formula (6.4). We have noted that it holds in almost every $\mathbf{A}$, since it follows from the conjunction of all $2, U$-extension axioms. As is easy to see, if this formula is satisfied in $\mathbf{A}$, then every subset of U is definable in first order logic of whole $\mathbf{A}$. (We have already made use of it in the proof of Lemma 6.12.) Hence, as it is not hard to see, the monadic second order logic of $\mathbf{A}_{\restriction U}$ is first order definable in $\mathbf{A}$.

Let $E \subseteq \overline{\Sigma}^i(n)$ consist of those structures $\mathbf{A}$, which satisfy

(i) $\mathbf{A}_{\restriction A \setminus U}$ is rigid,

(ii) $\mathbf{A}_{\restriction U}$ is rigid,

(iii) $\mathbf{A}_{\restriction U}$ is ordered by the Kaufmann's formula from Lemma 6.10,

(iv) complexity restriction $C(\mathbf{A}) \geq \mathrm{cp}_\sigma(n) - \log(n-i)! - \log i! - \log i$ is satisfied and

(v) $\mathbf{A}$ satisfies all k, U-extension axioms.

Let $S \subseteq A$. We can assume that in the process of constructing a random $\mathbf{A} \in \overline{\Sigma}'$ by coin tosses (recall Remark 6.6), first the edges of $\mathbf{A}_{\restriction S}$ are drawn, and only then the remaining ones. So if a property of $\mathbf{A}$ depends on $\mathbf{A}_{\restriction S}$ merely, we can stop after having edges in it and look if it satisfies this property or not. The probability won't change in this "reduced" experiment. This argument shows, that if almost every possible substructure $\mathbf{A}_{\restriction S}$ guarantees the truth of a property φ in $\mathbf{A}$, then almost every $\mathbf{A}$ has φ.

Now all the conditions (i)-(v) hold almost surely either of $\mathbf{A}$ or of $\mathbf{A}_{\restriction U}$ or of $\mathbf{A}_{\restriction A\setminus U}$, namely (i) and (ii) by Theorem 6.5, (iii) by Lemma 6.10, (iv) by Theorem 6.19 and (v) by Lemma 6.9. Therefore all of them hold almost surely in $\overline{\Sigma}^i$, and since there are finitely many of them, all of them hold simultaneously almost surely.

What remains to be done is to prove that there is a sufficiently large k such that for every structure $\mathbf{A} \in E$ holds

$$I_{\mathrm{FO}^k}(\mathbf{A}) = C(\mathbf{A}) - C(\mathbf{A} \mid \mathrm{FO}^k[\mathbf{A}]) \geq C(\mathbf{A}_{\restriction U}) - O(\log i(n)). \tag{6.7}$$

Firstly, from $\mathrm{FO}^k[\mathbf{A}]$ the isomorphism type of $\mathbf{A}_{\restriction U}$ can be reconstructed. Namely the Kaufmann's formula gives an ordering of $\mathbf{A}_{\restriction U}$ in every $\mathbf{A} \in E$ by (iii), and with this ordering and few extra variables it is possible to write a query describing $\mathbf{A}_{\restriction U}$ up to isomorphism (compare Proposition 5.5).

It follows, that in order to reconstruct $\mathbf{A}$ from $\mathrm{FO}^k[\mathbf{A}]$ we need as an additional information only:

(a) a description of $\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]$, i.e., of $\mathbf{A}$ with all edges inside U removed,

(b) the way to insert $\mathbf{A}_{\restriction U}$ into $\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]$ to get $\mathbf{A}$, and

(c) the bits necessary to separate these two strings.

Concerning (b) there are at most $i!$ ways to insert $\mathbf{A}_{\restriction U}$ into $\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]$. Hence (b) can be described by at most $\log i! + O(1)$ bits and consequently (c) requires $O(\log\log i!) = O(\log i)$ bits. This means that for $\mathbf{A} \in E$

$$
\begin{aligned}
I_{\mathrm{FO}^k}(\mathbf{A}) &= C(\mathbf{A}) - C(\mathbf{A} \,|\, \mathrm{FO}^k[\mathbf{A}]) \\
&\geq C(\mathbf{A}) - C(\mathbf{A} \,|\, \mathbf{A}_{\restriction U}) \\
&\geq \underbrace{\mathrm{cp}_\sigma(n) - \log(n-i)! - \log i! - \log i}_{\text{by (ii)}} - (C(\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]) + \log i! + O(\log i)) \\
&= \mathrm{cp}_\sigma(n) - C(\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]) - \log(n-i)! - 2\log i! + O(\log i).
\end{aligned}
$$

Consequently in order to prove (6.5) we need to show only that

$$
C(\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]) \leq \mathrm{cp}_\sigma(n) - \mathrm{cp}_\sigma(i) - \log(n-i)! - \log i! + O(1) \tag{6.8}
$$

holds for $\mathbf{A} \in E$. Let $\overline{F} \subseteq \overline{\Sigma}$ stand for $\{\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset] \ : \ \mathbf{A} \in \overline{E}\}$.

Note that all structures in F are rigid. Indeed, let $\mathbf{A} \in E$. As $\mathbf{A}_{\restriction A \setminus U}$ is rigid by (i), all automorphisms of $\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]$ must be identity outside of U. Suppose that α is a nontrivial automorphism of $\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]$, i.e., let $\alpha(u) = u'$, for some $u \neq u'$ from U. But all the k, U-extension axioms are true in $\mathbf{A}$, hence there is $v \in A \setminus U$ such that $R(v,u)$ but $\neg R(v,u')$. Since $\alpha(v) = v$, we get a contradiction.

It is trivially seen, that $|\overline{F}| \leq 2^{\mathrm{cp}_\sigma(n) - \mathrm{cp}_\sigma(i)}$. Indeed, there are precisely $\mathrm{cp}_\sigma(n) - \mathrm{cp}_\sigma(i)$ atoms to be decided to get a structure in $\overline{F}$, and all different choices give different structures. However, not all the possible choices lead to structures being in $\overline{F}$. Now remind that every $\mathbf{A}[\mathbf{A}_{\restriction U} := \emptyset]$ is rigid for $\mathbf{A} \in E$. This means that each isomorphism type found in F is represented by $i!(n-i)!$ copies in $\overline{F}$, since we can permute elements in U in any way, and elements in $A \setminus U$ in any way, getting each time different but isomorphic structures from $\overline{F}$. Therefore

$$
|F| \leq \frac{2^{\mathrm{cp}_\sigma(n) - \mathrm{cp}_\sigma(i)}}{(n-i)!i!},
$$

and this by Lemma 2.5 implies (6.8).

Thus the proof of 1. is finished.

To get 2. note, that by Lemma 6.9 Item 2., the logical properties of almost all structures in $\overline{\Sigma}^i$ are precisely the same as in $\overline{\Sigma}'$, i.e., in the class of structures obtained by starting from σ', adding U' and setting its cardinality to 0. Thus 2. follows from 1. applied to such "shifted" class of structures. □

Remark 6.21. Our result above supports the idea of marking elements as important. What we mark as important, becomes really visible to the query language—provided we do not want too many elements to be important. This follows from the analysis of the proof of the last theorem. It appears that almost always the query language allows us to reconstruct the isomorphism type of $\mathbf{A}_{\restriction U}$—the substructure generated by the important elements, while tuples made of remaining elements differ only by their local relations and the relations to important elements.

6.3 Split Optimization

We begin this section with a simple fact: we will see, that Theorem 5.10 gives not only a connection between $\pi_k(\mathbf{A})$ and $I_{\mathrm{FO}^k}(\mathbf{A})$, but implies also consequences concerning optimization by split.

Proposition 6.22. *Suppose that $\overline{D} \subseteq \overline{\Sigma}$ is infinite and let*

$$f(n) = \max_{\substack{\mathbf{A}\in\overline{D}\\ |A|=n}} |\pi_k(\mathbf{A})|.$$

Then for every l there are queries in LFP, *which for arguments from $\overline{D}$ require sequential time $f(n)^l$ if evaluated using split as $\pi_k \circ \varphi^{\pi_k}$, and* PFP *queries which require space $f(n)^l$ for the same evaluation method.*

Proof. $\pi_k(\mathbf{A})$ is ordered and asymptotically has cardinality at least $f(|A|)/5\log(f(|A|))$ for structures in $\overline{D}$. In LFP^{2k+2} we can define $\pi_k(\mathbf{A})$, and having 3 vectors of variables of k variables each we can distinguish all elements in $\pi_k(\mathbf{A})$. Hence it is easily seen that for every fixed l we can write a $3kl$ ary fixpoint whose evaluation needs the given time (space) to be evaluated in the naive semantics, depending on if it is a least or partial fixed point. □

The statement shows, that whenever the optimization by *split* gives any real benefit, due to small size of $|\pi_k(\mathbf{A})|$, it indicates a limitation of the expressive power of Boolean queries. Namely if we want to achieve fast evaluation of queries by *split*, we want f to grow as slowly as possible. But the slower f grows, the smaller is the value of I_{FO^k}, since it is bounded by a fixed, in fact almost linear, function of f.

We have just seen, that successful optimization by *split* technique implies a bound on the Kolmogorov expressive power of the query language. For classes $\overline{\Sigma}^i$ this expressive power is in most of structures limited. The question is: can we prove in these cases that the queries have lower average complexity than the pessimistic PTIME for LFP and PSPACE for PFP? It this section we prove by an application of split an average speed-up for PFP queries.

Theorem 6.23 (Space complexity of PFP**).** *1. If $i(n)/\log n \to 0$, then on average* PFP *queries in $\overline{\Sigma}^i(n)$ are computable in* $\mathrm{DSPACE}(\max(\log n, 2^{O(i(n))}))$.

2. If $i(n)/\log n \to \infty$, then on average PFP *queries in $\overline{\Sigma}^i(n)$ are computable in* $\mathrm{DSPACE}(\log n)$.

Proof. We begin with item 1. Suppose $i(n)/\log n \to 0$.

The bounded-space machine we need, given query φ, executes the following program opt(φ):

$$\textbf{if } Ext_{k,U} \textbf{ then } \widetilde{\varphi} \textbf{ else } \varphi. \tag{6.9}$$

The test $Ext_{k,U}$ is the first order equivalent of the conjunction of all k, U-extension axioms, Lemma 6.12, and can be performed in $\mathrm{DSPACE}(\log n)$.

The unoptimized procedure φ is computed according to the naive semantics and requires $\mathrm{DSPACE}(n^l)$ for some fixed l.

The optimized procedure $\widetilde{\varphi}$ is executed as follows: the machine, given input $\mathbf{A} \in \overline{\Sigma}^i(n)$, first extracts $\mathbf{A}_{|U}$, and then considers consecutively all possible structures $\mathbf{B} \in \overline{\Sigma}^j(m)$, where $m = 2^{2|U|}$ and $j = |U|$, and such that $\mathbf{B}_{|U} \cong \mathbf{A}_{|U}$. For each considered structure it tests if it satisfies $Ext_{k,U}$. Note that this brute force search doesn't require more space than polynomial in $2^{2|U|}$, i.e., $2^{O(|U|)}$. As we have $\log n/m \le 1/2$, for all large n the machine will find a structure $\mathbf{B}$ satisfying all k, U-extension axioms, by Lemma 6.9. Test if this is the case requires $\log m \le \log n$ space. Then it evaluates φ in this $\mathbf{B}$ using the naive semantics. It requires space polynomial in m, hence again is doable in $\mathrm{DSPACE}(2^{O(|U|)})$. From this evaluation the machine recognizes descriptions of all $\equiv_k$-classes that should be included in the output. The description of these classes is limited to the description of relations between the elements in the tuple and between these elements and elements in $\mathbf{B}_{|U}$, therefore fits in space $2^{O(|U|)}$. Then these descriptions are used to decide what tuples from $\mathbf{A}$ must be included in the output, and again for each separate tuple it doesn't require more space than $2^{O(|U|)}$, while cycling through all of them requires logarithmic space.

Generally, if $\mathbf{A} \models Ext_{k,U}$ then the whole computation of $\mathrm{opt}(\varphi)$ (including the test itself) requires space $\max(\log n, 2^{O(i(n))})$, and if not, then n^l. But the second possibility happens only very seldom—in $o(n^{-m})$ fraction of all cases, where m is arbitrary. Hence setting $m = l + 1$ we get

$$\begin{aligned}
(6.10) \qquad \operatorname*{average}_{\mathbf{A}\in\overline{\Sigma}^i(n)} space(\mathrm{opt}(\varphi)) &\leq \frac{\sum_{\mathbf{A}\in\overline{\Sigma}^i(n)} space(\mathrm{opt}(\varphi))}{|\overline{\Sigma}^i|} \\
&= \frac{\sum_{\substack{\mathbf{A}\in\overline{\Sigma}^i(n)\\ \mathbf{A}\models Ext_{k,U}}} \max(\log n, 2^{O(i(n))})}{|\overline{\Sigma}^i|} + \frac{\sum_{\substack{\mathbf{A}\in\overline{\Sigma}^i(n)\\ \mathbf{A}\not\models Ext_{k,U}}} n^l}{|\overline{\Sigma}^i|} \\
&\leq \max(\log n, 2^{O(i(n))}) + n^l \cdot o(n^{-m}) \\
&= \max(\log n, 2^{O(i(n))}).
\end{aligned}$$

In the following proofs we won't present this calculation any more, since it is always almost identical, and the result equals the complexity of the dominating case.

To get Item 2 note, that in this case $\pi_k(\mathbf{A})$ doesn't depend on $\mathbf{A}$ whenever $Ext_{k,U}$ holds, hence the procedure $\widetilde{\varphi}$ is just a first order query. Therefore the complexity of the dominating case is $\mathrm{DSPACE}(\log n)$, and so is the average complexity. $\square$

6.4 Beyond Split

In this section we discuss optimization of LFP queries. We will apply a technique, which can be seen a next (and the last at the same time) step beyond split. Split is based conceptually on the fact, that the evaluation of queries can be done, instead of the real structure $\mathbf{A}$, in its compressed representation $\pi_k(\mathbf{A})$, which in case the latter is smaller gives a speed-up. But sometimes we can compress the information about $\mathbf{A}$ even much more without loosing information: to a string of length $C(\mathrm{FO}^k[\mathbf{A}])$. By the definition of Kolmogorov complexity this is the maximal possible compression. Can we use this to get more speed-up than with split?

In the previous sections we have seen that in classes $\overline{\Sigma}^i$ for $i = o(\log n)$ for a typical structure $\mathbf{A}$ the value of $C(\mathrm{FO}^k\mathbf{A})$ is exponentially smaller than $|\pi_k(\mathbf{A})|$. So it is a model situation to try to answer our question. And we succeed, as the following theorem shows.

Theorem 6.24 (Parallel complexity). *For arbitrary function $i(n) \leq n/2$ we have the following:*

1. *If $i(n)/\sqrt[s]{\log n} \to 0$, where $s = \deg \mathrm{cp}_\sigma$, then* LFP *queries are computable on average in* AC^0, *i.e., by a polynomial size circuit in average constant time.*
2. *If $i(n)/\log n \to 0$, then in parallel every* LFP *query can be computed by a polynomial size circuit in average time $2^{O(i(n))}$.*
3. *If $i(n)/\log n \to \infty$, then on average* LFP *queries can be computed in* AC^0.

Proof. We begin with the easiest item 2. Suppose for simplicity that $\varphi = [\mathrm{lfp}\ \xi(X), X](\bar{x})$. Recall that the mapping $X \mapsto \xi(X)$ is first order definable, and hence computable by a fixed depth polynomial size circuit.

Then we set again as in (6.9) $\mathrm{opt}(\varphi)$ to be

$$\textbf{if } Ext_{k,U} \textbf{ then } \widetilde{\varphi} \textbf{ else } \varphi.$$

where $\widetilde{\varphi}$ is computed by circuit from Proposition 6.16. Formally, as we have defined it is Paragraph 6.1.4, $\text{opt}(\varphi)$ is evaluated by two circuits in parallel. Both of them begin with a check if $Ext_{k,U}$ holds and depending on it either the first or the second sets its output bit *output-here* and continues the computation. Note that $Ext_{k,U}$ is first order, so it can be checked by a constant depth circuit. (Now the reader knows, what we needed Lemma 6.12 for.)

The depth of this circuit we need can be estimated by Lemma 6.11 and Proposition 6.16 to be $2^{O(i(n))}$. The unoptimized φ is computed by an analogous circuit, but of depth n^l for some fixed l.

Now a computation similar to (6.10) shows the average complexity to be the depth of the shallower circuit, i.e., $2^{O(i(n))}$. This finishes case 2.

Next we consider case 3. It follows directly by the method used by the authors in [1]. Let us describe it briefly in our terminology. Again the optimized program $\text{opt}(\varphi)$ is

$$\textbf{if } Ext_{k,U} \textbf{ then } \widetilde{\varphi} \textbf{ else } \varphi.$$

This time when $Ext_{k,U}$ holds we have that φ is bounded, so we can use a circuit like on the Figure 2, but of constant depth. In the remaining cases it requires depth n^l, for some fixed l. And again computation similar to (6.10) shows that the average is equal to the dominating case, i.e., is $O(1)$.

The case 1. is a bit more difficult. We describe only the construction of the $O(1)$ depth circuit for the dominating case, the rest of the argument being analogous as in the previous items.

First we describe the intuitive idea: Again as before, the results of evaluation of queries depend with great probability on $\mathbf{A}_{\restriction U}$ only. Moreover, this dependence is rather simple: every query is union of some $\equiv_k$-classes of tuples. The $\equiv_k$-class of $\bar{a}$ on the other hand depends on the isomorphism type of $\langle \mathbf{A}_{\restriction U \cup \{a_1,\dots,a_k\}}, a_1, \dots, a_k \rangle$ only, as shown by Lemma 6.11.

Since there are at most $\sum_{j=0}^{k} 2^{\text{cp}'_\sigma(i(n)+j)}(i(n)+j)^k = n^{O(1)}$ different possible structures of this kind (recall that the encoding passed to the circuit is ordered, so we have to count ordered structures), we can encode *all of them* in the circuit together with a table, which of them are to be included in the output, depending on the real $\mathbf{A}_{\restriction U}$ included in the input. Then the computation of the circuit is essentially reduced to a table look-up.

Formally, every query φ in LFP^k on all structures in $\overline{\Sigma}^i(n)$ satisfying k, U-extension axioms is determined by a set $\overline{T}_\varphi$ of structures over $\sigma' \cup \{c_1, \dots, c_k\}$, which satisfy the following restrictions:

1. Cardinality of structures can be $i,\ i+1, \dots, i+k$.
2. U has precisely i elements $\{0, \dots, i-1\}$.
3. If there are elements not in U, they must be values of some of the constants $c_1, \dots, c_k$.

The query is then computed as follows:

$$\varphi(\mathbf{A}) = \{\bar{a} \in A^k \ : \ \exists \mathbf{B} \in \overline{T}_\varphi \ \langle \mathbf{A}_{\restriction U \cup \{a_1,\dots,a_k\}}, a_1, \dots, a_k \rangle \cong \mathbf{B}\}. \tag{6.11}$$

It is easily seen from this presentation, that if we can perform in parallel the tests

$$\langle \mathbf{A}_{\restriction U \cup \{a_1,\dots,a_k\}}, a_1, \dots, a_k \rangle \cong ?\mathbf{B}$$

in AC^0, then we will be done, since each of them just gives us a single tuple to be included in the output. And we can do it by Proposition 6.15. Note that, since $i \leq \sqrt[s]{\log n}$, the number of structures in $\overline{T}_\varphi$ cannot exceed

$$\sum_{j=0}^{k} 2^{\mathrm{cp}'_\sigma(i+j)}(i+j)^k \leq 2^{d \log n} = n^d$$

for some constant d, and this guarantees that we combine polynomially many polynomial size circuits. Precisely the same kind of isomorphism test is then necessary to choose the right tuples to be included in the output, depending on the table encoded in the circuit. $\square$

There are two comments concerning the two just proven theorems.

First is the question of uniformity of circuits we have constructed. It is easy to see, that the only problem is to construct a structure in $\overline{\Sigma}^i(n)$ satisfying the conjunction $Ext_{k,U}$ of all k, U-extension axioms. One such structure suffices, since extension axioms do not depend of the isomorphism type of $\mathbf{A}_{\restriction U}$. Then, by putting in all relevant $\mathbf{A}_{\restriction U}$'s and evaluating the query to be computed by the circuit one constructs the table to be encoded in it. The rest of the construction of the circuit is easy, except that a little care is needed to avoid problems when $i(n)$ is difficult to compute.

It is then a corollary of the proof of Theorem 6.23 that the circuits can be constructed in $\mathrm{DSPACE}(\max(\log n, 2^{O(i(n))}))$. For $i(n)/\log\log n \to 0$ this gives us a deterministic logarithmic space algorithm.

A substantially more difficult deterministic construction can be used in this proof, instead of the brute-force search we have used, and then it yields as a corollary, that for all functions i we have considered the circuits can be constructed in PTIME.

It is however relatively easy to see, that they can be constructed by a randomized algorithm, which works in average polynomial time. Indeed, note that the brute force algorithm we have just mentioned requires time at most 2^{n^l} for some fixed l. So let the randomized algorithm be as follows: choose at random n^l structures $\mathbf{A} \in \overline{\Sigma}^{1/2 \log n}(n)$; if any of them satisfies $Ext_{k,U}$, then it can be used to construct the circuit, as described above. This fragment of the algorithm runs in polynomial time, say at most n^m. Indeed, we have to consider polynomially many $\mathbf{A}_{\restriction U}$'s and for each of them we must perform a polynomial time computation. If this fails, we perform the brute force algorithm. As Lemma 6.9 shows, with probability at least $1 - 1/n$ a single randomly chosen structure will satisfy $Ext_{k,U}$. Therefore the probability that in n^l attempts we always get wrong structures is at most $n^{-n^l} = 2^{-n^l \log n}$. Now the average running time is at most $n^m \cdot (1 - 2^{-n^l \log n}) + 2^{-n^l \log n} \cdot 2^{n^l} = n^m + o(1)$.

Note that this allows us to run exactly once an average polynomial time algorithm to get a parallel device which will then give us an average parallel constant time computation of our query over all exponentially many n element structures in $\overline{\Sigma}^i(n)$. The naive evaluation requires polynomial time for every structure separately, and split $2^{\Omega(i)}$ parallel time.

The second comment is that of course nothing prevents us from extending the above theorems to functions $i(n)$ satisfying $\limsup_{n\to\infty} i(n)/\log n < 1$ (compare Lemma 6.9), but for such functions, missing in the present formulation, the complexity bounds we get do not differ from the trivial, pessimistic ones.

6.5 Conclusion

As we have seen in the proof of Theorem 6.24, in order to get optimization beyond split we have made simultaneous use of several facts:

First that $C(\mathrm{FO}^k[\mathbf{A}])$ is much (exponentially) smaller than $\pi_k(\mathbf{A})$ used in split and still contains the same information. This allowed us to reduce the problem to only polynomially many cases for each fixed n.

Second that a marker of each case was easily seen in the structures: it was the isomorphism type of $\mathbf{A}_{\restriction U}$.

Finally Boolean circuits allowed in AC^0 have a property not possessed by Turing machines. They can handle up to a polynomial number of "exceptions", i.e., inputs treated in a fixed way, independent of the main algorithm. This is, in fact, hidden in their nonuniformity. And we have made a really massive use of this, since our circuit algorithm is made entirely from exceptions. Turing machines in contrary can handle only a fixed number of such "exceptions".

It seems intriguing if this method works in other situations. If the value $C(L[\mathbf{A}])$ is at most logarithmic in $|A|$, then always among n-element structures there are only polynomially many possible $L[\mathbf{A}]$. Here we see the first problem: it is not always the case that the converse of Lemma 5.7 holds, and thus a low value of $I_L(\mathbf{A})$ is not yet a good news for query optimizers. But even if it does, there must be an easily seen marker of the case in structures, and finally a simple dependence between Boolean queries, to which KEP refers, and non-Boolean ones. (If we are interested in optimization of Boolean queries only, then the last point is unnecessary.)

The following figure represents all the complexities we have obtained together.

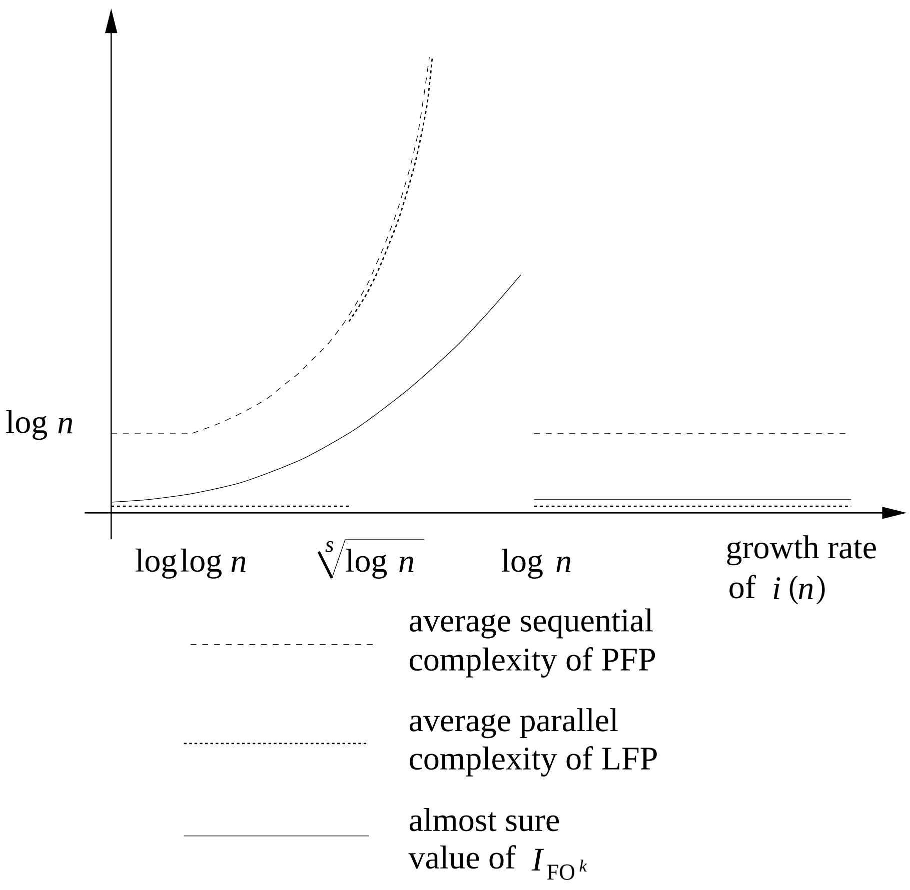

Figure 3: Overview of the results for classes $\overline{\Sigma}^i$.

7 Summary

In this paper we have introduced the Kolmogorov expressive power of Boolean query languages. It measures the expressiveness of the language in each particular database separately, expressing how much of the information stored in it can be retrieved by all the queries in the language.

It is an important property of the pair (database, query language). The reasons for it are as follows:

- If the Kolmogorov expressive power is low, then the user, most probably in contrary to his intentions, is not able to make use of all the data he has collected. In extremal cases almost all the collected information is inaccessible.
- The notion we have introduced can be easily adopted for almost every reasonable database model other than the flat relational one we have discussed in this paper, e.g., object oriented, complex-valued, etc., and almost every language of total Boolean queries. It will work even when it is not quite clear how to apply other expressiveness measures, because the Kolmogorov expressive power is independent of the chosen encoding of structures.
- If the Kolmogorov expressive power is high, then the possibility of split optimization of fixpoint and partial fixpoint queries is limited. The optimistic bound is polynomial time/space in the quantity of retrievable information (for LFP/PFP, respectively), whereas we have shown examples in which it is exponential.
- If the Kolmogorov expressive power is very low, then there is a chance to apply optimization techniques giving substantially more speed-up than split. We have shown an example of this phenomenon.

Acknowledgements. This paper is dedicated to my son Michał. I finished the conference version of it just few days before his was born. I finish this final version just few weeks before his second birthday. Whom else could I dedicate it to?

I am very grateful to Matt Kaufmann for his kind agreement to present the proof of his theorem in my paper.

I wish to thank Andrzej Tarlecki, who asked me the question if the restricted Kolmogorov expressive power had anything to do with optimization of queries.

References

[1] S. Abiteboul, K. Compton and V. Vianu, 'Queries are easier than you thought (probably)', in: *Proc. ACM Symp. on Principles of Database Systems, 1992.*

[2] S. Abiteboul, J. Hull and V. Vianu, *'Foundations of Databases',* Addison-Wesley 1995.

[3] S. Abiteboul and A. Van Gelder, 'Optimizing active databases using the split technique', in: *Proc. ICDT'92,* Springer Lecture Notes in Computer Science 646.

[4] S. Abiteboul and V. Vianu, 'Datalog extensions for database queries and updates', *Journal of Computer and System Sciences,* **43**(1991), pp. 62–124.

[5] S. Abiteboul and V. Vianu, 'Generic computation and its complexity', in: *Proc. ACM SIGACT Symp. on the Theory of Computing,* 1991, pp. 209–219.

[6] J. Barwise, 'On Moschovakis closure ordinals', *Journal of Symbolic Logic* **42**(1977), pp. 292–296.

[7] A. Blass, Y. Gurevich and D. Kozen, 'A zero-one law for logic with a fixed-point operator', *Inform. and Control* **67**(1985), pp. 70–90.

[8] A. Chandra, 'Programming primitives for database languages', in: *Proc. ACM Symp. on Principles of Programming Languages 1982,* pp. 50–62.

[9] A. Chandra and D. Harel, 'Computable queries for relational data bases', *Journal of Computer and System Sciences* **21**(1980), pp. 156–178.

[10] A. Dawar, S. Lindell and S. Weinstein, 'Infinitary logic and inductive definability over finite structures', *Information and Computation*, **119**, 1995.

[11] R. Fagin, 'Probabilities on finite models', *Journal of Symbolic Logic* **41**(1976), pp. 50–58.

[12] N. Immerman, 'Upper and lower bounds for first-order expressibility', *Journal of Computer System Sciences* **25**(1982), pp. 76–98.

[13] N. Immerman, 'Expressibility as a complexity measure: Results and directions', *Technical Report* DCS-TR-538, Yale Universsity, New Haven, CT, USA, 1987.

[14] N. Immerman, 'Languages that capture complexity classes', *SIAM Journal of Computing* **16**(1987).

[15] M. Kaufmann, 'A counterexample to the 0-1 law for existential monadic second-order logic', *CLI Internal Note 32, Computational Logic Inc., Dec. 1987.* Available by WWW from `http://www.informatik.rwth-aachen.de/WWW-math/papers.html`

[16] В.В. Князев, 'Закон нуля и единицы для многосортной логики предикатов первого порядка', *Дискретная Математика* том 2 вып. 3(1990), 97–101. [V.V. Knyazev, 'Zero-one law for many sorted predicate calculus of first order', Diskretnaya Matematika vol. 2(3),(1990), pp. 97–101.]

[17] Ph. Kolaitis and M. Vardi, 'Infinitary logic and 0-1 laws', *Information and Computation* **98**(1992), pp. 258-294.

[18] M. Li and P. Vitányi, 'Kolmogorov complexity and its applications', in: J. van Leeuven (ed.) *Handbook of Theoretical Computer Science,* North Holland, Amsterdam, 1990.

[19] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*, first edition, Springer Verlag, New York, 1993.

[20] B. Poizat, 'Deux ou trois choses que je sais de L_n', *Journal of Symbolic Logic* **47**(1982), pp. 641–658.

[21] G. Pólya, 'Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen', *Acta Math.* **68**(1937), pp. 145–254.

[22] J. Tyszkiewicz, 'On the Kolmogorov expressive power of Boolean query languages', in: *G. Gottlob, M.Y. Vardi (eds.) Proc. ICDT'95,* Lecture Notes in Computer Science No. 893, Springer Verlag, pp. 97–110.

[23] M. Vardi, 'Complexity of relational query languages', *Proc. 14th Symposium on Theory of Computation* 1982, pp. 137–146.

[24] P. Vitányi, *e-mail of March 29, 1995.*

A The Proof of Kaufmann's Theorem

The proof is a slight adaptation of the original Kaufmann's proof given in [15]. The difference is that there the structures consisted of four binary relations. We do the same for one, and the price we pay is that the proof is a bit uglier. Matt Kaufmann is responsible for everything that is nice in this proof. The author of this paper is responsible for all omissions, inaccuracies, English mistakes, etc., that appear in this presentation.

We recall the formulation of the result:

> *There exists a monadic second order formula of the form $\exists X_1 \ldots \exists X_m \varphi(x,y)$, where $X_1, \ldots, X_m$ are all set variables of φ and x, y are free in φ, such that with $\mu^{\overline{\Sigma}}$-probability* 1
>
> $$\mathbf{A} \models \exists X_1 \ldots \exists X_m \left(\begin{array}{c} \forall x \ \varphi(x,x) \\ \forall xy \ \varphi(x,y) \wedge \varphi(y,x) \rightarrow x = y \\ \forall xyz \ \varphi(x,y) \wedge \varphi(y,z) \rightarrow \varphi(x,z) \end{array} \right).$$
>
> *In other words, in almost every $\mathbf{A} \in \overline{\Sigma}$ there are choices of subsets of A witnessing existential set quantifiers such that $\varphi(x,y)$ defines a linear ordering of A.*

For simplicity of presentation we assume that $\sigma = \{R\}$ and that R is binary. The generalization of the argument to arbitrary non-unary signatures causes no problems.

Let $S, T \subseteq A$. We say that $a \in S$ *codes in* T the subset $\{b \in T \ : \ R(a,b)\}$. We say that $S \subseteq A$ *codes distinct subsets of* T iff no two elements of S code the same subset of T. We say that S *codes the powerset of* T iff S codes distinct subsets of T and moreover every subset of T is coded by some element of S.

Lemma A.1. *Let $S \subseteq T \subseteq A$ be fixed and let $|T| \geq |S| \cdot 2^{|S|}$. Then some subset of T codes the powerset of S with probability tending to 1 as $|S|$ and $|T|$ tend to ∞.*

Proof. It suffices to show that every subset of S is coded by some element of T with probability tending to 1. The probability of failure doesn't exceed

$$2^{|S|} \cdot \left(1 - 1/2^{|S|}\right)^{|T|} \leq 2^{|S|} \cdot \left(1 - 1/2^{|S|}\right)^{|S| \cdot 2^{|S|}}.$$

The second factor above is asymptotic to $1/e^{|S|}$, so the limit of the whole is 0. □

Lemma A.2. *Let S code distinct subsets of T and let $\varphi(x,y)$ be a monadic second order formula defining a linear order on T. Then there is monadic second order definition of order on S.*

Proof. Take the lexicographic ordering with respect to φ. □

Lemma A.3. *Let $n > k^2 4^k$ and let $\mathbf{B} \in \Sigma(k)$. Then the $\mu_n^{\overline{\Sigma}}$-probability that in $\mathbf{A} \in \overline{\Sigma}(n)$ there is a k-element subset $V \subseteq A$ such that $\mathbf{A}_{|V}$ is isomorphic to $\mathbf{B}$ approaches 1 as $n, k \rightarrow \infty$.*

Proof. For simplicity of presentation we assume σ to contain one binary relation symbol.

We embed the some fixed representative of the isomorphism type of $\mathbf{B}$ with universe set $\{1, \ldots, k\}$ into $\mathbf{A}$ in the following way: we divide A into k pieces of $k4^k$ vertices each and one piece containing remaining vertices, if there are any left. At each stage $i < k$ we try to extend the embedding by mapping i to some element in the i-th piece of A. The probability that we fail doesn't exceed the sum over i of the probabilities that there is no element of the i-th piece with the appropriate relation to the $i-1$ already chosen elements. The i-th such probability is $(1 - 1/4^{i-1})^{k4^k}$. Hence the sum is bounded above by $k(1 - 1/4^k)^{k4^k}$, and this number tends to 0 as $k \rightarrow \infty$. □

Proof of the Theorem. Fix a structure $\mathbf{A} \in \overline{\Sigma}(n)$. Let k be the greatest number such that $n \geq 3 \cdot 2^k 2^{2^k}$. Divide A into three subsets A_1, A_2, A_3 of $2^k 2^{2^k}$ elements each and one piece containing remaining vertices, if there are any left.

By Lemma A.3 with probability approaching 1 there is a $P_1 \subseteq A_1$ with $|P_1| = k$ such that R restricted to P_1 is a linear ordering.

By Lemma A.1 with probability tending to 1 there are $P_2 \subseteq A_2$ and $P_3 \subseteq A_3$ such that P_2 codes the powerset of P_1 and P_3 codes the powerset of P_2. Note that $|P_2| = 2^k$ and $|P_3| = 2^{2^k}$.

Finally with probability tending to 1 the whole A codes distinct subsets of P_3. Indeed, the probability that some two elements of A code the same subsets of P_3 doesn't exceed

$$\binom{n}{2} 2^{2^{2^k}} \left(1/2^{2^{2^k}}\right)^2 < 9 \cdot \left(2^{k+1} 2^{2^{k+1}}\right)^2 / 2^{2^{2^k}},$$

and the last number is easily seen to tend to 0.

All the probabilities we have mentioned are mutually independent: for the first we used edges inside A_1, for the second from A_2 to A_1, for the third from A_3 to A_2, and finally for the fourth from A to A_3. All these sets of edges are disjoint, and the existence of different edges is decided independently. Therefore the probability of failure of the four steps doesn't exceed the sum of probabilities of failures, which tends to 0.

Now starting from the ordering defined by R on P_1 by three applications of Lemma A.2 we get an ordering of A. $\square$

B The Proof of Proposition 5.8

We recall the formulation of the proposition to be proven:

> *There is a constant k_0 and an infinitely many models $\mathbf{A}$ of signature consisting of one unary and one binary symbol, and such that for every $k \geq k_0$ and every sufficiently large $\mathbf{A} \in D$ the following simultaneously hold*

$$C(\mathrm{FO}^k[\mathbf{A}]) = \Omega(\log C(\mathbf{A})) \tag{B.1}$$

$$I_{\mathrm{FO}^k}(\mathbf{A}) = O(1). \tag{B.2}$$

Before we start the proof, we should mention, that it is by no means a great achievement. It is rather a simple encoding of Example 2.27 from the book [19], presented there for pure Kolmogorov complexity. The following it Theorem 2.20 and Corollary 2.6 in the first edition of this book contain however a misprint in the formulation (but not in the proof!), which suggests the example is actually impossible [24].

Proof. We fix the signature σ to consist of one binary relation symbol R. We consider the following class D of structures: each $\mathbf{A} \in D$ is a directed graph, which consists of two connected components of equal cardinality. In one of them (call it $\mathbf{A}_1$) it is an arbitrary directed graph, in the other (call it $\mathbf{A}_2$) the relation R is a linear ordering.

It is an immediate observation that

$$C(\mathbf{A} \mid \mathbf{A}_2) = O(1).$$

Fixing $k \geq 3$ we see, that for any $\mathbf{A} \in D$ the theory $\mathrm{FO}^k[\mathbf{A}]$ allows the reconstruction of $|A|$ by a constant length program, by counting elements in $\mathbf{A}_2$. On the other hand, apart of the value of n, this theory contains only $O(1)$ bits of information about $\mathbf{A}$, for almost every $\mathbf{A} \in D$. This follows essentially from Theorem 6.18. The only difference is that we have artificially made the cardinality of the structure expressible in the logic, by adding $\mathbf{A}_2$ to $\mathbf{A}_1$. The conclusion is that the following is true for almost every $\mathbf{A} \in D$ of cardinality n:

$$C(\mathrm{FO}^k[\mathbf{A}] \mid n) = C(n \mid \mathrm{FO}^k[\mathbf{A}]) = O(1),$$

and, in particular,

$$C(\mathrm{FO}^k[\mathbf{A}]) = C(n) + O(1). \tag{B.3}$$

Now for an arbitrary sufficiently large n we consider structures in D of cardinality $2n$, i.e., those in which $\mathbf{A}_2$ has cardinality n. By combination of Theorem 6.5 and Proposition 2.5 there is always at least one among them satisfying

$$C(\mathbf{A}_1 \mid n) \geq cp_\sigma(n) - \log n!.$$

It follows, that this particular structure $\mathbf{A}$ satisfies

$$C(\mathbf{A}) = C(\mathbf{A} \mid n) + O(1) = C(\mathbf{A}_1 \mid n) + O(1) = n^2 - \log n!, \tag{B.4}$$

since it suffices to give the lexicographic number of $\mathrm{enc}(\mathbf{A})$ among codes of structures in D as the description, and this number doesn't exceed $(1 + o(1))2^{\mathrm{cp}_\sigma(n)}/n!$. (Compare Item 1 in the proof of Theorem 6.17.)

Now for such $\mathbf{A} \in D$ we get

$$I_{\mathrm{FO}^k}(\mathbf{A}) = C(\mathbf{A}) - C(\mathbf{A} \mid \mathrm{FO}^k[\mathbf{A}]) = C(\mathbf{A}) - C(\mathbf{A} \mid n) = O(1). \tag{B.5}$$

Finally if we require n to be itself incompressible, i.e., that $C(n) \geq \log n$, we get the desired example: (B.1) follows from (B.3), (B.4) and incompressibility of n, while (B.2) follows from (B.5). $\square$