Tailoring recursion for complexity*

Erich Grädel
Lehrgebiet Math. Grundlagen der Informatik
RWTH Aachen
Ahornstr. 55
D-52074 Aachen
Germany
graedel@informatik.rwth-aachen.de

Yuri Gurevich†
EECS Department
University of Michigan
Ann Arbor
MI 48109-2122
USA
gurevich@umich.edu

August 1994

Abstract

We design functional algebras that characterize various complexity classes of global functions. For this purpose, classical schemata from recursion theory are tailored for capturing complexity. In particular we present a functional analogue of first-order logic and describe algebras of the functions computable in nondeterministic logarithmic space, deterministic and nondeterministic polynomial time, and for the functions computable by AC^1-circuits.

1 Introduction

The design and investigation of logical languages that capture the major complexity classes has been (and still is) one of the most important topics in finite model theory. The modern history starts with Fagin's theorem [8]: Let L be class of finite structures of some fixed signature which is closed under isomorphisms. Then L is in NP if and only if there exists an existential second-order sentence ψ such that L is precisely the class of finite models of ψ. But there were earlier relevant results, most noticably, the characterization of the regular languages by means of monadic second-order logic [4, 27]. Immerman and Vardi [18, 28] proved that, on ordered structures, the problems solvable in polynomial time are exactly those definable in least fixed-point logic. A similar result was obtained independently by Livchak [23]. Immerman [17]–[21] systematically studied the problem of capturing complexity classes by logical languages and came up with logical characterizations for most popular complexity classes. For instance, logarithmic space complexity classes are captured by various forms of transitive closure logics [19]. The most important results in this field are surveyed in [14, 21].

These logical characterizations of complexity are model theoretic and based either on fragments of second-order logic [8, 10] or on extensions of first-order logic by additional means to construct

*To appear in Journal of Symbolic Logic. A prliminary version has appeared in Proceedings of 21st International Colloquium on Automata, Languages and Programming ICALP 94, Lecture Notes in Computer Science Nr. 820, Springer (1994), 118–129.

†partially supported by NSF and ONR

new relations (such as generalized quantifiers or predicate transformers). They are mostly *logics of relations*. Function symbols may appear in the signature, and may be composed to form terms, but the rôle of terms is very limited: in most model-theoretic languages we can only construct formulae from terms, not vice versa (for an exception, see [11]).

However, there is a related, but different approach to capturing complexity by logic, which is functional in nature and draws more on recursion theory than model theory. Gurevich noticed that interpreting the classical calculus of primitive recursive functions (resp. recursive functions) over finite structures gives precisely the log-space (resp. polynomial time) computable global functions [12, 14]. A variant of the latter result was obtained earlier by Sazonov [25] in a different context. Goerdt [9] and Libo Lo [24] extended Gurevich's results to higher complexity classes such as PSPACE and EXPTIME, and Compton and Laflamme [7] constructed a functional algebra for NC^1 in the same spirit.

This paper develops the functional approach further. We design functional algebras that characterize various complexity classes. We obtain new characterizations of the functions computable within polynomial space, deterministic and nondeterministic polynomial time, nondeterministic log-space and the functions computable by AC^1-circuits. (Computability by AC^1-circuits can be seen as a reasonable definition for functions computable in parallel with polynomial hardware and logarithmic time). We also define an algebra that provides a functional analogue to first-order logic, using *maximum and minimum operators* instead of Boolean connectives or quantifiers. Our algebras are constructed in a rather uniform way, with schemata using the max and min operators. Thus, this paper can also be seen as a study of the expressive power of these operators.

The nondeterministic classes are the more challenging for the functional approach. A priori, it is not even clear what the function is computed by agiven nondeterministic algorithm. We have rediscovered a definition first used by Krentel [22]: The value of the function in question at a given input is the maximum over all outputs produced by the algorithm on that input. In particular we present an algebra that captures the functions computed in this way in nondeterministic polynomial time. As an example, we construct a function in our algebra that computes the clique number of a given graph. Our approach permits one to deal with NP optimization problems (which provided one of the main motivations to study the notion of NP-completeness in the first place) in a direct way: the cost of optimal solutions can be expressed without any reference to the related decision problem.

Remark. It should be noted that there exists another approach to characterizing complexity using functional algebras. In fact, one of the very first papers on polynomial-time computability, by Cobham [6], constructs an algebra of polynomial-time computable functions. The distinction between the two approaches is this: Cobham and his followers (see [3, 5] and references there) deal with algebras of number-theoretic functions over $\mathbb{N}$. One of their goals is to relate complexity theoretic statements to proof-theoretic assertions about weak systems of arithmetic. Our algebras instead consist of global functions (see section 2) over finite structures.

2 Global functions

A *signature* is a finite set of relation symbols and function symbols. Constants are considered as functions of arity 0.

Definition 2.1 A *global function* f of arity r and co-arity s on a class $\mathcal{F}(\sigma)$ of σ-structures associates with every $\mathfrak{A} \in \mathcal{F}(\sigma)$ a function $f^{\mathfrak{A}} : A^r \to A^s$, where A is the universe of $\mathfrak{A}$.

Let $\mathcal{O}$ be the class of *finite ordered structures*, i.e. structures whose universe is an initial segment $\mathbf{n} = \{0, \ldots, n-1\}$ of $\mathbb{N}$ and whose signature contains a binary relation $\leq$, interpreted as the canonical order relation on $\mathbf{n}$, a unary function S, interpreted as the usual successor function (subject to the convention that $S(n-1) = n-1$), and two constants 0 and e, interpreted as the first and the last element of the universe. Let $\mathcal{O}(\sigma)$ be the class of structures in $\mathcal{O}$ with signature σ.

Proviso. In this paper:

- Structures are finite ordered structures
- Every global function is a global function on some $\mathcal{O}(\sigma)$.

Remark. Usually it is required that a global function be *abstract*, i.e., invariant under isomorphisms. However, in this paper we deal with rigid structures only and thus do not need the abstractness property.

Functions computed by nondeterministic algorithms. While there is a standard notion of the set of inputs accepted by a nondeterministic algorithm, it is not clear what the *function* computed by a given nondeterministic algorithm should be, since different computations on the same input may produce different outputs. One possibility is to require that all successful computations on a given input produce the same output. But this creates the undecidable problem whether a given nondeterministic algorithm satisfies this condition. Here we adopt a different approach which seems to fit the spirit of nondeterminism better.

Definition 2.2 Given a nondeterministic Turing machine M we define $f_M(x)$ to be the maximal output of M on input x with respect to the lexicographic ordering of strings in the output alphabet of M. (As usual, to order strings of arbitrary length lexicographically, first order them by length and then lexicographically.) Of course, $f_M(x)$ is undefined if no computation of M produces any output.

Function classes defined in this way were first investigated by Krentel [22]. Note that in the case that M is an acceptor, i.e. produces only outputs 0 and 1, the function f_M is the characteristic function of the set accepted by M according to the usual definition. We are particularly interested in *global functions* computed by nondeterministic algorithms.

Definition 2.3 For every signature σ, F-NLOG(σ) denotes the class of all global functions on $\mathcal{O}(\sigma)$ that are computable in nondeterministic logspace in the following sense: There is a nondeterministic algorithm M, which given (a suitable encoding of) a finite ordered structure $\mathfrak{A}$ and a tuple a of elements of $\mathfrak{A}$, computes the value $f^{\mathfrak{A}}(a)$ using workspace at most $O(\log n)$ where n is the cardinality of $\mathfrak{A}$. Let F-NLOG $= \bigcup_\sigma$ F-NLOG(σ).

Remark. The notion of F-NLOG would not change if we required all successful computations to compute the same value. Indeed, let M be a nondeterministic machine computing f with logarithmic space according to Definition 2.2. Then the set

$$L = \{(\mathfrak{A}, a, b) \mid f^{\mathfrak{A}}(a) \geq b\}$$

is in NLOGSPACE, and by the result of Immerman and Szelepcsényi [20, 26], so is its complement. Consider the algorithm M', which, given $(\mathfrak{A}, a)$, guesses b and then decides whether $(\mathfrak{A}, a, b) \in L$ and $(\mathfrak{A}, a, b+1) \notin L$. If yes, then M' produces output b, otherwise M' produces no output. Clearly, this algorithm uses nondeterministic logspace and each successful computation produces the correct output. Note that this argument crucially depends on the fact that NLOGSPACE is closed under complementation.

3 An algebra for first-order logic

We start with constructing a functional algebra with the expressive power of first-order logic.

Definition 3.1 Fix any signature σ, containing $S, \leq, 0, e$. Let x, y, z, t etc. stand for tuples of variables. The *initial global σ-functions* are:

(i) The functions named in σ and the characteristic functions of the predicates named in σ.

(ii) The characteristic function of equality: $\mathbf{eq}(x, y)$ equals **if** $x = y$ **then** 1 **else** 0.

(iii) The projections $\pi^k_{i_1,\ldots,i_j}$ of arity k and co-arity j, for $k, j \in \mathbb{N}$ and positive $i_1, \ldots, i_j \leq k$.

(iv) The selector function

$$\textbf{if-then}(x, y) = \begin{cases} y & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

The projection $\pi^k_{i_1,\ldots,i_j}$ maps the given tuple $(x_1, \ldots, x_k)$ to $(x_{i_1}, \ldots, x_{i_j})$. In particular, π^1_1 is the identity function.

Composition is defined in the usual way: if $h_1, \ldots h_m$ are functions of the same arity, with coarities summing up to the arity of g, then we can build the function $f = g(h_1, \ldots, h_m)$.

The *bounded maximum and minimum operators* assign to functions $f_1(x, y)$, $f_2(x, z)$ with the same coarity the functions

$$\begin{aligned} g(x, y, z) &= \min\{f_1(x, y), f_2(x, z)\} \\ h(x, y, z) &= \max\{f_1(x, y), f_2(x, z)\}. \end{aligned}$$

The *unbounded maximum and minimum operators* build from a function $f(x, y)$ new functions

$$\begin{aligned} g(x) &= \min_y[f(x, y)] \\ h(x) &= \max_y[f(x, y)]. \end{aligned}$$

Definition 3.2 The class of *first-order σ-functions* is the smallest class of global functions that contains the initial σ-functions and is closed under composition and the (bounded and unbounded) maximum and minimum operators.

Theorem 3.3 *The characteristic function of every first-order definable global relation is a first-order function. Moreover, a global function is first-order if and only if its graph is definable in first-order logic.*

PROOF. Let $\varphi(x)$ be a first-order formula; the characteristic function $F_\varphi(x)$ is built by induction on the complexity of φ. Clearly atomic formulae yield first-order functions; conjunction, disjunction and quantifiers are described by bounded and unbounded minimum and maximum operators. Negation is handled as follows:

$$F_{\neg\varphi}(x) := \mathbf{eq}(F_\varphi(x), 0).$$

To prove the second claim of the theorem, let $f(x)$ be a global function. First, we suppose that a first-order formula $\varphi(x, y)$ defines the graph of f. We have proved already that the characteristic function $F_\varphi(x, y)$ is first-order. To prove that f itself is first-order, we use the selector function:

$$f(x) = \max_y[\textbf{if-then}(F_\varphi(x, y), y)].$$

Conversely, suppose that f is first-order. We have to describe the equation $f(x) = y$ by a first-order formula $\varphi_f(x, y)$. For initial functions this is obvious. If f is built by composition of other functions, we can quantify over intermediate values: if $f(x) = g(h_1(x), h_2(x))$, then

$$\varphi_f(x, y) \equiv (\exists z_1)(\exists z_2)(\varphi_{h_1}(x, z_1) \wedge \varphi_{h_2}(x, z_2) \wedge \varphi_g(z_1, z_2, y)).$$

Finally the maximum and minimum operators are described in the obvious way. For instance, if $f(x) = \max_y g(x, y)$ then

$$\varphi_f(x, z) \equiv (\exists y)\varphi_g(x, y, z) \wedge (\forall y)(\forall z')(\varphi_g(x, y, z') \rightarrow z' \leq z).$$

■

Thus, on ordered structures, the class of first-order functions indeed provides a functional analogue to first order logic.

This new presentation of first-order logic suggests to extend the class of first-order global functions in many ways that are natural in the functional approach. Let us mention one such extension:

For every σ, let $I^+(\sigma)$ be the collection of initial global functions of signature σ together with

$$r(x) := e - x.$$

Let $F^+(\sigma)$ be the closure of $I^+(\sigma)$ under composition and maximum and minimum operators. In F^+ we have a generalized de Morgan identity

$$\max_y f(x, y) = e - \min_y\ [e - f(x, y)].$$

Note that F^+ contains functions that are not first-order. Indeed the function

$$\pi := \max_x[\mathbf{eq}(x, r(x))]$$

evaluates to 1 precisely on the structures of odd cardinality. Since this property is not expressible in first-order logic even on ordered structures [13], π cannot be first-order.

4 An algebra for F-NLOG

Gurevich considered in [12, 14] the closure of the initial functions under composition and the well-known schema for *primitive recursion*:

$$\begin{aligned} f(x,0) &= g(x) \\ f(x,t+1) &= h(x,t,f(x,t)). \end{aligned}$$

For given signature σ, we denote this class by $L(\sigma)$. Further, $L = \bigcup_\sigma L(\sigma)$. Although a slightly different class of initial functions is used in [12, 14], it is easy to see that semantically, the class L remains unchanged.

Gurevich proved

Theorem 4.1 *A global function is logspace computable if and only if it is in L.*

In this section we present an algebra describing the global functions computable in nondeterministic logspace.

Definition 4.2 A *specialization* of a global function F is obtained by identifying certain variables of F or by setting some of them to 0 or e. The *positive first-order operations* on a class of global functions are specializations and applications of (bounded and unbounded) max and min operators.

Definition 4.3 NL(σ) is the closure of the class of first-order functions of signature σ under positive first-order operations and the following recursion schema:

$$\begin{aligned} f(x,0) &= f_0(x) \\ f(x,t+1) &= \max_y[\textbf{if-then}(h(x,y), f(y,t))]. \end{aligned}$$

Further, $\mathrm{NL} = \bigcup_\sigma \mathrm{NL}(\sigma)$.

Example. The canonical NLOGSPACE-complete decision problem is REACHABILITY: Given a directed graph with two nodes a and b, decide whether there is a path from a to b. Let $edge(x,y)$ be the characteristic function of the edge predicate in the graph. Using the schema above, we build the function

$$\begin{aligned} d(x,0) &= \mathbf{eq}(x,b) \\ d(x,t+1) &= \max_y[\textbf{if-then}(edge(x,y), d(y,t))]. \end{aligned}$$

Obviously $d(x,t)$ evaluates to 1 if there is a path of length t from x to b, and to 0 otherwise. Thus REACHABILITY is expressed by the global function $reach = \max_t d(a,t)$.

Theorem 4.4 F-NLOG = NL.

PROOF. We first prove that every global function in NL is computable in nondeterministic logarithmic space. The proof proceeds by induction on the construction of functions $f \in \mathrm{NL}$. The only nontrivial case is when $f(x,t)$ is recursively defined from NL-functions $f_0(x)$ and $h(x,t)$ using the new recursion schema. By induction hypothesis, there is a nondeterministic algorithm

A_0 which given a structure $\mathfrak{A}$ and a tuple x computes $f_0^{\mathfrak{A}}(x)$ using space $O(\log n)$ where n is the cardinality of $\mathfrak{A}$. (Recall that, by the definition of nondeterministic computing of functions, if A_0 makes a wrong guess then it computes the same or a smaller number, but not a larger number.) Similarly, there is a nondeterministic algorithm B that computes $h^{\mathfrak{A}}(x,t)$ in space $O(\log n)$.

Given x and t, the desired algorithm A computes $f^{\mathfrak{A}}(x,t)$ as follows: If $t=0$ then A_0 is applied on x. If $t>0$ then A guesses y and simulates B to compute a number $z \leq h^{\mathfrak{A}}(x,y)$. If $z=0$, it outputs 0. Otherwise A makes the updates $x := y$ and $t := t-1$ and (recursively) applies A on this updated input. Under an arbitrary sequence of guesses A computes a number $\leq f^{\mathfrak{A}}(x,t)$ and A computes exactly $f^{\mathfrak{A}}(x,t)$ under some sequence of guesses.

Conversely, assume that f is a global function that is computed (according to Definition 2.3) by a nondeterministic Turing machine M with space $O(\log n)$. As input, M receives a pair $(\mathfrak{A}, a)$ where $\mathfrak{A}$ is an ordered structure over universe $\mathbf{n}$ (for some $n \in \mathbb{N}$), and a a tuple of elements of $\mathfrak{A}$. Without loss of generality, we may assume that every computation of M ends after precisely $n^k - 1$ steps, for some $k \in \mathbb{N}$.

A configuration in a computation of M on input $(\mathfrak{A}, a)$ reflects the input, the control state of M, the content of the work tapes, the position of the heads on the input and the work tapes and the content of the output tape. We call the collection of all these data, with the exception of the structure $\mathfrak{A}$, a *reduced configuration* of M. Clearly, every reduced configuration of M on $(\mathfrak{A}, a)$ can be represented by a word of length $O(\log n)$. Note that we define both the tuple a and the content of the output tape to be parts of the reduced configurations. We can do this, because M computes a global function; thus the output of M is (an encoding of) a tuple of elements of $\mathbf{n}$ and thus has logarithmic length. Given that reduced configurations of M have logarithmic length, we can represent them by tuples $c = c_1, \ldots, c_r$ of fixed length (where $c_i \in \mathbf{n}$). In particular, the initial reduced configuration on $(\mathfrak{A}, a)$ is represented by the tuple $a\mathbf{0} = (a, 0, \ldots, 0)$. Furthermore, there are standard techniques to write down global functions $N, V \in \mathrm{NL}$ satisfying the following conditions: For every $\mathfrak{A}$

- N is the characteristic function of the next move relation of M. More precisely $N^{\mathfrak{A}}(c,d) = 1$ if c, d are reduced configurations and M can reach d from c in one step; otherwise $N^{\mathfrak{A}}(c,d) = 0$.
- $V^{\mathfrak{A}}(c)$ is the value on the output tape of the reduced configuration represented by c.

Now let F be the global function of arity $r+k$, defined by the schema

$$\begin{aligned} F(c, 0) &= V(c) \\ F(c, t+1) &= \max_d [\textbf{if-then}(N(c,d), F(d,t))] \end{aligned}$$

If c encodes a reduced configuration, then $F^{\mathfrak{A}}(c,t)$ is the maximal content of the output tape of all configurations that are reachable by M in t steps from c. Thus, $f(x) = F(x\mathbf{0}, \mathbf{e})$. (Here and in the sequel, $\mathbf{e}$ denotes a tuple $e, \ldots, e$.) ■

5 Substitutions and an algebra for PTIME

As we have mentioned above, by interpreting the classical calculus of recursive functions over ordered finite structures, we obtain precisely the polynomial-time computable global functions. Here, we introduce a new schema that also gives the PTIME functions and which generalizes in a nice way to capture the NPTIME functions. The crucial feature is that some of the functions are not necessarily static, but may evolve. We first consider a general notion of *substitution* of global functions.

Definition 5.1 Let G be a global function of signature $\sigma \cup \{h\}$, arity r' and coarity s' with h of arity r and coarity s. Let F be a global function of signature τ, arity $r+k$ and coarity s. Then

$$[G \text{ where } h(x) = F(x,y)]$$

is the global function of signature $\sigma \cup \tau$, arity $r'+k$ and coarity s', defined on a given $(\sigma\cup\tau)$-structure $\mathfrak{A}$ by:

$$[G \text{ where } h(x) = F(x,y)]^{\mathfrak{A}}(z,y) = G^{\mathfrak{B}(y)}(z)$$

where $\mathfrak{B}(y)$ is a structure of signature $\sigma \cup \{h\}$ which has the same universe as $\mathfrak{A}$ and the same interpretation for symbols in $\sigma - \{h\}$ but with h interpreted by $h^{\mathfrak{B}(y)}(x) = F^{\mathfrak{A}}(x,y)$.

In this paper we restrict attention to the case when $\tau \subseteq \sigma \cup \{h\}$. Furthermore, for describing the polynomial-time computable functions it suffices to consider the case where the tuple y is empty, so that the function obtained by substitution does not depend on more variables than the original one. In such a case we drop x and write

$$[G \text{ where } h = F].$$

Example. Let g be a global function on directed graphs, and let $edge(x,y) = \max\{arc(x,y), arc(y,x)\}$. Then the global function $[\tilde{g} = g \text{ where } arc = edge]$ assigns to a directed graph $G = (\mathbf{n}, arc)$ the value of g at the corresponding undirected graph $G' = (\mathbf{n}, edge)$.

We now make use of the fact, that substitutions $G \longmapsto [G \text{ where } h = F]$ can be iterated if the substituting function F does again depend on h.

Definition 5.2 Let FP be the closure of the first-order functions under composition, substitution and the following recursion schema:

$$\begin{aligned} F(x,0) &= F_0(x) \\ F(x,t+1) &= [F \text{ where } h = G](x,t). \end{aligned}$$

where F_0, G are given global functions of signature σ and h is one of the function symbols in σ. The signature of the new function is also σ. Note that the function h evolves: To compute $F(x,t+1)$ we must evaluate $F(x,t)$ with an updated value for h.

Example: The number of connected components of a graph. We construct a function that assigns to a given undirected graph the number of its connected components. The graphs are given by the characteristic function of the edge predicate. To do this we use the new recursion schema to build a function F of signature $\{edge, h\}$. By substitution we then obtain the function $[F \text{ where } h = \pi^1_1]$ (of signature $\{edge\}$) whose value on large enough arguments t will be the number of connected components.

We suppose that we have already constructed (a notation for) the 0-ary function F_0 whose value (for any given h) is the cardinality of the image of h, i.e. $F_0 = |\{x : \exists y(h(y) = x)\}|$. This function is in $L(\{h\})$, i.e. it is definable by primitive recursion over the signature $\{h\}$. The function h is updated using the first-order function

$$G(z) = \max\{h(z), \max_y[\textbf{if-then}(edge(y,z), h(y))]\}.$$

The update $h = G$ defines $h(z)$ to be the maximal previous value of h on z itself and all its neighbours. If we start with $h = \pi_1^1$ (the identity function) and iterate this process, then the value of $h(z)$ will finally be the maximal node in the connected component of z. The function

$$\begin{aligned} F(0) &= F_0 \\ F(t+1) &= [F \text{ where } h = G](t) \end{aligned}$$

has the property that $[F \text{ where } h = \pi_1^1](t)$ for large enough t is the number of connected components of the graph.

Theorem 5.3 *A global function is* PTIME*-computable if and only if it is in FP.*

PROOF. We first show that every function in FP can be evaluated in polynomial time. It is clear that this holds for first-order functions and that the polynomial-time computable functions are closed under substitution and composition. Suppose that F is given by the recursion schema

$$\begin{aligned} F(x,0) &= F_0(x) \\ F(x,t+1) &= [F \text{ where } h = G](x,t). \end{aligned}$$

from functions G and F_0 which are already known to be polynomial-time computable.

To compute, for given $\mathfrak{A}$, a and t, the value of $F^{\mathfrak{A}}(a,t)$, we update h repeatedly using the function G and obtain a sequence $\mathfrak{A}_t, \mathfrak{A}_{t-1}, \dots, \mathfrak{A}_0$ where $\mathfrak{A}_t = \mathfrak{A}$ and $\mathfrak{A}_{i-1} = [\mathfrak{A}_i \text{ where } h = G]$. Note that $F^{\mathfrak{A}}(a,t) = F_0^{\mathfrak{A}_0}(a)$ which can be evaluated in polynomial-time.

To prove the converse, let M be a polynomial time Turing machine which operates on inputs $(\mathfrak{A}, a)$ where $\mathfrak{A}$ is a finite ordered structure of signature σ and a is an i-tuple of elements of $\mathfrak{A}$, and which produces as output a j-tuple $b = f^{\mathfrak{A}}(a)$ of elements in $\mathfrak{A}$. Here i and j are fixed once and for all. To simplify notation, we will omit a, i.e. we asume that f has arity 0 (which is no loss of generality since we can incorporate the elements of a as constants into $\mathfrak{A}$). It is convenient to assume that M has an input tape holding a suitable encoding of $(\mathfrak{A}, a)$, one work tape and a special output tape of length $O(\log n)$. A configuration of M can be described by a function $C(z)$ which provides the necessary information about the control state, the head positions, content of the work tape at cell z, and the current content of the output tape. Furthermore, we assume that after reaching a final configuration, M does not halt but becomes idle executing an instruction that does not change the configuration.

It is a matter of routine to define first-order functions C_0, N and V with the following properties:

- C_0 has signature σ, and $C_0^{\mathfrak{A}}$ describes the input configuration of M on $\mathfrak{A}$.
- N has signature $\sigma \cup \{C\}$. If C describes a configuration of M on input $\mathfrak{A}$, then $N^{\mathfrak{A},C}$ describes the successor configuration of C.
- V has signature $\sigma \cup \{C\}$. If C describes a configuration of M on input $\mathfrak{A}$, then $V^{\mathfrak{A},C}$ is the value on the output tape at configuration C (so V is just a simple projection).

We now define a global function $F(t)$ of signature $\sigma \cup \{C\}$ by

$$\begin{aligned} F(0) &= V \\ F(t+1) &= [F \text{ where } C = N](t). \end{aligned}$$

We claim that, for any configuration given by C, $F(t)$ describes the content of the output tape of M after t steps, starting at C.

For $t = 0$, this is clear. The value of $F(t+1)$ is $F(t)$ with C updated to the successor configuration of the given C, so the claim follows by induction.

This implies that $[F \text{ where } C = C_0](e)$ describes the output of M at the end of the computation on input $\mathfrak{A}$. ■

An alternative schema for the polynomial time computable functions. As we showed in the proof of Theorem 5.3, the recursion schema

$$\begin{aligned} F(x, 0) &= F_0(x) \\ F(x, t+1) &= [F \text{ where } h = G](x, t) \end{aligned}$$

can be efficiently evaluated in a top-down fashion: To compute $F(x,t)$, the function h is modified t times using G, and then $F_0(x)$ is evaluated with this updated value for h. There is an alternative schema which lends itself easily to a bottom-up evaluation:

Given functions f_0 of signature σ and g of signature $\sigma \cup \{h\}$, a new function f of signature σ is defined by

$$\begin{aligned} f(x, 0) &= f_0(x) \\ f(x, t+1) &= [g \text{ where } h(z) = f(z, t)](x). \end{aligned}$$

To evaluate, for a given structure $\mathfrak{A}$, the function $f^{\mathfrak{A}}(a, t+1)$, we have to compute the function $f_t(z) = f^{\mathfrak{A}}(z, t)$ and evaluate $g(a)$ at the $\sigma \cup \{h\}$-structure $(\mathfrak{A}, f_t)$ (where f_t interprets h, of course). Thus, a natural way to evaluate f is by computing the functions $f_0, f_1, \ldots, f_t$ in a bottom-up fashion.

It is easy to see that the closure of the first-order functions under composition, substitution and the alternative schema also coincides with the polynomial-time computable global functions.

Indeed, it is obvious that also the alternative schema can be evaluated in polynomial time. For the converse we assume that F is a global function, of signature σ which is computable in polynomial time by a Turing machine M; we can use precisely the same functions C_0, N and V as in the proof of Theorem 5.3. Recall that N and V have signature $\sigma \cup \{C\}$. With the alternative schema we can then build a function $f(x,t)$ of signature σ as follows:

$$\begin{aligned} f(x, 0) &= C_0(x) \\ f(x, t+1) &= [N \text{ where } C(z) = f(z, t)](x). \end{aligned}$$

At stage t the function $C(z) = f_t(z) = f(z,t)$ describes the configuration of M at time t in the same way as in the proof above. Finally, recall that V gives the content of the output tape at a given configuration C. Thus

$$[V \text{ where } C = f(z, e)]$$

is the desired description of F.

Exercise. Give direct translations from functions defined by the original schema to the alternative schema and vice versa.

6 An algebra for nondeterministic polynomial time

We now generalize the approach of the previous section to obtain a similar description of the global functions computable in nondeterministic polynomial time. First, it should be noted that the class of these functions is not closed under arbitrary compositions or substitutions, unless NP is closed under complementation. Indeed, if F is the characteristic function of a set $L \in$ NP, then $[\mathbf{eq}(f,0)$ where $f = F]$ or, equivalently $\mathbf{eq}(F,0)$ is the characteristic function of the complement of L.

However, substituting $[G$ where $f = F]$ will not take us out of nondeterministic polynomial time if F is, say, first-order. Given a global function G, we say that G' *is obtained by a substitution of a first-order function* if

$$G' = [G \text{ where } f = F]$$

where f is a function symbol in the signature of G and F is first-order.

Definition 6.1 Consider the global functions definable by the schema

$$\begin{aligned} F(x,0) &= F_0(x) \\ F(x,t+1) &= \max_y [F \text{ where } h(z) = G(z,y)](x,t) \end{aligned}$$

where F_0 and G are first-order functions and h is one of the function symbols in the signature of F_0 and the signature of G. We define FNP to be the closure of these functions under substitutions of first-order functions and under positive first-order operations.

The semantics of the function $F(x,t)$ at a given σ-structure $\mathfrak{A}$ can be explained by induction on t. It is obvious what $F^{\mathfrak{A}}(x,0)$ is. For each y in $\mathfrak{A}$, the substitution $h(z) = G(z,y)$ defines a function $h_y(z) = G^{\mathfrak{A}}(z,y)$ of z. The value of the function

$$F'(y,x,t) = [F \text{ where } h = h_y](x,t)$$

at structure $\mathfrak{A}$ is the value of $F(x,t)$ at the structure $\mathfrak{A}_y$ obtained from $\mathfrak{A}$ by replaing h with h_y. This defines $F^{\mathfrak{A}}(x,t+1)$. So let us understand $F(x,3)$ for example. It is the maximum over y of the function $F'(y,x,2)$, that is the maximum over y of $F(x,2)$ at all structures $\mathfrak{A}_y$. And $F(x,2)$ at $\mathfrak{A}_y$ is the maximum over y' of $F(x,1)$ at all structures $\mathfrak{A}_{y,y'} = (\mathfrak{A}_y)_{y'}$. And $F(x,1)$ at $\mathfrak{A}_{y,y'}$ is the maximum over y'' of $F(x,0)$ at all structures $\mathfrak{A}_{y,y',y''}$. In general, $F(x,t)$ at structure $\mathfrak{A}$ is the maximum over sequences $Y = (y_1,..,y_t)$ of $F(x,0)$ at all structures $\mathfrak{A}_Y$. Thus to evaluate $F(x,t)$ at structure $\mathfrak{A}$, one can guess a sequence $Y = (y_1,..,y_t)$ and evaluate $F^{\mathfrak{A}_Y}(x,0)$. Since the class of global functions computable in nondeterministic polynomial time is obviously closed under positive first-order operations and under substitutions of first-order functions, we obtain

Theorem 6.2 *Every global function in* FNP *can be computed in nondeterministic polynomial time (in the sense of Definition 2.2).*

Example: The clique number of a graph. In this example, a graph is always an undirected graph with vertex set $\mathbf{n} = \{0,\ldots,n-1\}$. The clique number of a graph G, denoted $\omega(G)$, is the size of the largest clique in G. The related decision problem — to decide whether there exists a clique of size k in G — is NP-complete. We show that the global function ω on graphs is in FNP. For simplicity, we restrict attention to incomplete graphs where $\omega(G) < |G| = n$.

Let us be a little bit more precise. By default, the domain of a global function of signature σ is the class of all finite σ-structures. In this case, we consider smaller domains. For every global function f below, the signature σ_f of f includes the binary function *edge* — to be understood as the characteristic function of the edge predicate — and the domain of f consists of finite σ_f-structures satisfying the following two restrictions:

1. For all x and y, $edge(x, y) = edge(y, x)$, and $edge(x, x) = 0$.
2. There are distinct x and y such that $edge(x, y) = 0$.

Let $c(x)$ be a unary function on the vertex set of a graph $G = (\mathbf{n}, edge)$. We say that c *defines an ordered clique* if there exists a clique $C = \{x_1, \ldots, x_r\}$ in G such that $c(x_j) = j$ for $j = 1, \ldots, r$ and $c(y) = 0$ for all $y \notin C$. We also permit the case where $r = 0$, i.e. also the function $c = 0$ defines an ordered clique.

We first describe a global first-order function $g(y, z)$ of signature $\{c, edge\}$ which will be used to update global functions $c(z)$ that describe ordered cliques. Suppose that c defines the clique C in G, and that y is a vertex of G. Then the function c_y defined by $c_y(z) := g(y, z)$ will define a new clique

$$C_y = \begin{cases} C \cup \{y\} & \text{if this happens to be a clique} \\ C & \text{otherwise.} \end{cases}$$

Moreover, $c_y(z) = c(z)$ for all z, with the following (possible) exception: if $y \in C_y - C$ then $c_y(y) = |C_y|$. The update function $g(y, z)$ that gives this, is:

$$\begin{aligned} g(y, z) \quad = \quad & \textbf{if } [y = z \wedge c(y) = 0 \wedge \forall x(c(x) > 0 \rightarrow edge(x, y) = 1)] \\ & \quad \textbf{then } \max_x c(x) + 1 \textbf{ else } c(z). \end{aligned}$$

It is not difficult to see that this is a first-order function: Let $f(y, z)$ be the characteristic function of the first-order formula in the brackets. Then

$$g(y, z) = \max\{c(z), \textbf{if-then}(f(y, z), \max_x c(x) + 1)\}.$$

Next, we define a global function F of signature $\{c, edge\}$ by the schema

$$\begin{aligned} F(0) &= \max_x c(x) \\ F(t+1) &= \max_y [F \text{ where } c(z) = g(y, z)](t) \end{aligned}$$

Finally, we define the 0-ary global function

$$\text{maxclique} = [F \text{ where } c = 0](e)$$

of signature $\{edge\}$.

Proposition 6.3 *For every graph G with $\omega(G) < |G|$*

$$\text{maxclique}^G = \omega(G).$$

This proposition follows immediately from the following lemma that describes the semantic of F on structures (G, c) where G is a graph and c is a unary function defining an ordered clique on G:

Lemma 6.4 *If c defines an ordered clique C in G, then*

$$F^{(G,c)}(t) = \max\{|C'| : C' \textit{ is a clique in } G, C \subseteq C' \textit{ and } |C'| \leq |C| + t\}.$$

PROOF. We proceed by induction on t. For $t = 0$ the statement is obvious.

By the definition of F we obtain

$$F^{(G,c)}(t+1) = \max_y F^{(G,c_y)}(t)$$

where $c_y(z) = g^{(G,c)}(y, z)$. Thus, by the construction of $g(y, z)$, the function c_y also defines an ordered clique C_y on G, namely $C_y = C \cup \{y\}$, if this happens to be a clique, or $C_y = C$ otherwise. We can therefore apply the induction hypothesis and obtain

$$F^{(G,c_y)}(t) = \max\{|C'| : C' \text{ is a clique in } G, C_y \subseteq C' \text{ and } |C'| \leq |C_y| + t\}.$$

But this immediately implies

$$F^{(G,c)}(t+1) = \max\{|C'| : C' \text{ is a clique in } G, C \subset C' \text{ and } |C'| \leq |C| + t + 1\}.$$

∎

For $t = |G| - 1$ and $c = 0$, the lemma implies the proposition.

With the same method, one can define functions in FNP that describe the length of the longest path, of the longest induced path, of the longest cycle in a graph, and so on.

Theorem 6.5 *A global function is computable in nondeterministic polynomial time if and only if it is in FNP.*

PROOF. One direction has already been established. The proof of the other direction is analogous to the proof of Theorem 5.3. The only difference is that the function N, which updates the configuration function C, depends on an additional parameter y, the nondeterministic choice: If $C^{\mathfrak{A}}(\,\text{-}\,)$ describes a configuration of the nondeterministic Turing machine M on input $\mathfrak{A}$, and y is one of the possible choices of M, then $N^{\mathfrak{A},C}(\,\text{-}\,, y)$ describes the corresponding next configuration. The functions C_0 and V are defined precisely as in the proof of Theorem 5.3. The value computed by M after $t+1$ steps from configuration C is the maximum over the values computed in t steps from one of the possible successor configurations. This is described by the function F defined by the schema

$$\begin{aligned} F(0) &= V \\ F(t+1) &= \max_y [F \text{ where } C(z) = N(z, y)](t). \end{aligned}$$

Again, the function computed by M is $[F \text{ where } C = C_0](\mathbf{e})$ which is obtained from F by a substitution of a first-order function and a specialization, and which is therefore in FNP. ∎

7 Fixed points

The idea of iterating function updates suggests a fixed point construction: Instead of performing the update a fixed number of times, we can do so until a stable situation is reached, starting from some default initial value.

Definition 7.1 Suppose F is a global function over σ where g is a function symbol in σ with the same arity and coarity as F. Then

$$\textbf{fixpoint}[g := F]$$

is a global function of signature σ whose semantic is defined by the following process. Set:

$$G^0 := g, \qquad G^{i+1} = [F \text{ where } g = G^i]$$

If, on a given structure $\mathfrak{A}$, this process converges, i.e. if there exists a number i such that $G^{i+1} = G^i$ then we define the semantic of $\textbf{fixpoint}[G := F]$ on $\mathfrak{A}$ to be the function G^i, otherwise we define $\textbf{fixpoint}[G := F]$ to be identically 0.

Note that convergence is not guaranteed: for instance, if $g = 0$ and $F(z) = \textbf{eq}(g(z), 0)$ then G^i is identically 0 for even i, and identically 1 for odd i.

In fact this fixpoint construction is analogous to *partial fixed point logic* which captures PSPACE on ordered structures (see [1, 28]).

So, if we denote the closure of the first-order functions under composition, substitution and under this fixpoint construction by FFP (for functional fixed points), then it is not surprising that we have

Theorem 7.2 *A global function is computable in* PSPACE *if and only if it is equivalent to a function in* FFP.

PROOF. It is clear that every function in FFP is computable with polynomial space. To prove the converse, we can use the same representation of Turing machine computations by functions C_0 (for the initial configuration), N (for the next-move update of configurations) and V (for the value on the output tape) as in the proof of Theorem 5.3. Without loss of generality we can assume that we have a Turing machine M computing f within polynomial space and which has the following additional property: every configuration c of M has a well-defined successor configuration which equals c if and only if c is a final configuration.

Then, for any function C representing a configuration, the function

$$F = \textbf{fixpoint}[C := N]$$

defines the final configuration reached eventually by M from C if it exists, otherwise $F = 0$. Thus, the function computed by M can be represented

$$[V \text{ where } C = [F \text{ where } C = C_0]].$$

■

We can ensure the convergence of the fixpoint construction by imposing a condition on the function updates that makes the fixpoint operator inflationary:

Definition 7.3 Suppose F is a global function over $\sigma \cup \{G\}$ where G has the same arity and coarity as F. Then

$$\mathbf{fixpoint}[G := \max\{F, G\}]$$

is a global function whose semantic is as in the previous definition. We denote the closure of the first-order function under this schema by IFFP (for inflationary functional fixed points).

IFFP is a functional analogue to the inflationary fixed point logic of Gurevich and Shelah [16]. On finite structures inflationary fixed point logic has the same expressive power as least fixed point logic [16], even without a built-in order. Therefore, it captures PTIME on ordered structures. It easy to see that this implies the following theorem.

Theorem 7.4 IFFP = FP *and contains therefore precisely the functions that are computable in* PTIME.

8 Functions computed by circuits

For our purpose it is convenient to consider a circuit of size at most n^k, where n is the number of input bits as an algebra $C = (\mathbf{n}, edge, and, or, neg, in, out)$ where the nodes of the circuits are represented by k-tuples over the universe $\mathbf{n}$, where the function $edge(x, y)$ has arity $2k$ and is the characteristic function of the edge predicate of the circuit (considered as a directed acyclic graph); the other functions have arity k and determine the type of a node.

It is useful to restate the uniformity condition in circuit complexity (see e.g. [2]) in algebraic terms. Note, that $L(\emptyset)$ is the algebra of primitive recursive, i.e. logspace-computable, global functions over the empty signature. Thus a function in $L(\emptyset)$ depends only on the cardinality of the input structure. We therefore can say, that a sequence $(C_n)_{n\in\mathbb{N}}$ is log-uniform if the global functions *edge, and, or, neg, in, out* that characterize $(C_n)_{n\in\mathbb{N}}$ belong to $L(\emptyset)$.

We are interested in circuit sequences that compute global functions. The following definition is taken from [7]:

Definition 8.1 Let σ be a signature and f a global function on $\mathcal{O}(\sigma)$ of arity r and coarity s. We say that a family $(C_n)_{n\in\mathbb{N}}$ of Boolean circuits *implements* f if the following conditions hold:

- The input nodes of each C_n specify a pair $(\mathfrak{A}, a)$ where $\mathfrak{A}$ is a σ-structure with universe $\mathbf{n}$ and a is an r-tuple of elements of $\mathfrak{A}$.
- C_n has $s\lceil\log n\rceil$ output nodes. The output computed by C_n on input $(\mathfrak{A}, a)$ is considered as the binary representation of an s-tuple of elements of $\mathfrak{A}$: the first $\log n$ output nodes code the first component, and so on. We denote this tuple by $f_n(\mathfrak{A}, a)$.
- For all n and all inputs $(\mathfrak{A}, a)$ for C_n,

  $$f_n(\mathfrak{A}, a) = f^{\mathfrak{A}}(a).$$

Definition 8.2 Given a signature σ, let $S(\sigma + L(\emptyset))$ be the closure under composition, maximum and minimum operators of the initial functions over σ and the functions defined by the schema for primitive recursion over the empty signature.

The algebra $S(\sigma + L(\emptyset))$ provides a characterization of the functions in AC^0. The following result is due to Gurevich and Lewis [15]:

Theorem 8.3 *A global function over σ can be implemented by a log-uniform sequence of constant-depth polynomial-size circuits if and only if it is in $S(\sigma + L(\emptyset))$.*

We will now define a recursion schema to build a more powerful algebra on top of $S(\sigma + L(\emptyset))$ that captures the global functions in AC^1, that is the functions computable by a log-uniform sequence of Boolean circuits of polynomial size and of depth $O(\log n)$ with unbounded fan-in. It is well-known that, without loss of generality, such circuits can be assumed to have the following properties. Negations occur only at the leafs, i.e., we have circuits of $\wedge$- and $\vee$-gates over inputs and negated inputs. Further, nodes of odd depth are or-nodes whose incoming arcs come from and-nodes and leafs, and nodes of positive even depth are and-nodes whose incoming arcs come from leafs and or-nodes.

Definition 8.4 Let $\mathbf{A}^1(\sigma)$ be the closure of $S(\sigma + L(\emptyset))$ under composition and the following schema: If $g \in \mathbf{A}^1(\sigma)$ is already defined, and h_1, h_2 belong to $S(\sigma + L(\emptyset))$, then we define the new function

$$\begin{aligned} f(x, 0) &= g(x) \\ f(x, 2t+1) &= \max_y h_1(x, y, f(y, t)) \\ f(x, 2t+2) &= \min_y h_2(x, y, f(y, t)). \end{aligned}$$

Theorem 8.5 *A global function over σ can be implemented by a log-uniform sequence of polynomial-size, log-depth circuits if and only if it belongs to $\mathbf{A}^1(\sigma)$.*

PROOF. Let $f \in \mathbf{A}^1(\sigma)$. Since functions in $S(\sigma + L(\emptyset))$ can be implemented by constant-depth polynomial-size circuits, we can suppose that f is defined by the recursion schema given above from the functions g which is computed by a circuit-family of depth $c \log n$, and from the functions h_1, h_2 computable by constant depth circuits.

Thus, $f(x, 2t+1)$ and $f(x, 2t+2)$ can be computed by log-uniform circuits of constant depth d using input-gates for the bits of $f(x, t)$. By induction, it then follows that the bits of $f(x, t)$ are computable by circuits of depths $c \log n + d\lfloor \log(t+1) \rfloor$. Thus f can be implemented by AC^1-circuits.

Conversely, suppose we have a global function that is implemented by a sequence $(C_n)_{n \in \mathbb{N}}$ of AC^1-circuits. Since the sequence is log-uniform, the global functions *edge, and, or, in, out* describing the circuit family are in $L(\emptyset)$.

The inputs for the circuits are encodings of structures $\mathfrak{B}$ such that every atom or negated atom (i.e. the diagram of the structure) is represented by an input node. We can choose this encoding to be first-order. Let $g(x)$ be the characteristic function of the (global) predicate expressing that x is an input node getting the value 1. This function is in $S(\sigma + L(\emptyset))$. It suffices to show that the global function $F(x)$, whose value $F^{\mathfrak{B}}(x)$ at a given input structure $\mathfrak{B}$ of cardinality n is the value computed by the circuit C_n at node x on input $\mathfrak{B}$, is in $\mathbf{A}^1(\sigma)$.

We construct a function $f(x, t)$ with the following properties: if x is a node of depth $d(x)$ then for some $t < 2^{d(x)}$, $f(x, t)$ is the value computed by the circuit at node x; for all other t, $f(x, t) = 0$. In particular, the value $F(x)$ computed at node x is $\max_t f(x, t)$.

Setting $f(x,0) = g(x)$ satisfies the required properties for the leafs. It remains to specify functions $h_1, h_2 \in S(\sigma \cup \{edge, and, or\})$ such that these properties translate, via the recursion schema, to nodes of higher depths. Let

$$h_1(x,y,z) = \textbf{if-then}(or(x), \textbf{if-then}(edge(y,x), z)).$$

Then, $\max_y h_1(x,y,f(y,t))$ evaluates to 1 if and only if x represents an or-node, and $f(y,t) = 1$ for at least one predecessor y of x. Thus $f(x, 2t+1) = \max_y h_1(x,y,f(y,t))$ satisfies the required condition (for x being an or-node). Similarly, let $h_2(x,y,z)$ be the characteristic function of the predicate

$$(and(x) = 1) \wedge (edge(x,y) = 1 \rightarrow z = 1)$$

which is obviously first-order. Then $f(x, 2t+2) = \min_y h_2(x,y,f(y,t))$ satisfies the required properties also for and-nodes. ∎

References

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

[2] J. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity*, vol I and II, Springer Verlag 1988/1990.

[3] S. Bellantoni and S. Cook, *A New Recursion-Theoretic Characterization of the Polytime Functions*, Proceedings of 24th ACM Symposium on the Theory of Computing (1992), 283–293.

[4] J. Büchi, *Weak second-order arithmetic and finite automata*, Z. Math. Logik Grundlagen Math. **6** (1960), 66–92.

[5] S. Buss, *Bounded Arithmetic*, Bibliopolis, Napels 1986.

[6] A. Cobham, *The intrinsic computational difficulty of functions*, in: Y. Bar-Hillel (Ed.), Logic, Methodology and Philosophy of Science II, North-Holland, Amsterdam 1965.

[7] K. Compton and C. Laflamme, *An Algebra and a Logic for NC^1*, Information and Computation **87** (1990), 241–263.

[8] R. Fagin, *Generalized first-order spectra and polynomial-time recognizable sets*, SIAM-AMS Proc. **7** (1974), 43–73.

[9] A. Goerdt, *Characterizing complexity classes by general recursive definitions in higher types*, Proceedings of 2nd workshop on Computer Science Logic CSL '88, Lecture Notes in Computer Science Nr. 385, Springer (1989), 99–117.

[10] E. Grädel, *Capturing Complexity Classes by Fragments of Second Order Logic*, Theoretical Computer Science **101** (1992), 35–57.

[11] E. Grädel and M. Otto, *Inductive Definability with Counting on Finite Structures*, Proceedings of CSL92, Lecture Notes in Computer Science Nr ??? (1993), 231–247.

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

[13] Y. Gurevich, *Toward logic tailord for computational complexity*, in: M. M. Richter et al. (Eds), Computation and Proof Theory, Springer Lecture Notes in Mathematics Nr. 1104 (1984), 175–216.

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

[15] Y. Gurevich and H. Lewis, *A Logic for Constant-Depth Circuits*, Information and Control **61** (1984), 65–74.

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

[17] N. Immerman, *Upper and lower bounds for first-order expressibility*, J. Comput. System Sci. **25** (1982), 76–98.

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

[19] N. Immerman, *Languages that Capture Complexity Classes*, SIAM J. Comput. **16** (1987), 760–778.

[20] N. Immerman, *Nondeterministic space is closed under complementation*, SIAM J. Comput. **17** (1988), 935–939.

[21] N. Immerman, *Descriptive and Computational Complexity*, in: J. Hartmanis (Ed.), Computational Complexity Theory, Proc. of AMS Symposia in Appl. Math. **38** (1989), 75–91.

[22] M. Krentel, *The Complexity of Optimizaton Problems*, Journal of Computer and System Sciences **36** (1988), 490–509.

[23] A. Livchak, *The Relational Model for Process Control*, Automatic Documentation and Mathematical Linguistics **4** (1983), 27–29 (in Russian).

[24] Libo Lo, *Functions and Functionals on Finite Systems*, Journal of Symbolic Logic **57** (1992), 118–130.

[25] V. Sazonov, *Polynomial computability and recursivity in finite domains*, Elektronische Datenverarbeitung und Kybernetik **16** (1980), 319–323.

[26] R. Szelepcsényi, *The Method of Forced Enumeration for Nondeterministic Automata*, Acta Informatica **26** (1988), 279–284.

[27] B. Trakhtenbrot, *Finite automata and the logic of monadic predicates*, Doklady Akademii Nauk SSR **140** (1961), 326–329.

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