Tailoring Recursion to Characterize Non-Deterministic Complexity Classes Over Arbitrary Structures

O. Bournez,[1] F. Cucker*[2], P. Jacobé de Naurois,[1] and J.-Y. Marion[1]

[1] *LORIA, 615 rue du Jardin Botanique,BP 101, 54602 Villers-lès-Nancy Cedex, Nancy, FRANCE*

{Olivier.Bournez,Paulin.De-Naurois,Jean-Yves.Marion}@loria.fr

[2] *Department of Mathematics, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, HONG KONG*

macucker@math.cityu.edu.hk

Abstract We provide machine-independent characterizations of some complexity classes, over an arbitrary structure, in the model of computation proposed by L. Blum, M. Shub and S. Smale. We show that the levels of the polynomial hierarchy correspond to safe recursion with predicative minimization. The levels of the digital polynomial hierarchy correspond to safe recursion with digital predicative minimization. Also, we show that polynomial alternating time corresponds to safe recursion with predicative substitutions and that digital polynomial alternating time corresponds to safe recursion with digital predicative substitutions.

1 Introduction

Classical complexity can be considered as the restriction to finite structures of a more general notion of computability and complexity over arbitrary structures, see [4, 20]. To understand computability in a whole perspective, it is therefore interesting to study machine-independent characterizations of complexity classes over arbitrary structures.

We focus on function algebras characterizing classical complexity classes as initiated by Bellantoni and Cook [3], Leivant [17] and Marion [18]. This *implicit* approach, stemming on Fagin seminal logical characterization of non-deterministic polynomial time and other works [11, 6, 16, 10, 15, 21] is based on a purely syntactic distinction between different types of arguments, and

*Partially supported by City University of Hong Kong SRG grant 7001290.

avoids explicit upper bounds on computational resources or restrictions on the growth as originally done by Cobham in [7].

In a previous paper [5], based on classical characterizations in [3] and [19], we exhibited machine-independent characterizations of the classes of functions over an arbitrary structure computable in polynomial sequential or parallel time. Our aim here is to provide such machine-independent characterizations over an arbitrary structure for polynomial hierarchy and polynomial alternating time. Our characterizations need to coincide with the classical ones when restricted to a structure yielding the classical notion of computation.

Over an arbitrary structure, two kinds of nondeterminism may be considered according to whether the witness is allowed to be an arbitrary element of the structure or is restricted to be in $\{0,1\}$. The latter is usually called *digital* and a letter D is used to denote complexity classes arising from the use of digital nondeterminism. Note that in classical complexity theory, i.e., over a finite structure, these two notions of nondeterminism coincide and they yield the same polynomial hierarchy and class of polynomial alternating time. Moreover, polynomial alternating time coincides with PSPACE and with PAR (the class of sets decided in parallel polynomial time). This need not to be so over infinite structures. For instance, over $(\mathbb{R}, +, -, *, /, \leq)$, we have the following inclusions of complexity classes [9]

$$\begin{tikzcd}
 & \mathrm{DPAT}_{\mathbb{R}} \arrow[dr] & & \mathrm{PAT}_{\mathbb{R}} \arrow[dr] & \\
\mathrm{DPH}_{\mathbb{R}} \arrow[ur] \arrow[dr] & & \mathrm{PAR}_{\mathbb{R}} \arrow[ur] \arrow[dr] & & \mathrm{PEXP}_{\mathbb{R}} \\
 & \mathrm{PH}_{\mathbb{R}} \arrow[ur] & & \mathrm{EXP}_{\mathbb{R}} \arrow[ur] &
\end{tikzcd}$$

where an arrow means inclusion, $\mathrm{EXP}_{\mathbb{R}}$ denotes exponential time, $\mathrm{PEXP}_{\mathbb{R}}$ parallel exponential time, $\mathrm{PH}_{\mathbb{R}}$ is the polynomial hierarchy, and $\mathrm{PAT}_{\mathbb{R}}$ polynomial alternating time. In addition the two inclusions $\mathrm{PAR}_{\mathbb{R}} \subset \mathrm{PAT}_{\mathbb{R}}$ and $\mathrm{PAR}_{\mathbb{R}} \subset \mathrm{EXP}_{\mathbb{R}}$ are known to be strict.

Concerning classical complexity, our characterizations of PAT and DPAT, combined with our previous one of PAR in [5], provide several new original alternative characterizations of PSPACE.

Concerning complexity over arbitrary structures, we believe our characterizations to provide nice and natural definitions of complexity classes. First, our characterizations are machine-independent and avoid usual technical considerations about machines. Second, they do not require over arbitrary structure to distinguish two (not so natural) types of functions (called “number terms” and “index terms” in [14]) in order to be able to use finiteness considerations over the models even in presence of infinite underlying domains like the field of real numbers as in [13, 14].

We believe that the minimization schemes we introduce for coping with non-determinism, related to Hilbert choice operator and to the operators used to tailor recursion [1, 12] shed some light on the nature of choice operators.

Based on previous characterizations of deterministic complexity classes [5], recalled in Section 2 and 3, we provide in Section 4 a characterization of the polynomial hierarchy. Minor changes allow us to characterize the digital polynomial hierarchy in Section 5. Section 6 is devoted to a characterization of polynomial alternating time, with a similar characterization of digital polynomial alternating time in Section 7.

2 Arbitrary Structures

Definition 1

A structure $\mathcal{K} = (\mathbb{K}, \{op_i\}_{i \in I}, rel_1, \ldots, rel_l, \mathbf{0}, \mathbf{1})$ *is given by some underlying set* $\mathbb{K}$*, a family of operators* op_i*, and a finite number of relations* $rel_1, \ldots, rel_l$*. Constants correspond to operators of arity 0. While the index set* I *may be infinite, the number of operators of non-null arity needs to be finite.*

We will not distinguish between operator and relation symbols and their corresponding interpretations as functions and relations respectively over the underlying set $\mathbb{K}$. We assume that the equality relation $=$ is a relation of the structure, and that there are at least two constant symbols, with different interpretations (denoted by $\mathbf{0}$ and $\mathbf{1}$ in our work) in the structure.

An example of structure is $\mathcal{K} = (\mathbb{R}, +, -, *, =, \leq, \{c_r\}_{r \in \mathbb{R}})$. Another example, corresponding to classical complexity and computability theory, is $\mathcal{K} = (\{0, 1\}, =, \mathbf{0}, \mathbf{1})$.

We denote by $\mathbb{K}^* = \bigcup_{i \in \mathbb{N}} \mathbb{K}^i$ the set of words over the alphabet $\mathbb{K}$. The space $\mathbb{K}^*$ is the analogue to Σ^* the set of all finite sequences of zeros and ones. Words of elements in $\mathbb{K}$ will be represented with overlined letters, while elements in $\mathbb{K}$ will be represented by letters: $a.\overline{x}$ stands for the word in $\mathbb{K}^*$ whose first letter is a and which ends with the word $\overline{x}$. We denote by ϵ the empty word. The length of a word $\overline{w} \in \mathbb{K}^*$ is denoted by $|\overline{w}|$.

We assume that the reader has some familiarities with the BSS model of computation. Detailed accounts can be found in [4] —for structures like real and complex numbers— or [20] —for considerations about more general structures.

Roughly speaking, a BSS machine over $\mathcal{K}$ is a a kind of Turing machine which is able to perform the basic operations op_i and the basic tests $rel_1, \ldots, rel_l$ at unit cost, and whose tape cells can hold arbitrary elements of the underlying set $\mathbb{K}$. Operations op_i of arity 0, i.e., constants, occur in a finite number in every machine. [20, 4].

In this setting resources such as time, parallel time or alternating time can be considered allowing one to define several complexity classes. For example, a problem $P \subset K^*$ will be said polynomial iff there exists a machine that, given some word $w = a_1.a_2.\ldots.a_n \in K^*$, determine whether $w \in P$ using a polynomial number in the length n of w of basic operations and basic tests. For most natural complexity classes, complete problems can be exhibited [4].

In a previous paper [5], we provided machine independent characterizations of the class of computable functions and of the class of functions computable in polynomial time. Since this work is based on the latter characterization, we next briefly recall our previous result.

3 Safe Recursive Functions

We shall define formally the set of safe recursive functions over an arbitrary structure $\mathcal{K}$, extending the notion of safe recursive functions over the natural numbers found in [3]. Safe recursive functions are defined in a similar manner as primitive recursive functions, i.e. as the closure of some basic functions under the application of some operations, among which one operation of safe recursion. However, in the spirit of [3], safe recursive functions have two different types of arguments, each of them having different properties and purposes. The first type of argument, called *normal*, can be used to make basic computation steps or to control recursion. The second type of argument, called *safe*, can not be used to control recursion. This distinction between safe and normal arguments ensures that safe recursive functions can be computed in polynomial time. Algebras of functions with this distinction between safe and normal arguments are sometimes denoted as BC functions, referring to Bellantoni and Cook [3].

To emphasize the distinction between normal and safe variables we will write $f : N \times S \to R$ where N indicates the domain of the normal arguments and S that of the safe arguments. If all the arguments of f are of one kind, say safe, we will write $\emptyset$ in the place of N. If $\overline{x}$ and $\overline{y}$ are these arguments, we will write $f(\overline{x};\overline{y})$ separating them by a semicolon ";". Normal arguments are placed at the left of the semicolon and safe arguments at its right.

Definition 2 We call *basic functions* the following four kinds of functions:

(i) functions making elementary manipulations of words over $\mathbb{K}$. For any $a \in \mathbb{K}, \overline{x}, \overline{x_1}, \overline{x_2} \in \mathbb{K}^*$

$$\begin{array}{rclrclrcl} \mathsf{hd}(;a.\overline{x}) & = & a & \mathsf{tl}(;a.\overline{x}) & = & \overline{x} & \mathsf{cons}(;a.\overline{x_1},\overline{x_2}) & = & a.\overline{x_2} \\ \mathsf{hd}(;\epsilon) & = & \epsilon & \mathsf{tl}(;\epsilon) & = & \epsilon & \mathsf{cons}(;\epsilon,\overline{x_2}) & = & \overline{x_2}. \end{array}$$

(ii) projections. For any $n \in \mathbb{N}$, $i \leq n$, $\mathsf{Pr}_i^n(;\overline{x_1},\ldots,\overline{x_i},\ldots,\overline{x_n}) = \overline{x_i}$.

(iii) functions of structure. For any operator (including the constants treated as operators of arity 0) op_i or relation rel_i of arity n_i we have the following initial functions (the equality relation will be denoted Equal).

$$\begin{aligned}\mathsf{Op}_i(; a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) &= (op_i(a_1, \ldots, a_{n_i})).\overline{x_{n_i}} \\ \mathsf{Rel}_i(; a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) &= \begin{cases} \mathbf{1} \text{ if } rel_i(a_1, \ldots, a_{n_i}) \\ \mathbf{0} \text{ otherwise.} \end{cases}\end{aligned}$$

(iv) a selector function

$$\mathsf{Select}(; \overline{x}, \overline{y}, \overline{z}) = \begin{cases} \overline{y} & \text{if } \mathsf{hd}(\overline{x}) = \mathbf{1} \\ \overline{z} & \text{otherwise.} \end{cases}$$

Definition 3 The set of *safe recursive functions* over $\mathcal{K}$, denoted by $\mathrm{SR}_{\mathcal{K}}$, is the smallest set of functions $f : (\mathbb{K}^*)^p \times (\mathbb{K}^*)^q \to \mathbb{K}^*$ containing the basic safe functions, and closed under the following operations:

(1) *Safe composition.*$g : (\mathbb{K}^*)^m \times (\mathbb{K}^*)^n \to \mathbb{K}^*$, $h_1, \ldots, h_m : \mathbb{K}^* \times \emptyset \to \mathbb{K}^*$ and $h_{m+1}, \ldots, h_{m+n} : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ are given safe recursive functions. Their safe composition is the function $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ defined by

$$f(\overline{x}; \overline{y}) = g\left(h_1(\overline{x};), \ldots, h_m(\overline{x};); h_{m+1}(\overline{x}; \overline{y}), \ldots, h_{m+n}(\overline{x}; \overline{y})\right).$$

(2) *Safe recursion.* $h : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ and $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ are given functions. $f : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ is defined by safe recursion as follows

$$\begin{aligned} f(\epsilon, \overline{x}; \overline{y}) &= h(\overline{x}; \overline{y}) \\ f(a.\overline{z}, \overline{x}; \overline{y}) &= g(\overline{z}, \overline{x}; f(\overline{z}, \overline{x}; \overline{y}), \overline{y}). \end{aligned}$$

When Φ is a set and F a complexity class, we denote by F^{Φ} the class F with oracle Φ. When G is another complexity class, F^G denotes the class F with oracles in G.

Definition 4 Given a function $\phi : \mathbb{K}^* \times \emptyset \to \mathbb{K}^*$, the set of *safe recursive functions relative to* ϕ over $\mathcal{K}$, denoted by $\mathrm{SR}_{\mathcal{K}}(\phi)$, is the smallest set of functions $f : (\mathbb{K}^*)^p \times (\mathbb{K}^*)^q \to \mathbb{K}^*$ containing the basic safe functions and ϕ, and closed under safe composition and safe recursion.

Theorem 1 *Let $\Phi \in \mathbb{K}^*$ be a decision problem over $\mathcal{K}$, and denote by $\phi : \emptyset \times \mathbb{K}^* \to \{\mathbf{0}, \mathbf{1}\}$ its characteristic function. Then, a function $f : \mathbb{K}^* \to \mathbb{K}^*$ is in the class $\mathrm{FP}_{\mathcal{K}}^{\Phi}$ of functions computable in polynomial time with oracle Φ if and only if $f : \mathbb{K}^* \times \emptyset \to \mathbb{K}*$ can be defined in $SR_{\mathcal{K}}(\phi)$.*

We consider only decision oracles and not functional oracles in order to avoid problems related to the output size of these oracles, see [8].

Corollary 1 ([5]) *Over any structure $\mathcal{K} = (\mathbb{K}, \{op_i\}_{i\in I}, rel_1, \dots, rel_l, \mathbf{0}, \mathbf{1})$, a function is computed in polynomial time by a BSS machine if and only if it is defined as a safe recursive function over $\mathcal{K}$.*

We shall now introduce a technical lemma needed further in our proofs.

Lemma 1 *Assume $f : (\mathbb{K}^*)^2 \times \emptyset \to \mathbb{K}^*$ is in $SR_{\mathcal{K}}(\phi)$. Moreover, assume that there exists a polynomial p such that, for all $\overline{x}, \overline{y} \in \mathbb{K}^*$, $f(\overline{x}, \overline{y};)$ can be evaluated in time bounded by $p(|\overline{x}|)$. Then, there exists $f' : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^* \in SR_{\mathcal{K}}(\phi)$ such that $f'(\overline{x}; \overline{y}) = f(\overline{x}, \overline{y};)$.*

Proof. The idea is to follow the proof of Corollary 1: a BSS machine, on input $\overline{z}$, can be simulated by a safe recursive function Eval such that $\mathsf{Eval}(\mathbf{0}^t; \overline{z})$ gives the content of the tape after t computation steps. Its normal argument $\mathbf{0}^t$ can be seen as a clock for the BSS machine. Assume M is a BSS-machine computing $f(\overline{x}, \overline{y};)$ in time $p(|\overline{x}|)$. Corollary 1 gives a safe recursive function $f_p : \mathbb{K}^* \times \emptyset \to \mathbb{K}^*$ such that $f_p(\overline{x};) = \mathbf{0}^{p(|\overline{x}|)}$. Consider a safe recursive function Cons such that $\mathsf{Cons}(\overline{x}; \overline{y}) = \overline{x}.\overline{y}$. Then, $f'(\overline{x}; \overline{y}) = \mathsf{Eval}(f_p(\overline{x}); \mathsf{Cons}(\overline{x}; \overline{y}))$.

4 A Characterization of $\mathrm{PH}_{\mathcal{K}}$

As in the classical settings, the polynomial hierarchy over a given structure $\mathcal{K}$ can be defined in several equivalent ways, including syntactic descriptions, or semantic definitions by successive relativizations of non-deterministic polynomial time (see [4]).

Recall some basic complexity classes:

- $\mathrm{P}_{\mathcal{K}}$ is the class of problems over $\mathcal{K}$ decided in polynomial time. We denote by $\mathrm{FP}_{\mathcal{K}}$ the class of functions over $\mathcal{K}$ computed in polynomial time.
- A decision problem A is in $\mathrm{NP}_{\mathcal{K}}$ if and only if there exists a decision problem B in $\mathrm{P}_{\mathcal{K}}$ and a polynomial p_B such that $\overline{x} \in A$ if and only if there exists $\overline{y} \in \mathbb{K}^*$ with $|\overline{y}| \leq p_B(|\overline{x}|)$ satisfying $(\overline{x}, \overline{y}) \in B$.
- A decision problem A is in $\mathrm{coNP}_{\mathcal{K}}$ if and only if there exists a decision problem B in $\mathrm{P}_{\mathcal{K}}$ and a polynomial p_B such that $\overline{x} \in A$ if and only if for all $\overline{y} \in \mathbb{K}^*$ with $|\overline{y}| \leq P_B(|\overline{x}|)$, $(\overline{x}, \overline{y})$ is in B.

Definition 5 Let $\Sigma^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$ and, for $i \geq 1$, $\Sigma^i_{\mathcal{K}} = \mathrm{NP}_{\mathcal{K}}^{\Sigma^{i-1}_{\mathcal{K}}}$, $\Pi^i_{\mathcal{K}} = \mathrm{coNP}_{\mathcal{K}}^{\Sigma^{i-1}_{\mathcal{K}}}$. The *polynomial time hierarchy* over $\mathcal{K}$ is $\mathrm{PH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \Sigma^i_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \Pi^i_{\mathcal{K}}$. A function in $\mathrm{F}\Delta^i_{\mathcal{K}}$ is a polynomial time function over $\mathcal{K}$ which queries $\Sigma^i_{\mathcal{K}}$ oracles: $\mathrm{F}\Delta^i_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}} = \mathrm{FP}_{\mathcal{K}}^{\Pi^i_{\mathcal{K}}}$. The *functional polynomial time hierarchy* over $\mathcal{K}$ is $\mathrm{FPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{F}\Delta^i_{\mathcal{K}}$.

Remark Extending the classical notion of polynomial time reduction between decision problems, complete problems for every of the $\Sigma^i_{\mathcal{K}}$ and $\Pi^i_{\mathcal{K}}$ have been shown to exist [4].

In the spirit of [2], we now introduce the notion of predicative minimization (we use the terminology “minimization” taken from [2], even if this might be considered as not being a true minimization.).

Definition 6 Given $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \rightarrow \mathbb{K}^*$, we define $f : \mathbb{K}^* \times \mathbb{K}^* \rightarrow \mathbb{K}$ by *predicative minimization* as follows

$$f(\overline{x};\overline{a}) = \exists\overline{b}(h(\overline{x};\overline{a},\overline{b})) = \begin{cases} \mathbf{1} & \text{if there exists } \overline{b} \in \mathbb{K}^* \text{ with } h(\overline{x};\overline{a},\overline{b}) = \mathbf{0} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Remark Predicative minimization applied on functions h for which one can guarantee the existence of a $\overline{b}$ of polynomial size in the length of $\overline{x}$ when there is one, preserves (non-deterministic) computability. This consideration will assure computability of functions of our considered classes, in analogy with the “polychecking-lemma” used in [2] to guarantee computability.

We now introduce new sets of functions.

Definition 7 *Let F be a class of BC functions. The set of* restricted safe recursive functions relative to F *over $\mathcal{K}$, denoted by $RSR_{\mathcal{K}}(F)$, is the smallest set of functions containing the basic safe functions and F, and closed under the following* restricted safe composition *scheme*

$$f(\overline{x};\overline{y}) = g\left(h_1(\overline{x};),\ldots,h_m(\overline{x};);h_{m+1}(\overline{x};\overline{y}),\ldots,h_{m+n}(\overline{x};\overline{y})\right).$$

where the h_i belong to $RSR_{\mathcal{K}}(F)$ and g to $SR_{\mathcal{K}}$, and the following restricted safe recursion *scheme*

$$\begin{aligned} f(\epsilon,\overline{x};\overline{y}) &= h(\overline{x};\overline{y}) \\ f(a.\overline{z},\overline{x};\overline{y}) &= g(\overline{z},\overline{x};f(\overline{z},\overline{x};\overline{y}),\overline{y}) \end{aligned}$$

where h belongs to $RSR_{\mathcal{K}}(F)$ and g to $SR_{\mathcal{K}}$. This implies that no function in $F \backslash SR_{\mathcal{K}}$ may be involved in the definition of g.

Definition 8 Assume F is a class of functions: a function f is in $\exists$F if it is defined with one predicative minimization over a function h of F.

We define by induction the following sets:

- $\mathrm{F}^0_{\mathcal{K}} = \mathrm{SR}_{\mathcal{K}}$.
- $\mathrm{F}^{i+1}_{\mathcal{K}} = \mathrm{RSR}_{\mathcal{K}}(\mathrm{F}^i_{\mathcal{K}} \bigcup \exists\mathrm{F}^i_{\mathcal{K}})$, for $i \geq 0$.

We denote by $\exists\mathrm{PH}_{\mathcal{K}} = \bigcup_{i\in\mathbb{N}} \mathrm{F}^i_{\mathcal{K}}$ the closure of the basic safe functions over $\mathcal{K}$ under the application of restricted safe recursion, predicative minimization and safe composition.

Lemma 2 *This notion of restricted safe recursion ensures that, for any function f in $F^i_{\mathcal{K}}$, there are at most i nested predicative minimizations. This bound does not depend on the arguments of f. In other words, there exists h in $SR_{\mathcal{K}}$ and $f_1, \dots, f_n$ in $F^{i-1}_{\mathcal{K}}$, such that, for all $\mathbf{x} = (\overline{x_1}, \dots, \overline{x_l})$,*

$$f(\mathbf{x};) = h(\mathbf{x}; \boldsymbol{\ni}\overline{z_1}(f_1(\mathbf{x}; \overline{z_1})), \dots, \boldsymbol{\ni}\overline{z_n}(f_n(\mathbf{x}; \overline{z_n}))).$$

We denote this as a normal form for f.

Lemma 3 *Assume $f : (\mathbb{K}^*)^n \times \emptyset \to \mathbb{K}^*$ is a function in $\mathrm{F}\Delta^i_{\mathcal{K}}$. Then f can be defined in $F^i_{\mathcal{K}}$.*

PROOF. By induction on i. For $i = 0$, f is in $\mathrm{F}\Delta^0_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}$ and we may apply Corollary 1. Assume now that the result holds for $i > 0$.

Let f be a function in $\mathrm{F}\Delta^i_{\mathcal{K}}$. By definition of $\mathrm{F}\Delta^i_{\mathcal{K}}$, there exist a polynomial time BSS machine M_f and a set Φ in $\Sigma^i_{\mathcal{K}}$ such that, for all $\overline{x} \in \mathbb{K}^*$, $f(\overline{x})$ is computed by M_f with oracle Φ. We are now establishing that the oracle Φ can be denoted by a function in $\mathrm{F}^i_{\mathcal{K}}$.

Since $\Phi \in \Sigma^i_{\mathcal{K}} = \mathrm{NP}^{\Sigma^{i-1}_{\mathcal{K}}}_{\mathcal{K}}$ there exist a deterministic polynomial-time BSS machine M_g over $\mathcal{K}$, a polynomial p and a set $\Psi \in \Sigma^{i-1}_{\mathcal{K}}$ such that

$$\overline{x} \in \Phi \Leftrightarrow \exists \overline{y} \text{ s.t. } M_g \text{ accepts } (\overline{x}, \overline{y}) \text{ with oracle } \Psi \text{ and} |\overline{y}| < p(|\overline{x}|).$$

Denote by g the characteristic function computed by M_g with oracle Ψ and let ψ be the characteristic function of Ψ. Then, apply Theorem 1: g belongs to $\mathrm{SR}(\psi)_{\mathcal{K}}$. Since the evaluation time of M_g on $(\overline{x}, \overline{y})$ is polynomial in $|\overline{x}|$, Lemma 1 gives g' in $\mathrm{SR}(\psi)_{\mathcal{K}}$ such that: $g'(\overline{x}; \overline{y}) = g(\overline{x}, \overline{y};)$. Therefore $\phi(\overline{x};) = \boldsymbol{\ni}\overline{y}(g'(\overline{x}; \overline{y}))$ decides Φ, and, since $\Sigma^{i-1}_{\mathcal{K}} \subseteq \mathrm{F}\Delta^{i-1}_{\mathcal{K}}$, we may apply the induction hypothesis on ψ to deduce that ϕ belongs to $\mathrm{F}^i_{\mathcal{K}}$ and therefore so does f.

Lemma 4 *Assume $f : (\mathbb{K}^*)^n \times \emptyset \to \mathbb{K}^*$ is a function in $F^i_{\mathcal{K}}$. Then it belongs to $\mathrm{F}\Delta^i_{\mathcal{K}}$.*

PROOF. By induction on i. For $i = 0$, the result is a straightforward consequence of Corollary 1. Assume now that the result holds for $i > 0$.

Assume f is a function in $\mathrm{F}^i_{\mathcal{K}}$. Then, as in Lemma 2,

$$f(\mathbf{x};) = h(\mathbf{x}; \boldsymbol{\ni}\overline{z_1}(f_1(\mathbf{x}; \overline{z_1})), \dots, \boldsymbol{\ni}\overline{z_n}(f_n(\mathbf{x}; \overline{z_n}))).$$

By induction hypothesis, the functions $f_1, \dots, f_n$ belong to $\mathrm{F}\Delta^{i-1}_{\mathcal{K}}$. The corresponding decision problems $f_1(\mathbf{x}; \overline{z_1}) = \mathbf{0}, \dots, f_n(\mathbf{x}; \overline{z_n}) = \mathbf{0}$ belong to $\mathrm{P}^{\Sigma^{i-1}_{\mathcal{K}}}_{\mathcal{K}} = \Sigma^{i-1}_{\mathcal{K}}$. Indeed, they use a polynomial number of queries in $\Sigma^{i-1}_{\mathcal{K}}$. If S_{i-1} denotes a complete problem in $\Sigma^{i-1}_{\mathcal{K}}$ (see Remark 1), we can replace these different oracles by S_{i-1} (by making the oracle machine compute the reductions).

Define $g_j(\mathbf{x};) = \ni\overline{z_j}(f_j(\mathbf{x};\overline{z_j}))$ for $1 \leq j \leq n$. Then, g_j is the characteristic function of a set in $\Sigma^i_{\mathcal{K}}$. Indeed, if there exists $\overline{z_j} \in \mathbb{K}^*$ such that $f_j(\mathbf{x};\overline{z_j}) = \mathbf{0}$, since the evaluation time for $f_j(\mathbf{x};\overline{z_j})$ is bounded by $p_j(|\mathbf{x}|)$ for some polynomial p_j, only the first $p_j(|\mathbf{x}|)$ elements of $\overline{z_j}$ may possibly be taken into account. Therefore, there exists $\overline{z'_j} \in \mathbb{K}^*$ of length $p_j(|\mathbf{x}|)$ such that $f_j(\mathbf{x};\overline{z'_j}) = \mathbf{0}$, which proves the claim. Therefore, f can be computed in polynomial time using n oracles in $\Sigma^i_{\mathcal{K}}$. If S_i denotes a complete problem in $\Sigma^i_{\mathcal{K}}$, again, we can replace these n different oracles by S_i: $f \in \mathrm{FP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}} = \mathrm{F}\Delta^i_{\mathcal{K}}$.

This gives our first main characterization.

Theorem 2 *A function: $(\mathbb{K}^*)^n \times \emptyset \to \mathbb{K}^*$ belongs to $\mathrm{F}\Delta^i_{\mathcal{K}}$ if and only if it is defined in $F^i_{\mathcal{K}}$.*

Example Over the real numbers, an example of $\mathrm{NP}_{\mathbb{R}}$-complete problem is $4-$FEAS: does a given polynomial of degree four have a zero? Assume by Corollary 1 that the safe recursive function $p(\overline{x};\overline{y})$ evaluates a polynomial encoded in $\overline{x}$ on an input encoded in $\overline{y}$. $4-$FEAS is then decided on $\overline{x}$ by $f(\overline{x};) = \ni\overline{y}(p(\overline{x};\overline{y}))$.

Corollary 2 *A decision problem over $\mathcal{K}$ belongs to $\mathrm{PH}_{\mathcal{K}}$ if and only if its characteristic function is defined in $\ni\mathrm{PH}_{\mathcal{K}}$.*

5 A Characterization of $\mathrm{DPH}_{\mathcal{K}}$

Definition 9 A set $S \subseteq \mathbb{K}^*$ belongs to $\mathrm{DNP}_{\mathcal{K}}$ if and only if there exist a polynomial p and a polynomial time BSS machine M over $\mathcal{K}$ such that, for all $\overline{x} \in \mathbb{K}^*$,

$$\overline{x} \in S \Leftrightarrow \exists \overline{y} \in \{\mathbf{0},\mathbf{1}\}^* \text{ s.t. } |\overline{y}| \leq p(|\overline{x}|) \text{ and } M \text{ accepts } (\overline{x},\overline{y}).$$

Let $\mathrm{D}\Sigma^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$ and, for $i \geq 1$, $\mathrm{D}\Sigma^i_{\mathcal{K}} = \mathrm{DNP}_{\mathcal{K}}^{\mathrm{D}\Sigma^{i-1}_{\mathcal{K}}}$, $\mathrm{D}\Pi^i_{\mathcal{K}} = \mathrm{coDNP}_{\mathcal{K}}^{\mathrm{D}\Sigma^{i-1}_{\mathcal{K}}}$. The *digital polynomial time hierarchy* is $\mathrm{DPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{D}\Sigma^i_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{D}\Pi^i_{\mathcal{K}}$. A function in $\mathrm{DF}\Delta^i_{\mathcal{K}}$ is a polynomial time function over $\mathcal{K}$ which queries $\mathrm{D}\Sigma^i_{\mathcal{K}}$ oracles: $\mathrm{DF}\Delta^i_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{D}\Sigma^i_{\mathcal{K}}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{D}\Pi^i_{\mathcal{K}}}$. The *functional digital polynomial time hierarchy* is $\mathrm{DFPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{DF}\Delta^i_{\mathcal{K}}$.

In this digital version of the polynomial hierarchy, witnesses for a given problem are discrete choices among given values, and not arbitrary elements of the structure. As in the previous section, complete problems have been shown to exist for every level of this hierarchy [4].

Similarly to the notion of predicative minimization of the previous section, we introduce the notion of digital predicative minimization.

Definition 10 Given $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, we define $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ by *digital predicative minimization* as follows

$$f(\overline{x}; \overline{a}) = \ni_{\mathrm{D}} \overline{b}(h(\overline{x}; \overline{a}, \overline{b})) = \begin{cases} \mathbf{1} & \text{if there is a } \overline{b} \in \{\mathbf{0}, \mathbf{1}\}^* \text{ with } h(\overline{x}; \overline{a}, \overline{b}) = \mathbf{0} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Definition 11 Let F be a class of functions. A function f is in $\ni_{\mathrm{D}}\mathrm{F}$ if it is defined with one predicative minimization over a function h of F.

We define by induction the following sets:

- $\mathrm{dF}^0_{\mathcal{K}} = \mathrm{SR}_{\mathcal{K}}$
- $\mathrm{dF}^{i+1}_{\mathcal{K}} = \mathrm{RSR}_{\mathcal{K}}(\mathrm{dF}^i_{\mathcal{K}} \bigcup \ni_{\mathrm{D}} \mathrm{F}^i_{\mathcal{K}})$, for $i \geq 0$.

We denote by $\ni_{\mathrm{D}}\mathrm{PH}_{\mathcal{K}}$ the closure of the basic safe functions over $\mathcal{K}$ under the application of projections, restricted safe recursion, digital predicative minimization and safe composition.

The proof of Theorem 2, *mutatis mutandis*, yields the following results.

Theorem 3 *A function:* $(\mathbb{K}^*)^n \times \emptyset \to \mathbb{K}^*$ *belongs to* $\mathrm{DF}\Delta^i_{\mathcal{K}}$ *if and only if it is defined in* $dF^i_{\mathcal{K}}$.

Corollary 3 *A decision problem over* $\mathcal{K}$ *belongs to* $\mathrm{DPH}_{\mathcal{K}}$ *if and only if its characteristic function is defined in* $\ni_{\mathrm{D}}\mathrm{DPH}_{\mathcal{K}}$.

Example Over the real numbers, a problem in $\mathrm{D}\Sigma^1_{\mathbb{R}}$ is KNAPSACK: given n objects of weight $w_i \in \mathbb{R}$ and value $v_i \in \mathbb{R}$, a weight limit W and a minimal value V, can we select a subset of objects of total value greater than V and of total weight less than W? Assume by Corollary 1 that the safe recursive function $v(\overline{x}; \overline{y})$ decides whether, for an instance described by $\overline{x}$ in size polynomial in n, a choice among the objects described by $\overline{y} \in \{\mathbf{0}, \mathbf{1}\}^n$, the requirements of weight and value are satisfied. KNAPSACK is then decided on $\overline{x}$ by$f(\overline{x};) = \ni_{\mathrm{D}} \overline{y}(v(\overline{x}; \overline{y}))$. When considering finite structures, this yields naturally a characterization of the classical polynomial hierarchy alternative to the one found in [2]:

Corollary 4 *A decision problem belongs to* PH *if and only if its characteristic function is defined in* $\ni_{\mathrm{D}}\mathrm{DPH}_{\{\mathbf{0},\mathbf{1}\}}$.

6 A Characterization of $\mathrm{PAT}_{\mathcal{K}}$

Definition 12 A set $S \subseteq \mathbb{K}^*$ belongs to $\mathrm{PAT}_{\mathcal{K}}$ (*polynomial alternating time*) if and only if there exist a polynomial function $q : \mathbb{N} \to \mathbb{N}$ and a polynomial time BSS machine M_S over $\mathcal{K}$ such that, for all $\overline{x} \in \mathbb{K}^*$,

$$\overline{x} \in S \quad \Leftrightarrow \quad \exists a_1 \in \mathbb{K}\, \forall b_1 \in \mathbb{K} \, \ldots \, \exists a_{q(|\overline{x}|)} \in \mathbb{K}\, \forall b_{q(|\overline{x}|)} \in \mathbb{K}$$
$$M_S \text{ accepts } (\overline{x}, a_1.b_1 \ldots a_{q(|\overline{x}|)}.b_{q(|\overline{x}|)}).$$

In addition, we define $\mathrm{FPAT}_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{PAT}_{\mathcal{K}}}$.

When $\mathcal{K}$ is the classical structure $\{\{0,1\},=,\mathbf{0},\mathbf{1}\}$, $\mathrm{PAT}_{\mathcal{K}}$ is PSPACE.

It is important to note that the number of quantifier alternations is not fixed, but depends on the length of the input and is polynomial in that length. It follows that $\mathrm{PH}_{\mathcal{K}} \subseteq \mathrm{PAT}_{\mathcal{K}}$.

Definition 13 Given $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, we define $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ by *predicative substitution* as follows

$$f(\overline{x};\overline{a}) = \ni^{[1]}c(h(\overline{x};\overline{a},c)) = \begin{cases} \mathbf{1} & \text{if there is a } c \in \mathbb{K} \text{ with } h(\overline{x};\overline{a},c) = \mathbf{0} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Definition 14 Assume $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ and $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ are given functions. The function $f : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ is defined by *safe recursion with predicative substitution* as follows

$$\begin{aligned} f(\epsilon,\overline{x};\overline{u},\overline{y}) &= h(\overline{x};\overline{u},\overline{y}) \\ f(a.\overline{z},\overline{x};\overline{u},\overline{y}) &= g(\overline{z},\overline{x};\ni^{[1]}c(f(\overline{z},\overline{x};c.\overline{u},\overline{y})),\overline{y}). \end{aligned}$$

Definition 15 The set $\ni^{[1]}\mathrm{PAT}_{\mathcal{K}}$ of *safe recursive functions with predicative substitutions* over $\mathcal{K}$ is the closure of the basic safe functions under the application of safe composition, safe recursion and safe recursion with predicative substitutions.

Theorem 4 *A function is computed in* $\mathrm{FPAT}_{\mathcal{K}}$ *if and only if it can be defined in* $\ni^{[1]}\mathrm{PAT}_{\mathcal{K}}$.

PROOF. Let F be a function in $\mathrm{FPAT}_{\mathcal{K}}$, and denote by G the associated oracle in $\mathrm{PAT}_{\mathcal{K}}$. There exists a polynomial time BSS machine M over $\mathcal{K}$, and a polynomial function $q : \mathbb{N} \to \mathbb{N}$ such that, for all $\overline{x} \in \mathbb{K}^*$,

$$\begin{aligned} \overline{x} \in G \quad \Leftrightarrow \quad & \exists a_1 \in \mathbb{K}\, \neg\exists b_1 \in \mathbb{K} \,\ldots\, \exists a_{q(|\overline{x}|)} \in \mathbb{K}\, \neg\exists b_{q(|\overline{x}|)} \in \mathbb{K} \\ & M \text{ accepts } (\overline{x}, a_1.b_1 \ldots a_{q(|\overline{x}|)}.b_{q(|\overline{x}|)}). \end{aligned}$$

Corollary 1 and Lemma 1 ensure that there exists a safe recursive function f_M over $\mathcal{K}$ such that, for any $(\overline{x},\overline{y}) \in (\mathbb{K}^*)^2$, M accepts on input $(\overline{x},\overline{y})$ if and only if $f_M(\overline{x};\overline{y}) = \mathbf{1}$.

Consider now the function $F_G : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ deciding G. $F_G(\epsilon,\overline{x};\overline{u})$ simulates M on input $\overline{x},\overline{u}$. The recurrence parameter $a.\overline{z}$ in $F_G(a.\overline{z},\overline{x};\overline{u})$ describes the shape of the quantifier sequence. F_G is defined with quantified safe recursion as follows,

$$\begin{aligned} F_G(\epsilon,\overline{x};\overline{u}) &= f_M(\overline{x};\overline{u}) \\ F_G(a.\overline{z},\overline{x};\overline{u}) &= \mathsf{Select}(;\mathsf{Equal}(;\mathsf{hd}(;\overline{z}),\mathbf{1}),\ni^{[1]}c(F_G(\overline{z},\overline{x};c.\overline{u})), \\ &\mathsf{Select}(;\mathsf{Equal}(;\mathsf{hd}(;\overline{z}),\mathbf{0}),\mathsf{Select}(;\ni^{[1]}c(F_G(\overline{z},\overline{x};c.\overline{u})),\mathbf{0},\mathbf{1}),f_M(\overline{x};\overline{u}))). \end{aligned}$$

In addition, let $g_q: \mathbb{K}^* \times \emptyset \to \mathbb{K}^*$ such that $g_q(\overline{x};) = (\mathbf{1}.\mathbf{0})^{q(|\overline{x}|)}$. Since g_q is computable in polynomial time over $\mathcal{K}$, by Corollary 1, it is safe recursive. This function g_q actually gives the type of the quantifier at every level of the quantifier alternation for any input $\overline{x}$ to the problem G.

It is easy to check by induction on $|\overline{x}|$ that $F_G(\mathsf{cons}(\mathbf{1}, g_q(\overline{x};);), \overline{x}; \mathbf{0})$ decides whether $\overline{x}$ belongs to G. Therefore, the characteristic function χ_G of G belongs to $\ni^{[1]}\mathrm{PAT}_{\mathcal{K}}$.

Consider a polynomial time machine M' with oracle G computing F. We apply Theorem 1: F belongs to $\mathrm{SR}_{\mathcal{K}}(F_G)$, i.e., $F \in \ni^{[1]}\mathrm{PAT}_{\mathcal{K}}$.

The other direction of the proof is by induction on the definition of f. The only critical case is when f is defined by safe recursion with predicative substitution, as in Definition 14. In this case, $f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y})$ equals $\mathbf{1}$ if and only if

$$\begin{array}{ll} & (\exists c \in \mathbb{K}\ f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y}) = \mathbf{0} \wedge g(\overline{z}, \overline{x}; \mathbf{1}, \overline{y}) = \mathbf{1}) \\ \vee & (\forall c \in \mathbb{K}\ f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y}) \neq \mathbf{0} \wedge g(\overline{z}, \overline{x}; \mathbf{0}, \overline{y}) = \mathbf{1})\,. \end{array}$$

If $g(\overline{z}, \overline{x}; \mathbf{1}, \overline{y}) = \mathbf{1}$ and $g(\overline{z}, \overline{x}; \mathbf{0}, \overline{y}) = \mathbf{1}$, then $f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) = \mathbf{1}$ and there is no need for a recursive call. If $g(\overline{z}, \overline{x}; \mathbf{1}, \overline{y}) \neq \mathbf{1}$ and $g(\overline{z}, \overline{x}; \mathbf{0}, \overline{y}) \neq \mathbf{1}$, then $f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) \neq \mathbf{1}$ and there is no need for a recursive call either. If $g(\overline{z}, \overline{x}; \mathbf{1}, \overline{y}) = \mathbf{1}$ and $g(\overline{z}, \overline{x}; \mathbf{0}, \overline{y}) \neq \mathbf{1}$, then $f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) = \mathbf{1}$ if and only if

$$\exists c \in \mathbb{K}\ f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y}) = \mathbf{0}.$$

If $g(\overline{z}, \overline{x}; \mathbf{1}, \overline{y}) \neq \mathbf{1}$ and $g(\overline{z}, \overline{x}; \mathbf{0}, \overline{y}) = \mathbf{1}$, then $f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) = \mathbf{1}$ if and only if

$$\forall c \in \mathbb{K}\ f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y}) \neq \mathbf{0}.$$

Therefore, at every level of the recursion, the choice is determined by the function g. By induction hypothesis, this can be done in $\mathrm{FPAT}_{\mathcal{K}}$. When unfolding the recursion, we get a sequence of quantifiers $Q_1, \ldots, Q_{|\overline{z}|+1}$ and a relation symbol $r \in \{=, \neq\}$ such that

$$\begin{array}{c} f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) = \mathbf{1} \\ \text{iff } Q_1 c_1 \in \mathbb{K}, \ldots, Q_{|\overline{z}|+1} c_{|\overline{z}|+1} \in \mathbb{K}\ h(\overline{x}; c_1. \ldots .c_{|\overline{z}|+1}.\overline{u}, \overline{y})\ r\ \mathbf{0}. \end{array}$$

Apply the induction hypothesis on function h. Then, f belongs to $\mathrm{FPAT}_{\mathcal{K}}^{\mathrm{FPAT}_{\mathcal{K}}}$, with an oracle which computes g and gives the quantifier sequence. One just needs to note that $\mathrm{FPAT}_{\mathcal{K}}^{\mathrm{FPAT}_{\mathcal{K}}} = \mathrm{FPAT}_{\mathcal{K}}$ to conclude.

7 A Characterization of $\mathrm{DPAT}_{\mathcal{K}}$

Class $\mathrm{DPAT}_{\mathcal{K}}$ is similar to $\mathrm{PAT}_{\mathcal{K}}$ but with all quantified variables belonging to $\{\mathbf{0}, \mathbf{1}\}$. Similarly, we can define $\mathrm{DFPAT}_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{DPAT}_{\mathcal{K}}}$.

Similarly to the notion of predicative substitution, we define the notion of digital predicative substitution.

Definition 16 Given $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, we define $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ by *predicative substitution*,

$$f(\overline{x}; \overline{a}) = \ni_{\mathsf{D}}^{[1]} c(h(\overline{x}; \overline{a}, c)) = \begin{cases} \mathbf{1} & \text{if there is } c \in \{\mathbf{0}, \mathbf{1}\} \text{ with } h(\overline{x}; \overline{a}, c) = \mathbf{0} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Definition 17 Assume $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ and $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ are given functions. The function $f : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ is defined by *safe recursion with digital predicative substitution* as follows

$$\begin{aligned} f(\epsilon, \overline{x}; \overline{u}, \overline{y}) &= h(\overline{x}; \overline{u}, \overline{y}) \\ f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) &= g(\overline{z}, \overline{x}; \ni_{\mathsf{D}}^{[1]} c f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y}), \overline{y}). \end{aligned}$$

Definition 18 The set $\ni_{\mathsf{D}}^{[1]}\mathrm{PAT}_{\mathcal{K}}$ of *safe recursive functions with digital predicative substitutions* over $\mathcal{K}$ is the closure of the basic safe functions under the application of safe composition, safe recursion and safe recursion with digital predicative substitution.

Again, the proof of Theorem 4 yields, *mutatis mutandis*, the following result.

Theorem 5 *A function is computed in* $\mathrm{DFPAT}_{\mathcal{K}}$ *if and only if it can be defined in* $\ni^{[1]}\mathrm{DPAT}_{\mathcal{K}}$.

When restricted to finite structures, this yields another characterization of PSPACE:

Corollary 5 *A decision problem is decided in* PSPACE *if and only if its characteristic function can be defined in* $\ni^{[1]}\mathrm{DPAT}_{\{\mathbf{0},\mathbf{1}\}}$.

References

[1] A.Blass and Y. Gurevich. The logic of choice. *Journal of Symbolic Logic*, 65(3):1264–1310, Sept. 2000.

[2] S. Bellantoni. Predicative recursion and the polytime hiearchy. In Peter Clote and Jeffrey B. Remmel, editors, *Feasible Mathematics II, Perspectives in Computer Science*. Birkhauser, 1994.

[3] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2:97–110, 1992.

[4] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998.

[5] O. Bournez, F. Cucker, P. Jacobe de Naurois, and J.-Y. Marion. Computability over an arbitrary structure. sequential and parallel polynomial time. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International*

Conference (FOSSACS'2003), volume 2620 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2003.

[6] P. Clote. Computational models and function algebras. In D. Leivant, editor, *LCC'94*, volume 960 of *Lecture Notes in Computer Science*, pages 98–130. Springer-Verlag, 1995.

[7] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1962.

[8] S. Cook. Computability and complexity of higher-type functions. In Y. Moschovakis, editor, *Logic from Computer Science*, pages 51–72. Springer-Verlag, New York, 1992.

[9] F. Cucker. On the complexity of quantifier elimination: the structural approach. *The Computer Journal*, 36:400–408, 1993.

[10] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1995.

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

[12] E. Gradel and Y. Gurevich. Tailoring recursion for complexity. *Journal of Symbolic Logic*, 60(3):952–969, Sept. 1995.

[13] Erich Gradel and Yuri Gurevich. Metafinite model theory. *Information and Computation*, 140(1):26–81, 10 January 1998.

[14] Erich Gradel and Klaus Meer. Descriptive complexity theory over the real numbers. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 315–324, Las Vegas, Nevada, 29 May–1 June 1995.

[15] Y. Gurevich. Algebras of feasible functions. In *Twenty Fourth Symposium on Foundations of Computer Science*, pages 210–214. IEEE Computer Society Press, 1983.

[16] N. Immerman. *Descriptive Complexity*. Springer-Verlag, 1999.

[17] D. Leivant. Predicative recurrence and computational complexity I: Word recurrence and poly-time. In Peter Clote and Jeffery Remmel, editors, *Feasible Mathematics II*, pages 320–343. Birkhauser, 1994.

[18] D. Leivant and J-Y Marion. Lambda calculus characterizations of poly-time. *Fundamenta Informaticae*, 19(1,2):167,184, September 1993.

[19] D. Leivant and J.-Y. Marion. Ramified recurrence and computational complexity II: substitution and poly-space. In L. Pacholski and J. Tiuryn, editors, *Computer Science Logic, 8th Workshop, CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 369–380, Kazimierz, Poland, 1995. Springer-Verlag.

[20] B. Poizat. *Les Petits Cailloux*. Aléas, 1995.

[21] V. Sazonov. Polynomial computability and recursivity in finite domains. *Elektronische Informationsverarbeitung und Kybernetik*, 7:319–323, 1980.