Available online at www.sciencedirect.com

Information and Computation 204 (2006) 210–230

Information and Computation

www.elsevier.com/locate/ic

Implicit complexity over an arbitrary structure: Quantifier alternations

Olivier Bournez, [a] Felipe Cucker, [b,*,1] Paulin Jacobé de Naurois, [a,1] Jean-Yves Marion [a]

[a] *LORIA, 615 rue du Jardin Botanique, BP 101, 54602 Villers-lès-Nancy Cedex, Nancy, France*
[b] *Department of Mathematics, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong*

Received 10 May 2004; revised 21 July 2005
Available online 2 November 2005

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 and the levels of the digital polynomial hierarchy 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.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Implicit complexity; Safe recursion; Arbitrary structures; BSS computation

1. Introduction

In the last decades complexity theory developed in many directions to offer a broad perspective of the complexity of computations. Two directions which are relevant for this paper are the extension

* Corresponding author.
Email addresses: macucker@math.cityu.edu.hk (F. Cucker), Olivier.Bournez@loria.fr (O. Bournez), Paulin.De-Naurois@loria.fr (P.J. de Naurois), Jean-Yves.Marion@loria.fr (J.-Y. Marion).
[1] This work was partially supported by City University of Hong Kong SRG Grant 7001290.

0890-5401/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2005.07.005

of complexity theory to domains other than the set $\{0,1\}$ and the characterization of complexity classes in machine-independent terms.

A seminal paper for the first direction above is the one by Blum et al. [4], where a theory of computation and complexity over the real numbers was developed with the aim of capturing the features of numeric computations such as those done in numerical analysis. The setting of [4] actually considers computations over arbitrary rings R, the case $R = \mathbb{R}$ being their major case of interest and the case $R = \mathbb{Z}/2$ being the classical one. A further extension is carried out in [23] where computations over arbitrary structures are pursued.

Concerning the second direction above, a seminal result is the characterization by Fagin [11] of non-deterministic polynomial time relying on finite model theory. This result can be said to be *implicit* in the sense that it characterizes complexity classes in machine-independent terms. Other works followed along these lines [7,17,10,14,24]. Another approach giving implicit characterizations of complexity classes was initiated by Bellantoni and Cook [3] who characterized the class of functions computable in polynomial time as the smallest class of functions containing some basic functions and closed under some operations. This result is a subrecursive version of the classical result by Kleene characterizing Turing computable functions as those being recursive and is based on a purely syntactic distinction between different types of arguments (which avoids explicit upper bounds on computational resources). Further results characterizing function algebras in complexity classes appear in [19,20].

A natural research line is to combine the two directions above by looking for implicit characterization of complexity classes over the reals (or, more generally, over arbitrary structures). A first step towards this goal was given in [15] where the basis for an extension of Fagin's result to arbitrary structures was set, and in [13,9] where several complexity classes over $\mathbb{R}$ were characterized in this way. Then, in [5,6], we exhibited machine-independent characterizations of the classes of functions over an arbitrary structure computable in polynomial sequential or parallel time. This extended the classical characterizations in [3,21].

The goal of this paper is to further characterize other classes of computable functions over an arbitrary structure. We will do so for classes of function computable in polynomial time by machines making use of diverse forms of alternation.

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 $(\mathbb{R}, +, -, *, /, \leqslant)$, we have the following inclusions of complexity classes [8]

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

where an arrow means inclusion, $\text{EXP}_{\mathbb{R}}$ denotes exponential time, $\text{PEXP}_{\mathbb{R}}$ parallel exponential time, $\text{PH}_{\mathbb{R}}$ is the polynomial hierarchy, and $\text{PAT}_{\mathbb{R}}$ polynomial alternating time. In addition the two inclusions $\text{PAR}_{\mathbb{R}} \subset \text{PAT}_{\mathbb{R}}$ and $\text{PAR}_{\mathbb{R}} \subset \text{EXP}_{\mathbb{R}}$ are known to be strict. Inclusions in any direction between $\text{DPAT}_{\mathbb{R}}$ and $\text{PH}_{\mathbb{R}}$ are not known to hold. The same can be said of $\text{PAT}_{\mathbb{R}}$ and $\text{EXP}_{\mathbb{R}}$.

Our main results characterize, over an arbitrary structure $\mathcal{K}$, classes of functions corresponding to the different levels of the polynomial hierarchy and the digital polynomial hierarchy, polynomial alternating time and digital polynomial alternating time.

When restricted to classical complexity our characterizations, combined with our previous one for parallel polynomial time in [6], provide several new alternative characterizations of FPSPACE, the class of Boolean functions computable in polynomial space.

Furthermore, 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], may shed some light on the nature of choice operators.

Our results provide a background for designing methods deriving computational properties from programs along the lines of [16,18,22] in classical complexity.

The rest of this paper is structured as follows.

In Sections 2 and 3, we define basic notions and recall the characterizations of deterministic complexity classes from [6]. Then, in Section 4, we provide 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 result for digital polynomial alternating time in Section 7. In Section 8, we use another approach, related to parallelism, to characterize differently the digital polynomial alternating time class. Section 9 contains a technical result showing the equivalence between simple safe recursion and simultaneous safe recursion.

2. Arbitrary structures

Definition 1. A *structure* $\mathcal{K} = \left(\mathbb{K}, \{op_i\}_{i\in I}, rel_1, \ldots, rel_l, \mathbf{0}, \mathbf{1}\right)$ 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 arity greater than zero 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 the sequel) in the structure.

An example of structure is $\mathcal{K} = (\mathbb{R}, +, -, *, =, \leqslant, \{c_r\}_{r\in\mathbb{R}})$. The theory of complexity over the reals developed in [4] corresponds to computations over this structure. Another example, corresponding to classical complexity and computability theory, is $\mathcal{K} = \left(\{0,1\}, =, \mathbf{0}, \mathbf{1}\right)$.

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: $a.\bar{x}$ 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 $|\overline{w}|$.

Roughly speaking, a BSS machine over $\mathcal{K}$ is a RAM whose registers can store elements of the underlying structure, and that can, with unit cost, evaluate the basic operations op_i and test the basic relations $rel_1,\ldots,rel_l$ of the structure. Operations op_i of arity 0, i.e., constants, occur in a finite number in every machine. We assume the reader familiar with the notion of BSS machine. Detailed accounts can be found in [4]—for structures like real and complex numbers—or [23]—for considerations about more general structures.

In this setting of machines over $\mathcal{K}$ resources such as time, parallel time or alternating time can be considered allowing one to define several complexity classes. In addition, complete problems for many of these classes can be exhibited. In a previous paper [6], we provided machine independent characterizations of the class functions computable in polynomial time. Since our work here relies on this characterizations we next briefly recall it.

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-algebras, 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, S that of the safe arguments, and R the codomain of f. If all the arguments of f are of one kind, say safe, we will write $\emptyset$ in the place of N. Also, 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 $a \in \mathbb{K}, \overline{x}, \overline{x_1}, \overline{x_2} \in \mathbb{K}^*$

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

(ii) projections. For any $n \in \mathbb{N}, i \leqslant n$,

$$\mathsf{Pr}_i^n(; \overline{x_1}, \ldots, \overline{x_i}, \ldots, \overline{x_n}) = \overline{x_i}.$$

(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:

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

The equality relation will be denoted Equal.

(iv) a selector function

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

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

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

(2) *Safe recursion.* Let $h : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ and $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$. We define $f : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ by safe recursion as follows:

$$f(\epsilon, \bar{x};\, \bar{y}) = h(\bar{x};\, \bar{y})$$
$$f(a.\bar{z}, \bar{x};\, \bar{y}) = g(\bar{z}, \bar{x};\, f(\bar{z}, \bar{x};\, \bar{y}), \bar{y}).$$

Theorem 1 ([5,6]). *Over any structure $\mathcal{K} = \left(\mathbb{K}, \{op_i\}_{i \in I}, rel_1, \ldots, rel_l, \mathbf{0}, \mathbf{1}\right)$, a function is computed in polynomial time by a BSS machine if and only if it can be defined as a safe recursive function over $\mathcal{K}$.*

Sketch of proof. To prove Theorem 1, one needs first to show that any function computable in polynomial time by a BSS machine can be defined in $\mathrm{SR}_{\mathcal{K}}$. This is done in [5] in a rather straightforward way: Fix a BSS machine M and a polynomial time bound P for it. One can define safe recursive functions which, on input $(\mathbf{0}^t;\, \bar{x})$, describe the configuration reached by M on input $\bar{x}$ after t steps. One can also define a safe recursive funciton p such that $p(\bar{x};) = \mathbf{0}^{P(|\bar{x}|)}$. Composing these functions is enough to obtain a safe recursive function which produces on input $\bar{x}$ the same output as M after $P(|\bar{x}|)$ computation steps. Note that in [5] we were dealing with simultaneous safe recursion instead of simple safe recursion. Therefore we alse need to prove that simple safe recursion is enough. This is done in Proposition 1 in Section 9.

Remark that the same result is obtained directly in [6], where a P-uniform family of circuits of polynomial size is simulated by a simple safe recursive function.

For the other direction, one needs to show that any safe recursive function can be evaluated in deterministic polynomial time. This is done as follows: given any safe recursive function f, one shows by induction on the definition tree of f that the time needed to evaluate $f(\overline{x};\overline{y})$ is bounded by $p_f(|\overline{x}|)$, where p_f is a polynomial. □

Let Φ be a set and F a complexity class of functions (respectively, of sets). We denote by F^{Φ} the class of functions computable (respectively, sets decidable) by a machine in F with oracle Φ. When G is another complexity class, F^G denotes the class

$$\bigcup_{\Phi\in G} F^{\Phi}.$$

Definition 4. Given a function $\phi : \emptyset \times \mathbb{K}^* \to \mathbb{K}^*$, 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.

The following result is a relativization of our previous Theorem 1.

Theorem 2. *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 is in the class $\mathrm{FP}^{\Phi}_{\mathcal{K}}$ of functions computable in polynomial time with oracle Φ if and only if it can be defined in $\mathrm{SR}_{\mathcal{K}}(\phi)$.*

Proof. The proof is based upon that of Theorem 1 as it appears in [5]. The idea is to write a safe recursive function computing the output of a deterministic polynomial time BSS machine over $\mathcal{K}$. One just needs to add one case in the enumeration of all types of nodes of the machine: An oracle node $q \in \mathbb{N}$ calling for an oracle function ϕ has associated functions

$$\begin{aligned}
\mathcal{G}_i(;\overline{y_1},\overline{y_2}) &= \alpha^{q'}\\
\mathcal{H}_i(;\overline{y_1},\overline{y_2}) &= \overline{y_1}\\
\mathcal{I}_i(;\overline{y_1},\overline{y_2}) &= \phi(;\overline{y_2}).
\end{aligned}$$

Here, we recall from [5], $\alpha = \mathbf{0}$ and the word $\alpha^{q'}$ denotes the next node q' of q. The rest of the proof carries on without modification. Note that in [5] we were dealing with simultaneous safe recursion. To be fully formal we need to prove that simple safe recursion is enough. We do so in Proposition 1 in Section 9. □

We shall next introduce a technical lemma needed later in our proofs.

Lemma 1. *Assume $f : (\mathbb{K}^*)^2 \times \emptyset \to \mathbb{K}^*$ is in $\mathrm{SR}_{\mathcal{K}}(\phi)$. 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 $f' : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^* \in \mathrm{SR}_{\mathcal{K}}(\phi)$ such that*

$$f'(\overline{x};\overline{y}) = f(\overline{x},\overline{y};).$$

Proof. The idea is once again to follow the proof of Theorem 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 $p(|\overline{x}|)$. Theorem 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 PH $_{\mathcal{K}}$

4.1. Polynomial hierarchy over a structure $\mathcal{K}$

As in the classical setting, 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 $|\overline{y}| \leqslant p_B(|\overline{x}|)$ 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 $|\overline{y}| \leqslant P_B(|\overline{x}|)$, $(\overline{x}, \overline{y})$ is in B.

Definition 5. Let $\Sigma^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$ and, for $i \geqslant 1$, $\Sigma^i_{\mathcal{K}} = \mathrm{NP}_{\mathcal{K}}^{\Sigma^{i-1}_{\mathcal{K}}}$, $\Pi^i_{\mathcal{K}} = \mathrm{coNP}_{\mathcal{K}}^{\Sigma^{i-1}_{\mathcal{K}}}$.

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 is in $\mathrm{F}\Delta^i_{\mathcal{K}}$ if it computable in polynomial time by a machine over $\mathcal{K}$ which queries an oracle in $\Sigma^i_{\mathcal{K}}$. That is, $\mathrm{F}\Delta^i_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}} = \mathrm{FP}_{\mathcal{K}}^{\Pi^i_{\mathcal{K}}}$.

The *functional polynomial time hierarchy* over $\mathcal{K}$ is $\mathrm{FPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{F}\Delta^i_{\mathcal{K}}$.

Remark 1. 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 [4,23].

4.2. Safe recursion with predicative minimization

In the spirit of [2], we now introduce the notion of predicative minimization.

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

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

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 $\mathrm{RSR}_{\mathcal{K}}(\mathrm{F})$, is the smallest set of functions containing the basic safe functions and F, and closed under the following *restricted safe composition* scheme

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

where the h_i belong to $\mathrm{RSR}_{\mathcal{K}}(\mathrm{F})$ and g to $\mathrm{SR}_{\mathcal{K}}$, 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 $\mathrm{RSR}_{\mathcal{K}}(\mathrm{F})$ and g to $\mathrm{SR}_{\mathcal{K}}$. This implies that no function in $\mathrm{F}\backslash\mathrm{SR}_{\mathcal{K}}$ may be involved in the definition of g.

Definition 8. Assume F is a class of functions: a function f is in $\exists\mathrm{F}$ 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 $i \geqslant 0$.

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 $\mathrm{F}^i_{\mathcal{K}}$, there are at most i nested predicative minimizations. This bound does not depend on the arguments of f. In other words, there exist h in $\mathrm{SR}_{\mathcal{K}}$ and $f_1,\ldots,f_n$ in $\mathrm{F}^{i-1}_{\mathcal{K}}$, such that, for all $\mathbf{x} = (\overline{x_1},\ldots,\overline{x_l})$,*

$$f(\mathbf{x};) = h(\mathbf{x};\exists\overline{z_1}(f_1(\mathbf{x};\overline{z_1})),\ldots,\exists\overline{z_n}(f_n(\mathbf{x};\overline{z_n}))).$$

We denote this as a normal form for f.

Proof. By induction on i and on the definition of f:

- If $i = 0$, this normal form holds.
- If f is a basic safe function, this normal form holds as well.
- If f is defined with restricted safe composition as in Definition 7, g belongs to $\mathrm{SR}_{\mathcal{K}}$, and we may apply the induction hypothesis on the h_i.
- If f is defined with restricted safe recursion as in Definition 7 define:

$$\begin{aligned} f'(\epsilon,\overline{x};\overline{y},\overline{t}) &= \overline{t}\\ f'(a.\overline{z},\overline{x};\overline{y},\overline{t}) &= g(\overline{z},\overline{x};f'(\overline{z},\overline{x};\overline{y},\overline{t}),\overline{y}) \end{aligned}$$

  f' belongs to $\mathrm{SR}_{\mathcal{K}}$, $f'(\overline{z},\overline{x};\overline{y},h(\overline{x};\overline{y})) = f(\overline{z},\overline{x};\overline{y})$, and we may apply the induction hypothesis on h.
- If f is defined with predicative minimization $f(\overline{x};\overline{a}) = \exists\overline{b}(h(\overline{x};\overline{a},\overline{b}))$, by definition of $\mathrm{F}^i_{\mathcal{K}}$ h belongs to $\mathrm{F}^{i-1}_{\mathcal{K}}$ and this is also a normal form.

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 $\mathrm{F}^i_{\mathcal{K}}$.*

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 Theorem 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 $\Phi \in \Sigma^i_{\mathcal{K}} = \mathrm{NP}_{\mathcal{K}}^{\Sigma^{i-1}_{\mathcal{K}}}$ 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 2: 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 establish that ψ belongs to $\mathrm{F}^{i-1}_{\mathcal{K}}$. Then, we 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* $\mathrm{F}^i_{\mathcal{K}}$. *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 Theorem 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,

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

By induction hypothesis, the functions $f_1, \ldots, f_n$ belong to $\mathrm{F}\Delta^{i-1}_{\mathcal{K}}$. The corresponding decision problems $f_1(\mathbf{x}; \overline{z_1}) = \mathbf{0}, \ldots, f_n(\mathbf{x}; \overline{z_n}) = \mathbf{0}$ belong to $\mathrm{P}_{\mathcal{K}}^{\Sigma^{i-1}_{\mathcal{K}}} = \Sigma^{i-1}_{\mathcal{K}}$. 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 $g_j(\mathbf{x};) = \boldsymbol{\ni}\overline{z_j}(f_j(\mathbf{x}; \overline{z_j}))$ for $1 \leqslant j \leqslant n$. 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: $f \in \mathrm{FP}_{\mathcal{K}}^{\Sigma^i_{\mathcal{K}}} = \mathrm{F}\Delta^i_{\mathcal{K}}$. □

Lemmas 3 and 4 yield our first main characterization.

Theorem 3. *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* $\mathrm{F}^i_{\mathcal{K}}$.

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

Corollary 1. *A decision problem over* $\mathcal{K}$ *belongs to* $\mathrm{PH}_{\mathcal{K}}$ *if and only if its characteristic function is defined in* $\boldsymbol{\ni}\mathrm{PH}_{\mathcal{K}}$.

5. A characterization of DPH $_{\mathcal{K}}$

5.1. *Digital polynomial hierarchy over a structure $\mathcal{K}$*

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 $\overline{x} \in \mathbb{K}^*$,

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

Let $\mathrm{D}\Sigma^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$ and, for $i \geqslant 1$, $\mathrm{D}\Sigma^i_{\mathcal{K}} = \mathrm{DNP}_{\mathcal{K}}^{\mathrm{D}\Sigma^{i-1}_{\mathcal{K}}}$, $\mathrm{D}\Pi^i_{\mathcal{K}} = \mathrm{coDNP}_{\mathcal{K}}^{\mathrm{D}\Sigma^{i-1}_{\mathcal{K}}}$.

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 is in $\mathrm{DF}\Delta^i_{\mathcal{K}}$ if it computable in polynomial time by a machine over $\mathcal{K}$ which queries an oracle in $\mathrm{D}\Sigma^i_{\mathcal{K}}$. That is, $\mathrm{DF}\Delta^i_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{D}\Sigma^i_{\mathcal{K}}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{D}\Pi^i_{\mathcal{K}}}$.

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.

5.2. *Safe recursion with digital predicative minimization*

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:

$$f(\overline{x}; \overline{a}) = \boldsymbol{\ni}_{\mathrm{D}}\overline{b}(h(\overline{x}; \overline{a},\overline{b})) = \begin{cases} \mathbf{1} & \text{if there exists } \overline{b} \in \{\mathbf{0},\mathbf{1}\}^* \text{ such that } h(\overline{x}; \overline{a},\overline{b}) = \mathbf{0} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

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

We define by induction the following sets:

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

We denote by $\boldsymbol{\ni}_{\mathrm{D}}\mathrm{PH}_{\mathcal{K}}$ 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 3, *mutatis mutandis*, yields the following results.

Theorem 4. *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 2. *A decision problem over $\mathcal{K}$ belongs to* $\mathrm{DPH}_{\mathcal{K}}$ *if and only if its characteristic function is defined in* $\boldsymbol{\ni}_{\mathrm{D}}\mathrm{DPH}_{\mathcal{K}}$.

Example 2. 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 carry a total value at least V with total weight at most W? Assume by Theorem 1 that the safe recursive function $v(\overline{x};\overline{y})$ decides wether, 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 naturally yields a characterization of the classical polynomial hierarchy alternative to the one found in [2].

Corollary 3. *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 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 $\overline{x} \in \mathbb{K}^*$,

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

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

When $\mathcal{K}$ is the 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:

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

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 substitutions* as follows:

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

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.

Remark 2. In Definition 14 we have used a predicative substitution which checks the existence of a witness in $\mathbb{K}$. We could have equally used a predicative minimization which, as in Definition 6, checks the existence of a witness in $\mathbb{K}^*$. We have chosen the former for the sake of simplicity.

Theorem 5. *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 $\overline{x} \in \mathbb{K}^*$,

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

Theorem 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, where all tests can be easily done with composition and the Select function,

$$F_G(\epsilon,\overline{x};\ \overline{u}) = f_M(\overline{x};\ \overline{u})$$

$$F_G(a.\overline{z},\overline{x};\ \overline{u}) = \begin{cases} \ni^{[1]}c(F_G(\overline{z},\overline{x};\ c.\overline{u})) & \text{if } \mathsf{hd}(;\ \overline{z}) = \mathbf{1}\\ \mathbf{0} & \text{if } \mathsf{hd}(;\ \overline{z}) = \mathbf{0} \text{ and } \ni^{[1]}c(F_G(\overline{z},\overline{x};\ c.\overline{u})) = \mathbf{1}\\ \mathbf{1} & \text{if } \mathsf{hd}(;\ \overline{z}) = \mathbf{0} \text{ and } \ni^{[1]}c(F_G(\overline{z},\overline{x};\ c.\overline{u})) = \mathbf{0}\\ f_M(\overline{x};\ \overline{u}) & \text{otherwise.}\end{cases}$$

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 Theorem 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. By Theorem 2, 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 substitutions, as in Definition 14. In this case, $f(a.\overline{z},\overline{x};\ \overline{u},\overline{y})$ equals $\mathbf{1}$ if and only if

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

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

$$f(a.\overline{z},\overline{x};\overline{u},\overline{y}) = \mathbf{1} \quad \text{iff } Q_1c_1 \in \mathbb{K},\ldots,Q_{|\overline{z}|+1}c_{|\overline{z}|+1} \in \mathbb{K}\ h(\overline{x};c_1.\ldots.c_{|\overline{z}|+1}.\overline{u},\overline{y})\ r\ \mathbf{0}.$$

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 $\mathbf{DPAT}_{\mathcal{K}}$

The class $\mathrm{DPAT}_{\mathcal{K}}$ is defined similarly 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*,

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

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 substitutions* 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 substitutions.

Again, the proof of Theorem 5 yields, *mutatis mutandis*, the following result.

Theorem 6. *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 4. *A set* $S \subset \{0,1\}^*$ *is in* PSPACE *if and only if its characteristic function can be defined in* $\ni^{[1]}\mathrm{DPAT}_{\{\mathbf{0},\mathbf{1}\}}$.

8. An alternative characterization of DPAT$_{\mathcal{K}}$

8.1. *Safe recursive functions with substitutions*

In [6] we gave a characterization of the class of functions computable in parallel polynomial time in terms of a constructor called safe recursion with substitutions.

Definition 19. The set of functions defined with *safe recursion with substitutions* over $\mathcal{K}$ 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 safe composition and the following *Safe recursion with substitutions* scheme.

Assume $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^{l+1} \to \mathbb{K}^*$, and $\sigma_j : \emptyset \times \mathbb{K}^* \to \mathbb{K}^*$ for $0 < j \leqslant l$ are given functions. The function $f : (\mathbb{K})^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ is defined by safe recursion with substitutions as follows:

$$\begin{aligned} &f(\epsilon, \overline{x}; \overline{u}, \overline{y}), = h(\overline{x}; \overline{u}, \overline{y})\\ &f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) = g(\overline{z}, \overline{x}; f(\overline{z}, \overline{x}; \sigma_1(; \overline{u}), \overline{y}), \ldots, f(\overline{z}, \overline{x}; \sigma_l(; \overline{u}), \overline{y}), \overline{y}). \end{aligned}$$

The functions σ_j are called *substitution functions.*

Theorem 7 ([6]). *Over any structure $\mathcal{K} = \left(\mathbb{K}, \{op_i\}_{i\in I}, rel_1, \ldots, rel_l, \mathbf{0}, \mathbf{1}\right)$, a function is computed in parallel polynomial time by a (parallel) BSS machine if and only if it is defined as a safe recursive function with substitutions over $\mathcal{K}$.*

8.2. *Safe recursive functions with digital substitutions*

When restricted to finite structures, DPAT$_{\mathcal{K}}$ coincides with PAR$_{\mathcal{K}}$ and with PSPACE. However, when $\mathcal{K}$ is arbitrary, we only have the inclusion DPAT$_{\mathcal{K}}$ $\subset$ PAR$_{\mathcal{K}}$. Based on our previous characterization of PAR$_{\mathcal{K}}$, some small restrictions on the type of the functions involved in the recursion scheme yield another characterization of DPAT$_{\mathcal{K}}$.

Definition 20. We call *pseudo logical function* any function in the closure of operations of arity 0 (constants), projections and the selector function Select under the application of safe composition.

Since no recursion and no tl function is involved in the definition of a pseudo logical function, its output depends only on the value of the first letter of its arguments, more precisely, on whether these are **1** or not since no relation is allowed either.

Definition 21. Assume $h : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ is a given function, $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ is a pseudo logical function, and $\sigma_1, \sigma_2 : \emptyset \times \mathbb{K}^* \to \mathbb{K}^*$ are safe recursive functions. Function $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ can then be defined by *safe recursion with digital substitutions*:

$$\begin{aligned} &f(\epsilon, \overline{x}; \overline{u}) = h(\overline{x}; \overline{u})\\ &f(a.\overline{z}, \overline{x}; \overline{u}) = g(\overline{z}, \overline{x}; f(\overline{z}, \overline{x}; \sigma_1(; \overline{u})), f(\overline{z}, \overline{x}; \sigma_2(; \overline{u}))). \end{aligned}$$

We define the set of *safe recursive functions with digital substitutions* to be the closure of the basic safe functions under the application of safe composition, safe recursion and safe recursion with digital substitutions.

Theorem 8. *A function is in* $\mathrm{DFPAT}_{\mathcal{K}}$ *if and only if it can be defined as a safe recursive functions with digital substitutions over* $\mathcal{K}$.

Proof. Let F be a function in $\mathrm{DFPAT}_{\mathcal{K}}$, and denote by G the associate oracle in $\mathrm{DPAT}_{\mathcal{K}}$. There exists a polynomial time BSS machine M over $\mathcal{K}$, and a polynomial function $q : \mathbb{N} \to \mathbb{N}$ such that, for any $\overline{x} \in \mathbb{K}^*$, the following propositions are equivalent:

- (i): $\overline{x} \in G$
- (ii): $\exists b_1 \in \{\mathbf{0},\mathbf{1}\}\ \forall c_1 \in \{\mathbf{0},\mathbf{1}\}\ \ldots\ \exists b_{q(|\overline{x}|)} \in \{\mathbf{0},\mathbf{1}\}\ \forall c_{q(|\overline{x}|)} \in \{\mathbf{0},\mathbf{1}\}$ M accepts $(\overline{x}, b_1.c_1 \ldots b_{q(|\overline{x}|)}.c_{q(|\overline{x}|)})$.

Theorem 1 ensures 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 $f_M(\overline{x};\overline{y}) = \mathbf{1}$. Moreover, define $g_q : \mathbb{K}^* \times \emptyset \to \mathbb{K}^*$ such that $g_q(\overline{x};) = (\mathbf{1}.\mathbf{0})^{q(|\overline{x}|)}$. The existence of such a g_q is once again given by Theorem 1. 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.

Define now the following function,

$$F_G(\epsilon,\overline{x};\overline{u}) = f_M(\overline{x};\overline{u})$$
$$F_G(a.\overline{z},\overline{x};\overline{u}) = \begin{cases} F_G(\overline{z},\overline{x};\mathbf{1}.\overline{u}) = \mathbf{1} \vee F_G(\overline{z},\overline{x};\mathbf{0}.\overline{u}) = \mathbf{1} & \text{if } \mathsf{hd}(;\overline{z}) = \mathbf{1} \\ F_G(\overline{z},\overline{x};\mathbf{1}.\overline{u}) = \mathbf{1} \wedge F_G(\overline{z},\overline{x};\mathbf{0}.\overline{u}) = \mathbf{1} & \text{otherwise.} \end{cases}$$

The formal definition with safe recursion with digital substitutions of F_G is as follows:

$$F_G(\epsilon,\overline{x};\overline{u}) = f_M(\overline{x};\overline{u})$$
$$\begin{aligned} F_G(a.\overline{z},\overline{x};\overline{u}) = \mathsf{Select}\Big(&;\mathsf{hd}(;\overline{z}), \mathsf{Select}\Big(;F_G(\overline{z},\overline{x};\mathsf{cons}(;\mathbf{1},\overline{u})),\mathbf{1},\ F_G(\overline{z},\overline{x};\mathsf{cons}(;\mathbf{0},\overline{u}))\Big),\\ &\mathsf{Select}(;F_G(\overline{z},\overline{x};\mathsf{cons}(;\mathbf{1},\overline{u})),F_G(\overline{z},\overline{x};\mathsf{cons}(;\mathbf{0},\overline{u})),\mathbf{0})). \end{aligned}$$

It is clear from the definition that $F_G(\mathsf{cons}(\mathbf{1},g_S(\overline{x};);),\overline{x};\mathbf{0})$ decides whether $\overline{x}$ belongs to G.

It follows that F belongs to the set of safe recursive functions with digital substitutions.

The other direction of the proof is done by induction on the definition of F. The only critical case is when F is defined with safe recursion with digital substitutions:

$$F(\epsilon,\overline{x};\overline{u}) = h(\overline{x};\overline{u})$$
$$F(a.\overline{z},\overline{x};\overline{u}) = g(\overline{z},\overline{x};F(\overline{z},\overline{x};\sigma_1(;\overline{u})),F(\overline{z},\overline{x};\sigma_2(;\overline{u}))). \quad \square$$

In this case the result follows from the following lemma.

Lemma 5. *The relation* $F(a.\overline{z},\overline{x};\overline{u}) = \mathbf{1}$ *can be reduced in polynomial time to a decision problem in* $(\mathrm{D}\Sigma_{\mathcal{K}}^{2|\overline{z}|+2})^H$ *where* H *is an oracle deciding* $h(\overline{y};\overline{v}) = \mathbf{1}$.

Proof. By induction on $|\overline{z}|$. For $\overline{z} = \epsilon$, it is a consequence of Theorem 1.

Assume $\overline{z} \neq \epsilon$, and define:

$$a_1 = \begin{cases} \mathbf{1} & \text{if } \mathsf{hd}(;F(\overline{z},\overline{x};\sigma_1(;\overline{u}))) = \mathbf{1} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$a_2 = \begin{cases} \mathbf{1} & \text{if } \mathsf{hd}(;F(\overline{z},\overline{x};\sigma_2(;\overline{u}))) = \mathbf{1} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Since g is a pseudo logical function, it is computable in constant time, and the value of $F(a.\overline{z},\overline{x};\overline{u}) = g(\overline{z},\overline{x};F(\overline{z},\overline{x};\sigma_1(;\overline{u})),F(\overline{z},\overline{x};\sigma_2(;\overline{u})))$ depends only on the relations $\mathsf{hd}(;F(\overline{z},\overline{x};\sigma_1(;\overline{u}))) = \mathbf{1}$ and $\mathsf{hd}(;F(\overline{z},\overline{x};\sigma_2(;\overline{u}))) = \mathbf{1}$. Thus,

$$g(\overline{z},\overline{x};F(\overline{z},\overline{x};\sigma_1(;\overline{u})),F(\overline{z},\overline{x};\sigma_2(;\overline{u}))) = g(\overline{z},\overline{x};a_1,a_2).$$

Define

$$F'(\overline{z},\overline{x};\overline{c},\overline{u}) = \begin{cases} F(\overline{z},\overline{x};\sigma_1(;\overline{u})) & \text{if } \mathsf{hd}(;\overline{c}) = \mathbf{1} \\ F(\overline{z},\overline{x};\sigma_2(;\overline{u})) & \text{otherwise.} \end{cases}$$

Consider now the four possible values for (a_1,a_2). The relation $F(a.\overline{z},\overline{x};\overline{u}) = \mathbf{1}$ is given by:

- $(a_1 = \mathbf{1}, a_2 = \mathbf{1})$: $\forall c \in \{\mathbf{1},\mathbf{0}\}\ (g(\overline{z},\overline{x};a_1,a_2) = \mathbf{1}) \wedge (F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1})$
- $(a_1 = \mathbf{1}, a_2 = \mathbf{0})$: $\forall c \in \{\mathbf{1},\mathbf{0}\}\ (g(\overline{z},\overline{x};a_1,a_2) = \mathbf{1}) \wedge (c = \mathbf{1} \Leftrightarrow F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1})$
- $(a_1 = \mathbf{0}, a_2 = \mathbf{1})$: $\forall c \in \{\mathbf{1},\mathbf{0}\}\ (g(\overline{z},\overline{x};a_1,a_2) = \mathbf{1}) \wedge (c = \mathbf{0} \Leftrightarrow F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1})$
- $(a_1 = \mathbf{0}, a_2 = \mathbf{0})$: $\forall c \in \{\mathbf{1},\mathbf{0}\}\ (g(\overline{z},\overline{x};a_1,a_2) = \mathbf{1}) \wedge (F'(\overline{z},\overline{x};c,\overline{u}) \neq \mathbf{1})$.

Note that the logical operations "$\wedge, \vee, \Leftrightarrow$" can be easily computed with the basic function Select, projections and safe composition.

The relation $F(a.\overline{z},\overline{x};\overline{u}) = \mathbf{1}$ is therefore given by

$$\begin{aligned} \exists a_1,a_2 \in \{\mathbf{0},\mathbf{1}\}, \forall c \in \{\mathbf{0},\mathbf{1}\} \quad & (g(\overline{z},\overline{x};a_1,a_2) = \mathbf{1}) \\ \bigwedge & [((a_1 = \mathbf{1} \wedge a_2 = \mathbf{1}) \wedge (F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1})) \\ & \vee((a_1 = \mathbf{1} \wedge a_2 = \mathbf{0}) \wedge (c = \mathbf{1} \Leftrightarrow F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1})) \\ & \vee((a_1 = \mathbf{0} \wedge a_2 = \mathbf{1}) \wedge (c = \mathbf{0} \Leftrightarrow F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1})) \\ & \vee((a_1 = \mathbf{0} \wedge a_2 = \mathbf{0}) \wedge (F'(\overline{z},\overline{x};c,\overline{u}) \neq \mathbf{1}))]. \end{aligned}$$

By induction hypothesis $F(\overline{z},\overline{x};\overline{u}) = \mathbf{1}$ can be reduced to a decision problem in $(\mathrm{D}\Sigma_{\mathcal{K}}^{2|\overline{z}|})^H$, and therefore $F'(\overline{z},\overline{x};c,\overline{u}) = \mathbf{1}$ can also be reduced to the same problem, which ends the proof. □

Remark 3. When $\mathcal{K}$ is a finite structure, i.e., when considering classical complexity, this characterization coincides with our previous characterization of $\mathrm{PAR}_{\mathcal{K}}$ in [6], and captures PSPACE.

9. Safe recursion versus simultaneous safe recursion

This section is devoted to the the proof that safe recursion yields the same algebras of function that simultaneous safe recursion.

Proposition 1. *The set of safe recursive functions over $\mathcal{K}$ and the set of simultaneous safe recursive functions of [6] over $\mathcal{K}$ coincide.*

Proof. For the sake of simplicity, we prove this result for a double recursion scheme. The generalization to an arbitrary simultaneous recursion scheme stems on the same principles. We only need to prove that safe recursive functions can compute double safe recursive functions. The other direction is trivial. We proceed by induction on the definition tree. For basic safe functions, the result is trivial. For a function defined with safe composition, the induction hypothesis gives the result. Assume that $f_1, f_2 : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ are defined by a double recursion scheme as follows:

$$\begin{aligned} f_1(\epsilon,\overline{x};\,\overline{y}) &= h_1(\overline{x};\,\overline{y}) \\ f_2(\epsilon,\overline{x};\,\overline{y}) &= h_2(\overline{x};\,\overline{y}) \\ f_1(a.\overline{z},\overline{x};\,\overline{y}) &= g_1(\overline{z},\overline{x};\,f_1(\overline{z},\overline{x};\,\overline{y}),f_2(\overline{z},\overline{x};\,\overline{y}),\overline{y}) \\ f_2(a.\overline{z},\overline{x};\,\overline{y}) &= g_2(\overline{z},\overline{x};\,f_1(\overline{z},\overline{x};\,\overline{y}),f_2(\overline{z},\overline{x};\,\overline{y}),\overline{y}). \end{aligned}$$

The induction hypothesis allows us to assume that functions h_1, h_2, g_1, g_2 are safe recursive. Assume moreover that they respect the following *homogeneous length hypothesis*: there exists a safe recursive function $L : (\mathbb{K}^*)^2 \to \{\mathbf{0}\}^*$ such that, for all $\overline{z},\overline{x},\overline{r_1},\overline{r_2},\overline{y} \in \mathbb{K}^*$,

$$\begin{aligned} |g_1(\overline{z},\overline{x};\,\overline{r_1},\overline{r_2},\overline{y})| &= |L(\overline{z},\overline{x};\,)| \\ |g_2(\overline{z},\overline{x};\,\overline{r_1},\overline{r_2},\overline{y})| &= |L(\overline{z},\overline{x};\,)| \\ |h_1(\overline{x};\,\overline{y})| &= |L(\epsilon,\overline{x};\,)| \\ |h_2(\overline{x};\,\overline{y})| &= |L(\epsilon,\overline{x};\,)|. \end{aligned}$$

This hypothesis allows us to define a function $F : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ such that:

$$F(\overline{z},\overline{x};\,\overline{y}) = f_1(\overline{z},\overline{x};\,\overline{y}).f_2(\overline{z},\overline{x};\,\overline{y}).$$

Let us define a safe recursive function p such that, forall $\overline{x},\overline{y},\overline{m} \in \mathbb{K}^*$,

$$\text{if} \quad \overline{m} = \overline{m_x}.\overline{m_y}.\overline{m_z} \quad \text{and} \quad |\overline{m_x}| = |\overline{x}|, |\overline{m_y}| = |\overline{y}| \quad \text{then} \quad p(\overline{x},\overline{y};\,\overline{m}) = \overline{m_y}.$$

This function realizes some kind of projection on $\overline{m}$, and is formally defined with safe recursion as follows:

$$\begin{aligned} p(\epsilon,\overline{y};\,\overline{m}) &= q(\overline{y};\,\overline{m}) \\ p(a.\overline{x},\overline{y};\,\overline{m}) &= \mathsf{tl}(;\,p(\overline{x},\overline{y};\,\overline{m})) \\ q(\overline{y};\,\overline{m}) &= q'(\overline{y};\,q'(\overline{y};\,\overline{m})) \end{aligned}$$

$$\begin{aligned}
q'(\epsilon;\overline{m}) &= \epsilon\\
q'(a.\overline{y};\overline{m}) &= \mathsf{cons}(;\,\mathsf{hd}(;\,\mathsf{Tl}(\overline{y};\overline{m})), q'(\overline{y};\overline{m}))\\
\mathsf{Tl}(\epsilon;\overline{m}) &= \overline{m}\\
\mathsf{Tl}(a.\overline{y};\overline{m}) &= \mathsf{tl}(;\,\mathsf{Tl}(\overline{y};\overline{m})).
\end{aligned}$$

Let us also define a generalized concatenation Cons with safe recursion:

$$\begin{aligned}
\mathsf{Cons}(\overline{z};\overline{x},\overline{y}) &= \mathsf{Cons}'(\overline{z};q'(\overline{z};\overline{x}),\overline{y})\\
\mathsf{Cons}'(\epsilon;\overline{x},\overline{y}) &= \overline{y}\\
\mathsf{Cons}'(a.\overline{z};\overline{x},\overline{y}) &= \mathsf{cons}(;\,\mathsf{hd}(;\,\mathsf{Tl}(\overline{z};\overline{x})),\mathsf{Cons}'(\overline{z};\overline{x},\overline{y})).
\end{aligned}$$

$\mathsf{Cons}(\overline{z};\overline{x},\overline{y})$ returns $\overline{x}.\overline{y}$ provided that $|\overline{z}| \geqslant |\overline{x}|$. Let us detail now how we can formally define F with safe recursion. Since h_1, h_2, g_1 and g_2 respect the homogeneous length hypothesis, we have:

$$\begin{aligned}
|h_1(\overline{x};\overline{y})| &= |L(\epsilon,\overline{x})|\\
|h_2(\overline{x};\overline{y})| &= |L(\epsilon,\overline{x})|\\
|f_1(\overline{z},\overline{x};\overline{y})| &= |L(\mathsf{tl}'(\overline{z};),\overline{x};)| \text{ with } \mathsf{tl}'(\overline{z};) = \mathsf{tl}(;\overline{z})\\
|f_2(\overline{z},\overline{x};\overline{y})| &= |L(\mathsf{tl}'(\overline{z};),\overline{x};)|.
\end{aligned}$$

Thus, the definition of F is:

$$\begin{aligned}
F(\epsilon,\overline{x};\overline{y}) &= \mathsf{Cons}(L(\epsilon,\overline{x};);\,h_1(\overline{x};\overline{y}),h_2(\overline{x};\overline{y}))\\
F(a.\overline{z},\overline{x};\overline{y}) &= \mathsf{Cons}(L(\overline{z},\overline{x};);\\
&\qquad g_1\big(\overline{z},\overline{x};\,p(\epsilon,L(\mathsf{tl}'(\overline{z};),\overline{x};),F(\overline{z},\overline{x};\overline{y})),\\
&\qquad p(L(\mathsf{tl}'(\overline{z};),\overline{x};),L(\mathsf{tl}'(\overline{z};),\overline{x}),F(\overline{z},\overline{x};\overline{y})),\overline{y}\big),\\
&\qquad g_2\big(\overline{z},\overline{x};\,p(\epsilon,L(\mathsf{tl}'(\overline{z};),\overline{x};),F(\overline{z},\overline{x};\overline{y})),\\
&\qquad p(L(\mathsf{tl}'(\overline{z};),\overline{x};),L(\mathsf{tl}'(\overline{z};),\overline{x}),F(\overline{z},\overline{x};\overline{y})\big),\overline{y})).
\end{aligned}$$

The rest of the proof follows from the following lemma. □

Lemma 6. *For any safe recursive functions $f_1, f_2 : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ defined with double safe recursion, there exist safe recursive functions $f_1', f_2' : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ defined with double safe recursion with respect to the homogeneous length hypothesis and such that, for all $\overline{z},\overline{x},\overline{y} \in \mathbb{K}^*$, $f_1'(\overline{z},\overline{x};\overline{y}) = f_1(\overline{z},\overline{x};\overline{y})$ and $f_2'(\overline{z},\overline{x};\overline{y}) = f_2(\overline{z},\overline{x};\overline{y})$.*

Proof. Consider $f_1, f_2 : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ defined by a double recursion scheme as follows:

$$\begin{aligned}
f_1(\epsilon,\overline{x};\overline{y}) &= h_1(\overline{x};\overline{y})\\
f_2(\epsilon,\overline{x};\overline{y}) &= h_2(\overline{x};\overline{y})\\
f_1(a.\overline{z},\overline{x};\overline{y}) &= g_1(\overline{z},\overline{x};\,f_1(\overline{z},\overline{x};\overline{y}),f_2(\overline{z},\overline{x};\overline{y}),\overline{y})\\
f_2(a.\overline{z},\overline{x};\overline{y}) &= g_2(\overline{z},\overline{x};\,f_1(\overline{z},\overline{x};\overline{y}),f_2(\overline{z},\overline{x};\overline{y}),\overline{y}).
\end{aligned}$$

Since they are computable in polynomial time, there exists a safe recursive function $B: (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \{0\}^*$ such that:

$$\begin{aligned}
&\forall \overline{z}, \overline{x}, \overline{r_1}, \overline{r_2}\, \overline{y} \in \mathbb{K}^*\\
&\quad |g_1(\overline{z},\overline{x};\overline{r_1},\overline{r_2},\overline{y})| < |B(\overline{z},\overline{x};)|\\
&\quad |g_2(\overline{z},\overline{x};\overline{r_1},\overline{r_2},\overline{y})| < |B(\overline{z},\overline{x};\,)|\\
&\quad\quad |h_1(\overline{x};\overline{y})| < |B(\epsilon,\overline{x};\,)|\\
&\quad\quad |h_2(\overline{x};\overline{y})| < |B(\epsilon,\overline{x};\,)|.
\end{aligned}$$

Let us define $C: \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ such that:

$$C(\overline{b};\overline{m}) = \mathbf{1}^{|\overline{m}|}.\mathbf{0}^{2(|\overline{b}|-|\overline{m}|)}.\overline{m}$$

and $C^{-1}: \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ such that:

$$C^{-1}(\overline{b}; C(\overline{b};\overline{m})) = \overline{m}.$$

Note that $|C(\overline{b};\overline{m})| = 2|\overline{b}|$. These functions are formally defined as follows:

$$\begin{aligned}
C(\overline{b};\overline{m}) &= \mathsf{Cons}(\overline{b}; q''(\overline{b};\overline{m}), \mathsf{Cons}(\overline{b}; q'''(\overline{b};\overline{m}), \mathsf{Cons}(\overline{b}; q'''(\overline{b};\overline{m}),\overline{m})))\\
q''(\epsilon;\overline{m}) &= \epsilon\\
q''(a.\overline{z};\overline{m}) &= \begin{cases} q''(\overline{z};\overline{m}) & \text{if } \mathsf{hd}(;\mathsf{Tl}(\overline{z};\overline{m})) = \epsilon\\ \mathsf{cons}(;\mathbf{1}, q''(\overline{z};\overline{m})) & \text{otherwise}\end{cases}\\
q'''(\epsilon,\overline{m}) &= \epsilon\\
q'''(a.\overline{z};\overline{m}) &= \begin{cases} \mathsf{cons}(;\mathbf{0}, q'''(\overline{z};\overline{m})) & \text{if } \mathsf{Tl}(\overline{z};\overline{m}) = \epsilon\\ q'''(\overline{z};\overline{m}) & \text{otherwise}\end{cases}\\
C^{-1}(\overline{b},\overline{m'}) &= q'(\overline{b}; T(\overline{b},\overline{b};\overline{m'}))\\
T(\epsilon,\overline{l};\overline{m'}) &= \epsilon\\
T(a.\overline{z},\overline{l};\overline{m'}) &= \begin{cases} \mathsf{cons}(;\mathsf{hd}(;\mathsf{Tl}(\overline{z};\overline{m'})), T(\overline{z},\overline{l};\overline{m'})) & \text{if } \mathsf{hd}(;\mathsf{Tl}(\mathsf{Sub}(\overline{z},\overline{l};\,);\overline{m'})) = \mathbf{1}\\ T(\overline{z},\overline{l};\overline{m'}) & \text{otherwise}\end{cases}\\
\mathsf{Sub}(\overline{z},\overline{l};\,) &= \mathsf{tl}(;\mathsf{Tl}(\overline{z};\overline{l})).
\end{aligned}$$

It is obvious that the matching cases in these definition can be formally defined with the test function Select. These functions are therefore safe recursive. Define now by double safe recursion:

$$\begin{aligned}
f_1''(\epsilon,\overline{x};\overline{y}) &= C\left(B(\epsilon,\overline{x};); h_1(\overline{x};\overline{y})\right)\\
f_1''(a.\overline{z},\overline{x};\overline{y}) &= C\Big(B(\overline{z},\overline{x};); g_1\Big(\overline{z},\overline{x}; C^{-1}(B(\mathsf{tl}'(\overline{z};),\overline{x};) f_1''(\overline{z},\overline{x};\overline{y})),\\
&\qquad C^{-1}(B(\mathsf{tl}'(\overline{z};),\overline{x};) f_2''(\overline{z},\overline{x};\overline{y})),\overline{y}\Big)\Big)
\end{aligned}$$

$$f_2''(\epsilon,\overline{x};\overline{y}) = C\,(B(\epsilon,\overline{x};);\, h_2(\overline{x};\overline{y}))$$
$$f_1''(a.\overline{z},\overline{x};\overline{y}) = C\left(B(\overline{z},\overline{x};);\, g_2\left(\overline{z},\overline{x};\, C^{-1}(B(\mathrm{tl}'(\overline{z};),\overline{x};) f_1''(\overline{z},\overline{x};\overline{y})),\right.\right.$$
$$\left.\left. C^{-1}(B(\mathrm{tl}'(\overline{z};),\overline{x};) f_2''(\overline{z},\overline{x};\overline{y})),\overline{y}\right)\right).$$

Then f_1' and f_2' are given by

$$f_1'(\overline{z},\overline{x};\overline{y}) = C^{-1}(B(\overline{z},\overline{x};);\, f_1''(\overline{z},\overline{x}))$$
$$f_2'(\overline{z},\overline{x};\overline{y}) = C^{-1}(B(\overline{z},\overline{x};);\, f_2''(\overline{z},\overline{x})). \qquad \square$$

References

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

[2] S. Bellantoni, Predicative recursion and the polytime hiearchy, in: P. Clote, J.B. Remmel (Eds.), Feasible Mathematics II Perspectives in Computer Science, Birkhaüser, 1994.

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

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

[5] O. Bournez, F. Cucker, P. Jacobé de Naurois, J.-Y. Marion, Computability over an arbitrary structure: sequential and parallel polynomial time, in: A.D. Gordon (Ed.), Foundations of Software Science and Computational Structures, 6th International Conference (FOSSACS'2003), Lecture Notes in Computer Sciences, vol. 2620, Springer, Berlin, 2003, pp. 185–199..

[6] O. Bournez, F. Cucker, P. Jacobé de Naurois, J.-Y. Marion, Implicit complexity over an arbitrary structure: sequential and parallel polynomial time, Journal of Logic and Computation 15 (2005) 41–58.

[7] P. Clote, Computational models and function algebras, In: D. Leivant, (Ed.), LCC94́, Lecture Notes in Computer Sciences, vol. 960, Springer, 1995, pp. 98–130.

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

[9] F. Cucker, K. Meer, Logics which capture complexity classes over the reals, Journal of Symbolic Logic 64 (1999) 363–390.

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

[11] R. Fagin, Generalized first order spectra and polynomial time recognizable sets, in: R. Karp (Ed.), Complexity of Computation, SIAM-AMS, 1974, pp. 43–73.

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

[13] E. Gradel, K. Meer, Descriptive complexity theory over the real numbers, in: 27th STOC, 1995, pp.315–324.

[14] Y. Gurevich, Algebras of feasible functions, in: Twenty Fourth Symposium on Foundations of Computer Science, IEEE Computer Society Press, London, 1983, pp. 210–214.

[15] Y. Gurevich, E. Grädel, Tailoring recursion for complexity, Journal of Symbolic Logic 60 (1995) 952–969.

[16] M. Hofmann, Type systems for polynomial-time computation, 1999, Habilitation.

[17] N. Immerman, Descriptive Complexity, Springer, Berlin, 1999.

[18] N. Jones, The expressive power of higher order types, Journal of Functional Programming 11 (2001) 55–94.

[19] D. Leivant, Predicative recurrence and computational complexity I: Word recurrence and poly-time, in: P. Clote, J. Remmel (Eds.), Feasible Mathematics II, Birkhäuser, 1994, pp. 320–343.

[20] D. Leivant, J.-Y. Marion, Lambda calculus characterizations of poly-time, Fundamenta Informaticae 19 (1,2) (1993) 167–184.

[21] D. Leivant, J.-Y. Marion, Ramified recurrence and computational complexity II: substitution and poly-space, in: L. Pacholski, J. Tiuryn (Eds.), Computer Science Logic, 8th Workshop, CSL'94, Lecture Notes in Computer Sciences, vol. 933, Kazimierz, Poland, Springer, Berlin, 1995, pp. 369–380.

[22] J.-Y. Marion, J.-Y. Moyen, Efficient first order functional program interpreter with time bound certifications, in: LPAR, Lecture Notes in Computer Sciences, vol. 1955, Springer, Berlin, 2000, pp. 25–42.
[23] B. Poizat, Les Petits Cailloux. Aléas, 1995.
[24] V. Sazonov, Polynomial computability and recursivity in finite domains, Elektronische Informationsverarbeitung und Kybernetik 7 (1980) 319–323.