Seminar Report

Term Rewriting Characterizations of Complexity Classes

Martin Avanzini
martin.avanzini@students.uibk.ac.at

30th July, 2007
Supervisor: Dr. Georg Moser

Abstract

Cichon and Weiermann introduced the notion of term rewriting characterizations, a rewriting theoretic framework which allows the application of techniques from the field of rewriting in the study of recursive function theory. In this report we introduce the reader to the topic by reflecting the results and reasoning carried out in two papers by Isabelle Oitavem, leading to a characterization of FPSPACE, the functions computable in polynomial space. We present a rewrite system $\mathcal{S}$ that naturally models the functions in FPSPACE. It is shown that every $\mathcal{S}$ reduction strategy yields an algorithm that runs in polynomial space. Therefore $\mathcal{S}$ exactly describes the functions in FPSPACE.

Contents

1. Introduction

In 1995, Cichon and Weiermann [4] introduced a rewriting theoretic framework for investigating recursive function theory. The general idea is very simple. Assume a class $\mathscr{C}$ of recursive functions where every function f is defined via an equation of shape

$$\mathrm{f}(x_1,\ldots,x_n) = \mathrm{g}(x_1,\ldots,x_n,\lambda y_1,\ldots,y_n.\,\mathrm{f}(y_1,\ldots,y_n)) \qquad (*)$$

Here g denotes some previously defined function. The corresponding rewrite system $\mathcal{R}$ consists of the rewrite rules

$$\mathsf{f}(x_1,\ldots,x_n) \to \mathsf{g}(x_1,\ldots,x_n,\lambda y_1,\ldots,y_n.\,\mathsf{f}(y_1,\ldots,y_n))$$

obtained by orienting the equations (*). In all non pathological cases the so obtained system is terminating and orthogonal, yielding a convenient model of computation for the functions in $\mathscr{C}$. $\mathcal{R}$ at hand allows then the application of techniques from the field of rewriting to gain intrinsic information about the considered class $\mathscr{C}$.

In [4], the authors demonstrate the application by presenting transparent and illustrative proofs of the fact that the recursion scheme of parameter recursion, simple nested recursion and unnested multiple recursion do not lead outside the class of primitive recursive functions (PR). To exemplify this, we show that PR is closed under the parameter recursion scheme. Assume that $\mathcal{P}$ is a characterization of PR and that the signature of $\mathcal{P}$ contains the constructor symbols s and $\mathsf{0}$. We map every natural number n to a numeral $\underline{n}$ by $\underline{0} := \mathsf{0}$ and $\underline{n+1} = \mathsf{s}(\underline{n})$. For each primitive recursive function f the signature of $\mathcal{P}$ contains a corresponding function symbol f such that any $\mathcal{P}$-reduction of $\mathsf{f}(\underline{n}_1,\ldots,\underline{n}_k)$ reduces to the numeral $\underline{\mathrm{f}(n_1,\ldots,n_k)}$. Hence a computation of $\mathrm{f}(n_1,\ldots,n_k)$ amounts to a reduction of $\mathsf{f}(\underline{n}_1,\ldots,\underline{n}_k)$ under $\mathcal{P}$. Next define the rewrite system $\mathcal{P}'$ as the least extension of $\mathcal{P}$ containing for every function symbol g of arity k, h of arity $k+2$ and all symbols $\mathbf{p} = \mathsf{p}_1,\ldots,\mathsf{p}_k$ of arity $k+1$ the rules

$$\begin{aligned}\mathsf{PR}[\mathsf{g},\mathsf{h},\mathbf{p}](0,y_1,\ldots,y_k) &\to \mathsf{g}(y_1,\ldots,y_k)\\ \mathsf{PR}[\mathsf{g},\mathsf{h},\mathbf{p}](\mathsf{s}(x),y_1,\ldots,y_k) &\to \mathsf{h}(x,y_1,\ldots,y_k,\mathsf{PR}[\mathsf{g},\mathsf{h},\mathbf{p}](x,\mathbf{p}(x,y_1,\ldots,y_k))\end{aligned}$$

Let g, h and p denote the functions represented by the symbols g,h and $\mathbf{p}$. Then a $\mathcal{P}'$-reduction of $\mathsf{PR}[\mathsf{g},\mathsf{h},\mathbf{p}](\underline{n},\underline{m}_1,\ldots,\underline{m}_k)$ corresponds to the computation of $\mathrm{f}(n,m_1,\ldots,m_k)$ where f is defined by parameter recursion on g, h and p. Hence $\mathcal{P}'$ yields a characterization of the class PRP that is obtained by closing PR under parameter recursion. To show that PRP $\subseteq$ PR we define for every function symbol f in the signature of $\mathcal{P}'$ the derivation length function $\mathrm{Dl}^{\mathsf{f}}_{\mathcal{P}'}$ as the maximum length of all possible rewrite sequences starting from $\mathsf{f}(\underline{m}_1,\ldots,\underline{m}_k)$. Then it can be shown that the derivation length function $\mathrm{Dl}^{\mathsf{f}}_{\mathcal{P}'}$ is primitive recursive for every function symbol f admitted by $\mathcal{P}'$. The number of steps for computing a derivation on a turing machine is primitive recursive (even elementary) in the derivation length, for example see [6]. Thus any $\mathcal{P}'$-reduction can

be computed by a primitive recursive function. Since every function f $\in$ PRP is naturally computed by $\mathcal{P}'$ it follows that PRP $\subseteq$ PR, concluding the claim. By similar reasoning it is shown in [4] that the primitive recursive functions are closed under simple nested recursion and unnested multiple recursion.

In the meantime, term rewriting characterizations of various classes have been introduced. In [1] Beckmann and Weiermann present a characterization of the class FP, the class of functions computable in polynomial time. For this, instead of the classical recursive-theoretic characterization given by Cobham in [5], they use as starting point the characterization of FP given by Bellantoni and Cook in [3]. Whereas Cobham's characterization is defined using a bounded recursion scheme, Bellantoni and Cook's characterization is entirely resource-bound free and translates seamless to a term rewriting characterization.

Following Beckmann and Weiermanns approach, Oitavem describes in [8] a term rewriting characterizations of FPSPACE, the class of functions computable in polynomial space. Again an alternative characterization [7] that does not depend on explicit bounds is used as a basis. However, a direct translation of the recursion schemes from [7] results in rewrite rules that give rise to exponentially fast growing terms. Clearly, if we consider the space complexity FPSPACE then the goal is to bind the lengths of terms occurring in a derivation by a polynomial in the size of the input arguments. Thus the recursion schemes from [7] are not admissible in the context of term rewriting. Oitavem solves this problem in [8] by replacing the recursion schemes by two adequate reformulations. This way an alternative characterization PS of the functions from FPSPACE is obtained. PS can then be almost straight forward translated into the term rewriting characterization $\mathcal{S}$ of the functions computable in polynomial space.

In this work we mainly reflect the results and reasoning carried out in [7, 8]. Although we study functions that are defined on binary words, all presented results carry over to functions defined on $\mathbb{N}$. We show that we can drop one of the recursion schemes in the definition of PS, resulting in a simplified version of the term rewriting characterization given in [8]. Whereas Oitavem gives a quite complicated termination proof of $\mathcal{S}$ spanning over multiple pages, we present a very clear and simple proof by means of semantic labeling. Finally it should be noted that some of the proofs given in [7, 8] lack details that need further explanation. For example, in the proof of the main theorem only the "most interesting cases" are discussed. However, it is not clear to the author how the theorem can be established without further modifications for one of the unexplained cases, namely when the considered root symbol denotes a function defined by composition. We therefore present a reformulation of the main theorem and provide a detailed proof.

The report is structured as follows. In Section 2.1 we start with the classic characterization of FPSPACE using bounded recursion schemes. In Section 3 we introduce the reader to PS, a resource-bound free characterization of the functions computable in polynomial space. This correspondence is shown in Appendix B and C, without introducing the previously mentioned intermediate characterization from [8]. Based on PS we finally present in Section 4 the rewrite system $\mathcal{S}$ that naturally models the functions from PS. It is proven that

any $\mathcal{S}$-reduction admits an algorithm that runs in polynomial space. Therefore $\mathcal{S}$ yields a term rewriting characterization of exactly those functions computable in polynomial space.

2. Preliminaries

Let $\mathbb{N} = \{1, 2, \dots\}$ denote the set of natural numbers and $\mathbb{W}$ denote the set of all binary words $\{0,1\}^*$. The empty word is denoted by ϵ.

Let $x, y, z \in \mathbb{W}$. We write xy for the concatenation of the sequences x and y. $|x|$ denotes the length of x, that is $|\epsilon| := 0$ and $|xi| := |x| + 1$ for $i \in \{0,1\}$. x' denotes the successor of x, i.e. the sequence that follows immediately after x when we consider all binary words ordered according to length and, within the same length, lexicographically. We write $x + y$ for the y-th successor of x, i.e. $x+\epsilon = x$ and $x+y' = (x+y)'$. Likewise $x\dot{-}y$ denotes y-th predecessor of x, that is $y\dot{-}\epsilon = y$, $\epsilon\dot{-}x = \epsilon$ and $y'\dot{-}x' = y\dot{-}x$. Then for all $a, b, c \in \mathbb{W}$, $a\dot{-}(b + c) = (a\dot{-}b)\dot{-}c$ holds. We denote by $<$ ($\leq$) both the natural order on $\mathbb{N}$ and the transitive (and reflexive) closure of the successor relation $\{(x, x') \mid x \in \mathbb{W}\}$. We may also write $x > y$ ($x \geq y$) instead of $y < x$ ($y \leq x$). $\exists z.xz = y$ abbreviates to $x \subseteq y$ and we call x a prefix of y. $x_{|y}$ denotes the truncation of x to the length of y, i.e. $x_{|y} = x$ if $|x| \leq |y|$ and otherwise $x_{|y} = z$ where $|z| = |y|$ and $z \subseteq x$. We write x^R to denote the sequence x written in reverse order and x^n for the n-times concatenation of x.

We write $\mathbf{x}$ for a n-tuple of arguments, the i-th element of $\mathbf{x}$ is denoted by x_i when clear from context. Likewise we write $\mathbf{g}$ for a k-tuple of n-ary functions, g_i denoting the i-th element of $\mathbf{g}$.

We follow the notion of rewriting from [2]. We call a set A equipped with interpretation functions $f_{\mathcal{A}} \colon A^k \to A$ for every $\mathsf{f} \in \mathcal{F}^k$ an *$\mathcal{F}$-algebra*. A is called the carrier of the algebra $\mathcal{A}$. We denote by $[\alpha]_{\mathcal{A}}(t)$ the interpretation of $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ under $\mathcal{A}$ with assignment α. In the case where t is ground we simply write $t_{\mathcal{A}}$. We call an algebra a *model* of a rewrite system $\mathcal{R}$ if for all rules $l \to r \in \mathcal{R}$ and all assignments α, $[\alpha]_{\mathcal{A}}(l) = [\alpha]_{\mathcal{A}}(r)$ holds. For a model $\mathcal{A}$ of $\mathcal{R}$ and for all ground terms s, t, if $s \to_{\mathcal{R}}^* t$ then $s_{\mathcal{A}} = t_{\mathcal{A}}$.

Let $\mathcal{R}$ be a TRS over signature $\mathcal{F}$ and $\mathcal{A}$ be a non-empty model for $\mathcal{R}$ with carrier A. A *labeling* l for $\mathcal{A}$ consists of sets of labels $L_{\mathsf{f}} \subseteq A$ for every n-ary function symbol $\mathsf{f} \in \mathcal{F}$ together with mappings $l_{\mathsf{f}} \colon A^n \to L_{\mathsf{f}}$ whenever $L_{\mathsf{f}} \neq \varnothing$. A term is *labeled* under an assignment α by $\mathrm{lab}_\alpha(t) = t$ if $t \in \mathcal{V}$, $\mathrm{lab}_\alpha(\mathsf{f}(t_1, \dots, t_n)) = \mathsf{f}(\mathrm{lab}_\alpha(t_1), \dots, \mathrm{lab}_\alpha(t_n))$ if $L_{\mathsf{f}} = \varnothing$ and if $L_{\mathsf{f}} \neq \varnothing$ then $\mathrm{lab}_\alpha(\mathsf{f}(t_1, \dots, t_n)) = \mathsf{f}^a(\mathrm{lab}_\alpha(t_1), \dots, \mathrm{lab}_\alpha(t_n))$. Here a denotes the label obtained by $l_f([\alpha]_{\mathcal{A}}(t_1), \dots, [\alpha]_{\mathcal{A}}(t_n))$. The labeled TRS is defined by $\mathcal{R}_{\mathrm{lab}} := \{\mathrm{lab}_\alpha(l) \to \mathrm{lab}_\alpha(r) \mid l \to r \in \mathcal{R} \text{ and every assignment } \alpha\}$. $\mathcal{R}$ is terminating if and only if $\mathcal{R}_{\mathrm{lab}}$ is terminating.

2.1. Classic Characterization of FPSPACE

Cobham [5] characterizes FP as the smallest class of functions containing the projection, i-concatenation and conditional functions and that is closed under the composition and bounded recursion on notation schemes. If we close

FP under bounded primitive recursion we obtain exactly the class of functions computable in polynomial space [9]. Intuitively, this is because primitive recursion allows exponential many recursion steps in the length of the input. At the same time, the bounding requirement ensures that the result is polynomially bounded in size. Therefore we may describe FPSPACE, the class of functions computable in polynomial space, as follows:

Definition 2.1. For $k \in \mathbb{N}$ the set of k-ary functions $\mathrm{FPSPACE}^k$ is inductively defined as follows:

1. **(Constant)** $\mathrm{E} : \mathbb{W}^k \to \mathbb{W} \in \mathrm{FPSPACE}^k$ denotes the function
$$\mathrm{E}^k(x_1, \ldots, x_k) = \epsilon$$
2. **(i-Concatenation)** For $i \in \{0, 1\}$, $\mathrm{C} : \mathbb{W} \to \mathbb{W} \in \mathrm{FPSPACE}^1$ denotes the function
$$\mathrm{C}_i(x) = xi$$
3. **(Projection)** For every $k \neq 0$ and $j \leq k$, $\mathrm{P}^k_j : \mathbb{W}^k \to \mathbb{W} \in \mathrm{FPSPACE}^k$ denotes the function
$$\mathrm{P}^k_j(x_1, \ldots, x_k) = x_j$$
4. **(Smash)** $\# \in \mathrm{FPSPACE}^2$ denotes the function
$$\#(x, y) = 2^{|x| \cdot |y|}$$
5. **(Composition)** If $\mathrm{h} \in \mathrm{FPSPACE}^l$ and $\mathrm{g}_1, \ldots, \mathrm{g}_l \in \mathrm{FPSPACE}^k$ then $COMP[\mathrm{h}, \mathrm{g}_1, \ldots, \mathrm{g}_l]$ denotes the function $\mathrm{f} : \mathbb{W}^k \to \mathbb{W} \in \mathrm{FPSPACE}^k$ satisfying
$$\mathrm{f}(x_1, \ldots, x_k) = \mathrm{h}(\mathrm{g}_1(x_1, \ldots, x_k), \ldots, \mathrm{g}_l(x_1, \ldots, x_k))$$
6. **(Bounded Recursion on Notation)** If $\mathrm{g} \in \mathrm{FPSPACE}^k$, $\mathrm{h}_0, \mathrm{h}_1 \in \mathrm{FPSPACE}^{k+2}$ and $\mathrm{t} \in \mathrm{FPSPACE}^{k+1}$ then by $BRECN[\mathrm{g}, \mathrm{h}_0, \mathrm{h}_1, \mathrm{t}]$ we denote the function $\mathrm{f} : \mathbb{W}^{k+1} \to \mathbb{W} \in \mathrm{FPSPACE}^{k+1}$ such that for $i \in \{0, 1\}$,
$$\begin{aligned} \mathrm{f}(\epsilon, y_1, \ldots, y_k) &= \mathrm{g}(y_1, \ldots, y_k) \\ \mathrm{f}(xi, y_1, \ldots, y_k) &= \mathrm{h}_i(x, y_1, \ldots, y_k, \mathrm{f}(x, y_1, \ldots, y_k)) \end{aligned}$$
provided that for all $x, y_1, \ldots, y_k \in \mathbb{W}$
$$\mathrm{h}_i(x, y_1, \ldots, y_k, \mathrm{f}(x, y_1, \ldots, y_k)) \leq \mathrm{t}(x, y_1, \ldots, y_k)$$
7. **(Bounded Primitive Recursion)** If $\mathrm{g} \in \mathrm{FPSPACE}^k$, $\mathrm{h} \in \mathrm{FPSPACE}^{k+2}$ and $\mathrm{t} \in \mathrm{FPSPACE}^{k+1}$ then by $BREC[\mathrm{g}, \mathrm{h}, \mathrm{t}]$ we denote the function $\mathrm{f} : \mathbb{W}^{k+1} \to \mathbb{W} \in \mathrm{FPSPACE}^{k+1}$ such that
$$\begin{aligned} \mathrm{f}(\epsilon, y_1, \ldots, y_k) &= \mathrm{g}(y_1, \ldots, y_k) \\ \mathrm{f}(x', y_1, \ldots, y_k) &= \mathrm{h}(x, y_1, \ldots, y_k, \mathrm{f}(x, y_1, \ldots, y_k)) \end{aligned}$$
provided that for all $x, y_1, \ldots, y_k \in \mathbb{W}$,
$$\mathrm{h}(x, y_1, \ldots, y_k, \mathrm{f}(x, y_1, \ldots, y_k)) \leq \mathrm{t}(x, y_1, \ldots, y_k)$$

Then $\mathrm{FPSPACE} = \bigcup_{k\in\mathbb{N}} \mathrm{FPSPACE}^k$.

Proposition 2.2. FPSPACE *coincides with the functions computable in polynomial space.*

Notice that if $\mathrm{f} \in \mathrm{FPSPACE}^k$ then there exists a (monotone) polynomial b_f such that for all $x_1, \ldots, x_k \in \mathbb{W}$, $|\,\mathrm{f}(x_1, \ldots, x_k)| \leq b_f(|x_1|, \ldots, |x_k|)$. We are going to use this observation several times.

3. New Characterization of FPSPACE

In [8] a new characterization of the polyspace computable function was introduced. This characterization uses similar techniques as employed by Bellantoni and Cook in [3], where the input variables can occur in two kinds of positions: "normal" and "safe". Instead of $f(m_1, \ldots, m_k, n_1, \ldots, n_l)$ we therefore write $f(m_1, \ldots, m_k; n_1, \ldots, n_l)$ where $m_1, \ldots, m_k$ denotes the normal and $n_1, \ldots, n_l$ denotes the safe arguments of f.

The purpose of this distinction is to break the strength of the recursion scheme. This will become more clear in a moment.

Definition 3.1. For $k, l \in \mathbb{N}$ the set of functions $\mathrm{PS}^{k,l}$ with k normal and l safe arguments is inductively defined as follows:

1. **(Constant)** $\mathrm{E} : \mathbb{W}^{k,l} \to \mathbb{W} \in \mathrm{FPSPACE}^{k,l}$ denotes the function
$$\mathrm{E}^{k,l}(x_1, \ldots, x_k; y_1, \ldots, y_l) = \epsilon$$
2. **(i-Concatenation)** For $i \in \{0, 1\}$, $\mathrm{C} : \mathbb{W}^{1,0} \to \mathbb{W} \in \mathrm{FPSPACE}^{1,0}$ denotes the function
$$\mathrm{C}_i(x;) = xi$$
3. **(Deletion)** $\mathrm{D} : \mathbb{W}^{0,1} \to \mathbb{W} \in \mathrm{FPSPACE}^{0,1}$ denotes the function
$$\mathrm{D}(;\epsilon) = \epsilon \qquad \mathrm{D}(; xi) = x \text{ where } x \in \{0, 1\}$$
4. **(Successor)** $\mathrm{D} : \mathbb{W}^{1,0} \to \mathbb{W} \in \mathrm{FPSPACE}^{1,0}$ denotes the function
$$\mathrm{S}(x;) = x'$$
5. **(Predecessor)** $\mathrm{P} : \mathbb{W}^{0,1} \to \mathbb{W} \in \mathrm{FPSPACE}^{0,1}$ denotes the function
$$\mathrm{P}(;\epsilon) = \epsilon \qquad \mathrm{P}(; x') = x$$
6. **(Projection)** For every $j \in \{1, \ldots, k + l\}$, $\mathrm{P}_j : \mathbb{W}^{k,l} \to \mathbb{W} \in \mathrm{FPSPACE}^{k,l}$ denotes the function
$$\mathrm{P}_j^{k,l}(x_1, \ldots, x_k; x_{k+1}, \ldots, x_{k+l}) = x_j$$

7. **(Modified Concatenation)** For $i \in \{0,1\}$, $\mathrm{B}_i : \mathbb{W}^{2,1} \to \mathbb{W} \in \mathrm{FPSPACE}^{2,1}$ denotes the function
$$\mathrm{B}_i(x, y; z) = x(z^R i)_{|y}$$

8. **(Modified Product)** $\times : \mathbb{W}^{2,0} \to \mathbb{W} \in \mathrm{FPSPACE}^{2,0}$ denotes the function
$$\times(x, y;) = \underbrace{x^R \cdots x^R}_{|y|}$$

9. **(Conditional)** $\mathrm{Q} : \mathbb{W}^{0,4} \to \mathbb{W} \in \mathrm{FPSPACE}^{0,4}$ denotes the function
$$\mathrm{Q}(; x, y, z_0, z_1) = \begin{cases} y \text{ if } x = \epsilon, \text{ else} \\ z_i \text{ where } x = wi, i \in \{0,1\} \end{cases}$$

10. **(Safe Composition)** If $\mathrm{g} \in \mathrm{FPSPACE}^{k',l'}$, $\mathrm{r}_1, \ldots, \mathrm{r}_{k'} \in \mathrm{FPSPACE}^{k,0}$ and $\mathrm{s}_1, \ldots, \mathrm{s}_{l'} \in \mathrm{FPSPACE}^{k,l}$ then by $SUB[\mathrm{g}, \mathrm{r}_1, \ldots, \mathrm{r}_{k'}, \mathrm{s}_1, \ldots, \mathrm{s}_{l'}]$ we denote the function $\mathrm{f} : \mathbb{W}^{k,l} \to \mathbb{W} \in \mathrm{FPSPACE}^{k,l}$ satisfying
$$\mathrm{f}(\mathbf{x}; \mathbf{y}) = g(\mathrm{r}_1(\mathbf{x};), \ldots, \mathrm{r}_{k'}(\mathbf{x};); \mathrm{s}_1(\mathbf{x}; \mathbf{y}), \ldots, \mathrm{s}_{l'}(\mathbf{x}; \mathbf{y}))$$

11. **(Generalized Safe Primitive Recursion)** If $\mathrm{h} \in \mathrm{FPSPACE}^{k,l}$ then $PREC[\mathrm{h}] \in \mathrm{FPSPACE}^{k+1,l}$ denotes the the function $\mathrm{f} : \mathbb{W}^{k+1,l} \to \mathbb{W}$ such that
$$\begin{aligned} \mathrm{f}(\epsilon, b, \mathbf{x}; \mathbf{y}, a) &= a \\ \mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) &= \mathrm{f}(z, b', \mathbf{x}; \mathbf{y}, \mathrm{h}(b, \mathbf{x}; \mathbf{y}, a)) \end{aligned}$$

The functions from FPSPACE are exactly those functions from PS with only normal input positions. Further every function definable in PS is definable in FPSPACE. A proof of this can be found in Appendix B and C.

Theorem 3.2. PS *coincides with the functions computable in polynomial space.*

As a consequence there exists a monotone polynomial p_f such that for all $x_1, \ldots, x_k \in \mathbb{W}$, $|\mathrm{f}(x_1, \ldots, x_k)| \leq p_f(|x_1|, \ldots, |x_k|)$ for every function $\mathrm{f} \in \mathrm{PS}$. To explain the effect of the employed recursion scheme, assume that f is defined by generalized safe primitive recursion on h. During the evaluation of f, when the first argument decreases from z to ϵ, the second argument increases from b to $b + z$. At each intermediate evaluation step of f, instead of the recursive argument, the second argument is given to the function h. This has the effect of inverting the usual evaluation order. At each step, the last argument a of f acts as an accumulator and holds the recursively computed result. a is then substituted into a safe position of the function h. As the recursion argument has to come from a normal position, and by the asymmetry of the safe composition scheme, h cannot be a function defined by recursion on a. Intuitively, this means that the depth of sub-recursion of h can not depend on the recursively computed value held by a.

Since functions without safe arguments can not be used for function h in the definition of the recursion scheme, they do not produce great increase. This is the reason why both i-concatenation functions posses no safe argument positions and is essential if we want to establish a polynomial size bound. To demonstrate this, assume that we join functions $C_i(;x) = xi$ for $i \in \{0,1\}$ to the initial functions. Notice that, although it is possible to define C_i based on C_i, the opposite does not hold. C_i at hand allows the definition of the function f such that

$$\mathrm{f}(\epsilon, b; a) = a \qquad \mathrm{f}(z', b; a) = \mathrm{f}(z, b'; C_1(;a))$$

Then $\mathrm{f}(z', m; n)$ produces a result that is exponential long in the length of the arguments.

As the i-concatenation functions with safe arguments are not admissible, the functions B_i and $\times$ have to be added to the initial functions. In fact without those PS would admit only functions with constant growth rate. The functions B_i allow the modification of safe arguments, however the length of the result is bounded by the length of the normal arguments. This restriction prevents the definition of functions where the recursively computed result grows larger then the values in normal position. Using modified product together with safe composition we can then construct sufficiently large normal arguments, without exceeding a polynomial size bounds.

As stated before, a resource-bound free characterization of FPSPACE has already been given in [7] earlier. In [7] it is proven that FPSPACE coincides with the smallest class of functions that contains certain initial functions and that is closed under the safe composition, safe primitive recursion and safe recursion on notation schemes. For PS, some of the initial functions from [7] have been slightly modified in favor of replacements yielding simplified rewrite rules. Further, the recursion schemes from [7] are the following:

1. **(Safe Recursion on Notation)** Define the new function f by

$$\begin{aligned} \mathrm{f}(\epsilon, \mathbf{x}; \mathbf{y}) &= \mathrm{g}(\mathbf{x}; \mathbf{y}) \\ \mathrm{f}(zi, \mathbf{x}; \mathbf{y}) &= \mathrm{h}_i(z, \mathbf{x}; \mathbf{y}, f(z, \mathbf{x}; \mathbf{y})) \end{aligned}$$

   for $i \in \{0,1\}$.

2. **(Safe Primitive Recursion)** Define the new function f by

$$\begin{aligned} \mathrm{f}(\epsilon, \mathbf{x}; \mathbf{y}) &= \mathrm{g}(\mathbf{x}; \mathbf{y}) \\ \mathrm{f}(z', \mathbf{x}; \mathbf{y}) &= \mathrm{h}(z, \mathbf{x}; \mathbf{y}, \mathrm{f}(z, \mathbf{x}; \mathbf{y})) \end{aligned}$$

The obvious term rewriting rules corresponding to the safe primitive recursion scheme are similar to

$$\begin{aligned} \mathsf{PREC}[\mathsf{g}, \mathsf{h}](\epsilon, \mathbf{x}; \mathbf{y}) &\to \mathsf{g}(\mathbf{x}; \mathbf{y}) \\ \mathsf{PREC}[\mathsf{g}, \mathsf{h}](z', \mathbf{x}; \mathbf{y}) &\to \mathsf{h}(z, \mathbf{x}; \mathbf{y}, \mathsf{PREC}[\mathsf{g}, \mathsf{h}](z, \mathbf{x}; \mathbf{y})) \end{aligned}$$

yielding for $\mathsf{f} = \mathsf{PREC}[\mathsf{g}, \mathsf{h}]$ the derivation

$$\begin{aligned}\mathsf{f}(z'', \mathbf{x}; \mathbf{y}) &\to \mathsf{h}(z', \mathbf{x}; \mathbf{y}, \mathsf{f}(z', \mathbf{x}; \mathbf{y})) \\ &\to \mathsf{h}(z', \mathbf{x}; \mathbf{y}, \mathsf{h}(z, \mathbf{x}; \mathbf{y}, \mathsf{f}(z, \mathbf{x}; \mathbf{y}))) \\ &\to^* \mathsf{h}(z', \mathbf{x}; \mathbf{y}, \mathsf{h}(z, \mathbf{x}; \mathbf{y}, \mathsf{h}(\dots \mathsf{h}(\epsilon, \mathbf{x}; \mathbf{y}, \mathsf{g}(\mathbf{x}; \mathbf{y})) \dots))\end{aligned}$$

Here the symbol h occurs exponentially often in the length of z'' in the last term. Thus the last term is exponential in length with respect to the first term. In the definition of PS both recursion schemes from [7] are replaced by the generalized safe primitive recursion scheme, the corresponding rewrite rules are similar to

$$\begin{aligned}\mathsf{PREC}[\mathsf{h}](\epsilon, b, \mathbf{x}; \mathbf{y}, a) &\to a \\ \mathsf{PREC}[\mathsf{h}](z', b, \mathbf{x}; \mathbf{y}, a) &\to \mathsf{PREC}[\mathsf{h}](z, b', \mathbf{x}; \mathbf{y}, \mathsf{h}(b, \mathbf{x}; \mathbf{y}, a))\end{aligned}$$

The term obtained by instantiating the second position of $\mathsf{f}' = \mathsf{PREC}[\mathsf{h}]$ with ϵ and the last position with $\mathsf{g}(\mathbf{x}; \mathbf{y})$ admits a derivation

$$\begin{aligned}\mathsf{f}'(z'', \epsilon, \mathbf{x}; \mathbf{y}, \mathsf{g}(\mathbf{x}; \mathbf{y})) &\to \mathsf{f}'(z', 0, \mathbf{x}; \mathbf{y}, \mathsf{h}(\epsilon, \mathbf{x}; \mathbf{y}, \mathsf{g}(\mathbf{x}; \mathbf{y}))) \\ &\to \mathsf{f}'(z, 1, \mathbf{x}; \mathbf{y}, \mathsf{h}(0, \mathbf{x}; \mathbf{y}, \mathsf{h}(\epsilon, \mathbf{x}; \mathbf{y}, \mathsf{g}(\mathbf{x}; \mathbf{y})))) \\ &\to^* \mathsf{h}(z', \mathbf{x}; \mathbf{y}, \mathsf{h}(z, \mathbf{x}; \mathbf{y}, \mathsf{h}(\dots \mathsf{h}(\epsilon, \mathbf{x}; \mathbf{y}, \mathsf{g}(\mathbf{x}; \mathbf{y})) \dots))\end{aligned}$$

The last term equals the last term in the derivation of $\mathsf{f}(z'', \mathbf{x}; \mathbf{y})$ given before, again exponentially in size compared to the first. However, we are now able to enforce the normalization of the safe arguments to avoid nested occurrences of h and prevent this way derivations that grow exponential in size.

It should be noted that the definition of PS from [8] is explicitly closed under a second recursion scheme. In essence, the second scheme is used to show that PS is closed under safe recursion on notation. Intuitively it is clear that safe primitive recursion, admitting exponentially many recursion steps in the length of the recursion input, is strictly more powerful than safe recursion on notation. Using the conditional function one can easily define a function h that checks the last bit of the recursion input and calls appropriate function h_0, h_1. Thus one can simulate safe recursion on notation using safe primitive recursion by skipping the recursion steps not performed by safe recursion on notation (cf. Appendix A). We can therefore safely omit this scheme in the definition of PS. This result in a simplified term rewriting characterization since no rules for the second scheme have to be introduced.

Finally let us state the following frequently used lemma.

Lemma 3.3. *Assume that* f *is defined by generalized safe recursion on* h*, i.e.* $f = PREC[h]$*. Then for all* $a, b, \mathbf{x}, \mathbf{y}, z \in \mathbb{W}$*,*

$$\mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) = \mathrm{h}(b + z, \mathrm{f}(z, b, \mathbf{x}; \mathbf{y}, a))$$

This directly follows from the observation that for all $a, b, \mathbf{x}, \mathbf{y}, z \in \mathbb{W}$

$$z \geq c \Rightarrow \mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) = \mathrm{f}(z' \dot{-} c', b + c', \mathbf{x}; \mathbf{y}, h(b + c, \mathrm{f}(c, b, \mathbf{x}; \mathbf{y}, a)))$$

Here the right-hand side may be perceived as the c-th recursion step of f. That is, when the first argument to f decreased from z' to $z' \dot{-} c'$, the second argument increased from b to $b + c'$. At the same time the last argument holds the value $h(b + c, \mathrm{f}(c, b, \mathbf{x}; \mathbf{y}, a)) = \mathrm{f}(c', b, \mathbf{x}; \mathbf{y}, a)$. For a proof of this we refer the reader to Appendix A.

4. A Rewrite System for FPSPACE

In this section we establish a term rewriting characterization of the functions computable in polynomial space. More precise, we construct a TRS $\mathcal{S}$ that naturally computes the functions in PS. That is, for any function $\mathrm{f} \in \mathrm{PS}$ and $m_1, \ldots, m_k, n_1, \ldots, n_l \in \mathbb{W}$ there exists a term that normalizes under $\mathcal{S}$ to a term corresponding to the result of $\mathrm{f}(m_1, \ldots, m_k; n_1, \ldots, n_l)$. We show that for each such derivation the size of every occurring term is bounded by a polynomial in the length of $m_1, \ldots, m_k, n_1, \ldots, n_l$. Therefore, based on $\mathcal{S}$, we can construct for any $\mathrm{f} \in \mathrm{PS}$ an algorithm that runs in polynomial space. Since PS coincides with FPSPACE, $\mathcal{S}$ effectively yields a characterization of exactly those functions that are computable in polynomial space. The signature of $\mathcal{S}$ contains following function symbols.

Definition 4.1. For $k, l \in \mathbb{N}$ we define the set $\mathcal{F}^{k,l}$ of *safe recursive function symbols* with k normal and l safe argument positions inductively as follows:

1. $\mathsf{E}^{k,l} \in \mathcal{F}^{k,l}$
2. $\mathsf{C}_i \in \mathcal{F}^{1,0}$ for $i \in \{0, 1\}$
3. $\mathsf{D} \in \mathcal{F}^{0,1}$
4. $\mathsf{S} \in \mathcal{F}^{1,0}$
5. $\mathsf{P} \in \mathcal{F}^{0,1}$
6. $\mathsf{P}^{k,l}_j \in \mathcal{F}^{k,l}$ for $1 \leq j \leq k + l$
7. $\mathsf{B}_i \in \mathcal{F}^{2,1}$ for $i \in \{0, 1\}$
8. $\times \in \mathcal{F}^{2,0}$
9. $\mathsf{Q} \in \mathcal{F}^{0,4}$
10. $\mathsf{SUB}[\mathsf{g}, \mathsf{r}_1, \ldots, \mathsf{r}_{k'}, \mathsf{s}_1, \ldots, \mathsf{s}_{l'}] \in \mathcal{F}^{k,l}$ if $\mathsf{g} \in \mathcal{F}^{k',l'}, \mathsf{r}_1, \ldots, \mathsf{r}_{k'} \in \mathcal{F}^{k,0}$ and $\mathsf{s}_1, \ldots, \mathsf{s}_{l'} \in \mathcal{F}^{k,l}$
11. $\mathsf{PREC}[\mathsf{h}] \in \mathcal{F}^{k+1,l+1}$ if $\mathsf{h} \in \mathcal{F}^{k,l+1}$

Then $\mathcal{F} = \bigcup_{k,l \in \mathbb{N}} \mathcal{F}^{k,l}$ is the set of safe recursive function symbols.

The correspondence between function symbols in $\mathcal{F}$ and functions in PS is made precise in the $\mathcal{F}$-algebra π.

Definition 4.2. For $k, l \in \mathbb{N}$ and $\mathsf{f} \in \mathcal{F}$ the mapping $\pi \colon \mathcal{F} \to \mathrm{PS}$ is recursively defined as follows:

$\pi(\mathsf{E}^{k,l}) := \mathrm{E}^{k,l}$
$\pi(\mathsf{C}_i) := \mathrm{C}_i$ for $i \in \{0, 1\}$
$\pi(\mathsf{D}) := \mathrm{D}$
$\pi(\mathsf{S}) := \mathrm{S}$
$\pi(\mathsf{P}) := \mathrm{P}$
$\pi(\mathsf{P}^{k,l}_j) := \mathrm{P}^{k,l}_j$ for $j \in \{1, \ldots, k + l\}$
$\pi(\mathsf{B}_i) := \mathrm{B}_i$ for $o \in \{0, 1\}$
$\pi(\times) := \times$
$\pi(\mathsf{Q}) := \mathrm{Q}$
$\pi(\mathsf{SUB}[\mathsf{g}, \mathsf{r}, \mathsf{s}]) := SUB[\pi(\mathsf{g}), \pi(\mathsf{r}), \pi(\mathsf{s})]$
$\pi(\mathsf{PREC}[\mathsf{h}]) := PREC[\pi(\mathsf{h})]$

Then $\mathbb{W}$ equipped with interpretations $\mathsf{f}_\pi := \pi(\mathsf{f})$ is a $\mathcal{F}$-algebra, denoted by π.

Lemma 4.3. *If* $\mathrm{f} \in \mathrm{PS}^{k,l}$ *then there exists a symbol* $\mathsf{f} \in \mathcal{F}^{k,l}$ *such that* $\mathrm{f} = \mathsf{f}_\pi$.

Definition 4.4. We associate with every element in $\mathbb{W}$ a *numeral* $\underline{n} \in \mathcal{T}(\mathcal{F})$ as follows:

1. $\underline{\epsilon} := \mathsf{E}^{0,0}$
2. $\underline{ni} := \mathsf{C}_i(\underline{n};)$ where $i \in \{0,1\}$

We write ϵ instead of $\mathsf{E}^{0,0}$. For any function symbol $\mathsf{f} \in \mathcal{F}$ we define the length $|\,\mathsf{f}\,|$ of f recursively as follows: $|\epsilon| := 0$, $|\mathsf{SUB}[\mathsf{g},\mathbf{r},\mathbf{s}]| := |\,\mathsf{g}\,| + |\mathbf{r}| + |\mathbf{s}| + 1$ and $|\mathsf{PREC}[\mathsf{h}]| := |\,\mathsf{h}\,| + 1$. In all other cases we define $|\,\mathsf{f}\,| := 1$. Further the length of a ground term is defined by $|\,\mathsf{f}(\mathbf{r};\mathbf{s})| := |\,\mathsf{f}\,| + |\mathbf{r}| + |\mathbf{s}|$.

Definition 4.5. The rewrite system $\mathcal{S}$ over the signature $\mathcal{F}$ consists of following rewrite rules:

1. $\mathsf{E}^{k,l}(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+l}) \to \epsilon$ for all $k,l \in \mathbb{N}$ with $k+l \geq 1$
2. $\mathsf{D}(;\epsilon) \to \epsilon$
3. $\mathsf{D}(;\mathsf{C}_i(x;)) \to x$
4. $\mathsf{S}(\epsilon;) \to \mathsf{C}_0(\epsilon;)$
5. $\mathsf{S}(\mathsf{C}_0(x;);) \to \mathsf{C}_1(x;)$
6. $\mathsf{S}(\mathsf{C}_1(\mathsf{C}_i(x;););) \to \mathsf{C}_0(\mathsf{S}(\mathsf{C}_i(x;););)$ for $i \in \{0,1\}$
7. $\mathsf{P}(;\epsilon) \to \epsilon$
8. $\mathsf{P}(;\mathsf{C}_0(\epsilon;)) \to \epsilon$
9. $\mathsf{P}(;\mathsf{C}_0(\mathsf{C}_i(x;);)) \to \mathsf{C}_1(\mathsf{P}(;\mathsf{C}_i(x;));)$ for $i \in \{0,1\}$
10. $\mathsf{P}(;\mathsf{C}_1(x;)) \to \mathsf{C}_0(x;)$
11. $\mathsf{P}^{k,l}_j(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+l}) \to x_j$ for $j \in \{1,\ldots,k+l\}$ and $k+l \geq 1$
12. $\mathsf{B}_i(x,\epsilon;z) \to x$ for $i \in \{0,1\}$
13. $\mathsf{B}_i(x,\mathsf{C}_j(y;);\epsilon) \to \mathsf{C}_i(x;)$ for $i,j \in \{0,1\}$
14. $\mathsf{B}_i(x,\mathsf{C}_j(y;);\mathsf{C}_l(z;)) \to \mathsf{B}_i(\mathsf{C}_l(x;),y;z)$ for $i,j,l \in \{0,1\}$
15. $\times(x,\epsilon;) \to \epsilon$
16. $\times(x,\mathsf{C}_i(y;);) \to \mathsf{B}_0(\times(x,y;),x;x)$ for $i \in \{0,1\}$
17. $\mathsf{Q}(\epsilon,y,z_0,z_1) \to y$
18. $\mathsf{Q}(\mathsf{C}_i(x;),y,z_0,z_1) \to z_i$ for $i \in \{0,1\}$
19. $\mathsf{SUB}[\mathsf{g},\mathbf{r},\mathbf{s}](\mathbf{x};\mathbf{y}) \to g(\mathbf{r}(\mathbf{x};);\mathbf{s}(\mathbf{x};\mathbf{y}))$ for all $\mathsf{SUB}[\mathsf{g},\mathbf{r},\mathbf{s}] \in \mathrm{PS}$
20. $\mathsf{PREC}[\mathsf{h}](\epsilon,b,\mathbf{x};\mathbf{y},\underline{a}) \to \underline{a}$ for all $a \in \mathbb{W}$.

21. $\mathsf{PREC[h]}(\mathsf{C}_i(\underline{z};), \underline{b}, \mathbf{x}; \mathbf{y}, \underline{a}) \to$
$\mathsf{PREC[h]}(\mathsf{P}(;\mathsf{C}_i(\underline{z};)), \mathsf{S}(\underline{b};), \mathbf{x}; \mathbf{y}, h(\underline{b}, \mathbf{x}; \mathbf{y}, \underline{a}))$ for all $\mathsf{PREC[h]} \in \mathrm{PS}$ and $a, b, z \in \mathbb{W}$.

As pointed out earlier, we require that the last argument to a function symbol $\mathsf{PREC[h]}$ has to be normalized before one of the last two rules is applicable. Similar, the first and second arguments of $\mathsf{PREC[h]}$ in the last rule have to be rewritten to numerals before the rule is applicable. This avoids undesired chains of the form

$$\begin{aligned}&\mathsf{PREC[h]}(\underline{m}, \underline{b}, \cdot\,; \cdot\,, \cdot\,) \to^* \\ &\quad\to^* \mathsf{PREC[h]}(\mathsf{S}(\mathsf{S}(\ldots\mathsf{S}(\mathsf{S}(\underline{m};);)\ldots;);), \mathsf{P}(\mathsf{P}(\ldots\mathsf{P}(\mathsf{P}(\underline{b};);)\ldots;);), \cdot\,; \cdot\,, \cdot\,)\end{aligned}$$

where the last term is exponential in size. In essence, this amounts to a call-by-value reduction strategy employed at these argument positions.

One easily verifies that whenever $s \to_{\mathcal{S}} t$ holds then $s_\pi = t_\pi$. This is an immediate consequence of the following Lemma.

Lemma 4.6. *π is a model of $\mathcal{S}$.*

Proof. This follows by considering the rewrite rules in $\mathcal{S}$. □

It is also easy to see that every normal form of $\mathcal{S}$ is a numeral.

Lemma 4.7. *Let $t \in \mathcal{T}(\mathcal{F})$ be a normal form of $\mathcal{S}$. Then $t = \underline{a}$ for some $a \in \mathbb{W}$.*

The first task is to show that $\mathcal{S}$ does not admit infinite rewrite sequences. For this we want to employ lexicographic path orderings (LPOs). However we can not employ LPOs on $\mathcal{S}$ directly. The reason is that for rule 21 the right-hand side is embedded in the left-hand side. Also rule 14 can not be handled by LPOs. If we consider the intended semantics of these rules then it is clear that they do not introduce infinite chains. As Zantema has shown, we can enrich the structure of $\mathcal{S}$ with semantic information. This technique is called semantic labeling and explained in detail in [10]. To orient rule 14, we label function symbols B_0 and B_1 by exploiting the fact that the second argument decreases. Likewise for rule 21 we attach the value of the recursion argument as label to function symbols $\mathsf{PREC[h]} \in \mathrm{PS}$.

Lemma 4.8. *$\mathcal{S}$ is terminating.*

Proof. We label $\mathcal{S}$ according to the model π from Definition 4.2. We define the labeling functions as follows:

1. $l_{\mathsf{B}_i}(x, y; z) := y$ for $i \in \{0, 1\}$.
2. $l_{\mathsf{PREC[h]}}(z, b, \mathbf{x}; \mathbf{y}, a) := z$ for any $\mathsf{PREC[h]} \in \mathcal{F}$.

Labeling $\mathcal{S}$ results in the system $\mathcal{S}_{\mathrm{lab}}$ consisting of rules 1–11,15 and 17–18 from Definition 4.5 and additionally

1. $\mathsf{B}_i^{\epsilon}(x, \epsilon; z) \to x$ for $i \in \{0, 1\}$.
2. $\mathsf{B}_i^{yj}(x, \mathsf{C}_j(y;); \epsilon) \to \mathsf{C}_i(x;)$ for $i, j \in \{0, 1\}$ and $y \in \mathbb{W}$.
3. $\mathsf{B}_i^{yj}(x, \mathsf{C}_j(y;); \mathsf{C}_l(z;)) \to \mathsf{B}_i^{y}(\mathsf{C}_l(x;), y; z)$ for $i, j, l \in \{0, 1\}$ and $y \in \mathbb{W}$.
4. $\times(x, \mathsf{C}_i(y;);) \to \mathsf{B}_0^{\times(x,y;)_\pi}(\times(x, y;), x; x)$ for $i \in \{0, 1\}$
5. $\mathsf{SUB}[\mathbf{g}, \mathbf{r}, \mathbf{s}](\mathbf{x}; \mathbf{y}) \to g'(r_1(\mathbf{x};), \ldots, r_k(\mathbf{x};); s'_1(\mathbf{x}; \mathbf{y}), \ldots, s'_l(\mathbf{x}; \mathbf{y}))$
   for all $\mathsf{SUB}[\mathbf{g}, \mathbf{r}, \mathbf{s}]$ and all possible combinations of, possibly labeled, symbols $g', s'_1, \ldots, s'_l$ for $g, s_1, \ldots, s_l$. The labeling of these functions is in this case unimportant.
6. $\mathsf{PREC}[\mathsf{h}]^{\epsilon}(\epsilon, b, \mathbf{x}; \mathbf{y}, \underline{a}) \to \underline{a}$ for all $a \in \mathbb{W}$.
7. For all $z, b, a \in \mathbb{W}$ and $i \in \{0, 1\}$

$$\mathsf{PREC}[\mathsf{h}]^{m'}(\mathsf{C}_i(\underline{z};), \underline{b}, \mathbf{x}; \mathbf{y}, \underline{a}) \to \\ \to \mathsf{PREC}[\mathsf{h}]^{m}(\mathsf{P}(; \mathsf{C}_i(\underline{z};)), \mathsf{S}(\underline{b};), \mathbf{x}; \mathbf{y}, h(\underline{b}, \mathbf{x}; \mathbf{y}, \underline{a}))$$

   where $m' = zi$, $h \neq \mathsf{B}_0, h \neq \mathsf{B}_1$ and $h \neq \mathsf{PREC}[\mathsf{h}']$ for any $\mathsf{PREC}[\mathsf{h}'] \in \mathrm{PS}$.
8. For all $z, b, a \in \mathbb{W}$ and $i \in \{0, 1\}$

$$\mathsf{PREC}[\mathsf{h}]^{m'}(\mathsf{C}_i(\underline{z};), \underline{b}, \mathbf{x}; \mathbf{y}, \underline{a}) \to \\ \to \mathsf{PREC}[\mathsf{h}]^{m}(\mathsf{P}(; \mathsf{C}_i(\underline{z};)), \mathsf{S}(\underline{b};), \mathbf{x}; \mathbf{y}, h^{l_{\mathsf{h}}(b, \mathbf{x}_\pi, \mathbf{y}_\pi, a)}(\underline{b}, \mathbf{x}; \mathbf{y}, \underline{a}))$$

   where $m' = zi$, $h = \mathsf{B}_0, h = \mathsf{B}_1$ or $h = \mathsf{PREC}[\mathsf{h}']$ for some $\mathsf{PREC}[\mathsf{h}'] \in \mathrm{PS}$.

We choose a precedence $>$ such that $\mathsf{E}^{k,l} > \epsilon$ for all $k, l \in \mathbb{N}$ where $k + l \neq 0$. Further $\mathsf{S} > C_0$, $\mathsf{S} > C_1$, $\mathsf{P} > C_0$, $\mathsf{P} > C_1$, $\mathsf{B}_i^y > C_i$,$\mathsf{B}_i^{yj} > \mathsf{B}_i^y$ and $\times > \mathsf{B}_0^y$ for all $y \in \mathbb{W}$, $i, j \in \{0, 1\}$. To orient the rules for safe composition and generalized safe primitive recursion, we set $\mathsf{f}^k > \mathsf{g}^l$ if $|\,\mathsf{f}\,| > |\,\mathsf{g}\,|$. Finally for all $\mathsf{PREC}[\mathsf{h}] \in \mathrm{PS}$ we set $\mathsf{PREC}[\mathsf{h}]^k > \mathsf{PREC}[\mathsf{h}]^l$ if $k > l$.

Then it can be shown that $>_{lpo}$ is compatible with the labeled system, i.e. $l >_{lpo} r$ for every rule $l \to r \in \mathcal{S}_{\mathrm{lab}}$. Therefore $s >_{lpo} t$ whenever $s \to_{\mathcal{S}_{\mathrm{lab}}} t$. As $>$ is well-founded, $>_{lpo}$ on ground terms is well-founded and we conclude termination of the labeled system. Finally from termination of $\mathcal{S}_{\mathrm{lab}}$ we conclude termination of $\mathcal{S}$. □

Lemma 4.9. *$\mathcal{S}$ is confluent.*

Proof. $\mathcal{S}$ is orthogonal, i.e. left-linear and does not contain critical pairs [2]. From termination and orthogonality confluence follows [2]. □

Up to this point we have observed that for all $\mathrm{f} \in \mathrm{PS}$ there exists a corresponding function symbol $\mathsf{f} \in \mathcal{F}$ satisfying $\mathsf{f}_\pi = \mathrm{f}$. For all $\mathbf{m}, \mathbf{n} \in \mathbb{W}$ any normalizing $\mathcal{S}$-reduction rewrites the term $\mathsf{f}(\underline{\mathbf{m}}; \underline{\mathbf{n}})$ to the numeral $\underline{a}$ for $a \in \mathbb{W}$ satisfying $a = \mathsf{f}_\pi(\underline{\mathbf{m}}_\pi; \underline{\mathbf{n}}_\pi) = \mathrm{f}(\mathbf{m}, \mathbf{n})$. This is a direct consequence of Lemmas 4.3–4.9. Therefore we can compute every function f from PS by rewriting the

corresponding term to normal form and extracting the final result from the so obtained numeral.

To show that this computation can be done in polynomial space, we show that the length of every term occurring in a derivation is bounded by a polynomial in the length of the arguments. In the following we write $\underline{\mathsf{f}(\mathbf{r};\mathbf{s})}$ for the numeral that is the unique normal form of $\mathsf{f}(\mathbf{r},\mathbf{s}) \in \mathcal{T}(\mathcal{F})$. This is justified by confluence and termination of $\mathcal{S}$. Notice that by definition $|\underline{m}| = |m|$ for all $m \in \mathbb{W}$. Therefore and by the observation that $\underline{\mathsf{f}(\underline{\mathbf{m}};\underline{\mathbf{n}})}_{\pi} = \mathrm{f}(\mathbf{m};\mathbf{n})$ holds for some $\mathrm{f} \in \mathrm{PS}$ the next lemma follows.

Lemma 4.10. *If $f \in \mathrm{PS}$ then there exists a monotone polynomial p_f such that*

$$\forall \mathbf{m}.\forall \mathbf{n} \in \mathbb{W}.|\underline{\mathsf{f}(\underline{\mathbf{m}};\underline{\mathbf{n}})}| \leq p_{\mathsf{f}}(|\mathbf{m}|,|\mathbf{n}|)$$

To show that the length of every term occurring in a derivation of $\mathsf{f}(\underline{\mathbf{m}},\underline{\mathbf{n}})$ is polynomially bounded in the lengths of $\mathbf{m},\mathbf{n} \in \mathbb{W}$ we prove a slightly more general result.

Theorem 4.11. *Assume $\mathsf{f} \in \mathcal{F}$ and $t_1,\ldots,t_{k+l} \in \mathcal{T}(\mathcal{F})$, $b_1,\ldots,b_{k+l} \in \mathbb{N}$ satisfying*

$$\max\{|s_i| \mid t_i \to_{\mathcal{S}}^* s_i\} \leq b_i \text{ for } i \text{ in } \{1,\ldots,k+l\}$$

Then there exists a polynomial p_f such that

$$\max\{|s| \mid \mathsf{f}(t_1,\ldots,t_k;t_{k+1},\ldots,t_{k+l}) \to_{\mathcal{S}}^* s\} \leq p_{\mathsf{f}}(b_1,\ldots,b_{k+l})$$

Proof. We proof the theorem by induction on $|\,\mathsf{f}\,|$. If $|\,\mathsf{f}\,| = 0$ then $\mathsf{f} = \epsilon$ and the claim trivially follows. Assume that $|\,\mathsf{f}\,| = 1$. Then $\mathsf{f} \in \{\mathsf{E}^{k,l},\mathsf{D},\mathsf{S},\mathsf{P},\mathsf{P}_j^{k,l},\mathsf{B}_i,\times\}$ for some $k,l,j \in \mathbb{N}$ and $i \in \{0,1\}$. When $\mathsf{f} \neq \times$ then it suffices to set $p_{\mathsf{f}}(b_1,\ldots,b_n) := \Sigma_{i=1}^n b_i + 1$ since all involved rewrite rules are non size-increasing.

The proofs of the remaining cases all follow a common scheme. We consider an arbitrary normalizing derivation of $\mathsf{f}(t_1,\ldots,t_k;t_{k+1},\ldots,t_{k+l})$ and show that any occurring term is bounded by a monotone polynomial under the conditions of the lemma. By confluence and termination of $\mathcal{S}$ this implies that the lemma holds for any derivation $\mathsf{f}(t_1,\ldots,t_k;t_{k+1},\ldots,t_{k+l}) \to^* t$ where t might not be a normal form.

For the case where $\mathsf{f} = \times$, we set $p_\times(b_x,b_y) := b_y * (2 * b_x + 1) + b_x + b_y + 1$. To prove the claim, assume a normalizing derivation starting from $\times(t_x,t_y;)$ where t_x,t_y satisfy the assumption of the lemma with bounds b_x and b_y respectively. We proceed by side induction on b_y. If $b_y = 0$ then $t_y = \epsilon$ and

$$\times(t_x,\epsilon;) \to^* \times(t'_x,\epsilon;) \to \epsilon$$

where $t_x \to^* t'_x$. In this case, the claim follows directly.

For the inductive step of the side induction, assume $b_y \geq 1$ and the lemma has been shown for $b_y - 1$. Then every normalizing derivation is of shape

$$\times(t_x,t_y;) \to^* \times(t'_x,\mathsf{C}_i(t'_y;);) \to \mathsf{B}_0(\times(t'_x,t'_y;),t'_x;t'_x)$$

where $t_x \to^* t'_x$ and $t_y \to^* \mathsf{C}_i(t'_y;)$. Therefore and by transitivity of $\to_{\mathcal{S}}$, if $t'_y \to^* t''_y$ then $t_y \to^* \mathsf{C}_i(t''_y;)$. Hence by the assumption on t_x, t_y and the definition of $|\cdot|$,

$$\max\{|t''_x| \mid t'_x \to^* t''_x\} \leq b_x \text{ and } \max\{|t''_y| \mid t'_y \to^* t''_y\} \leq b_y - 1$$

Since every reduct of t''_y is bounded by $b_y - 1$ by side induction hypothesis we have

$$\max\{|s| \mid \times(t'_x, t'_y;) \to^* s\} \leq p_\times(b_x, b_y - 1)$$

To conclude the claim one just has to unfold the definition of $|\cdot|$ and apply side induction hypothesis:

$$\begin{aligned}
|\,\mathsf{B}_0(\times(t'_x, t'_y;), t'_x; t'_x)| &\leq p_\times(b_x, b_y - 1) + 2 * b_x + |\,\mathsf{B}_0\,| \\
&= (b_y - 1) * (2 * b_x + 1) + b_x + (b_y - 1) + 1 + 2 * b_x + 1 \\
&= b_y * (2 * b_x + 1) + b_x + b_y \\
&< b_y * (2 * b_x + 1) + b_x + b_y + 1 \\
&= p_\times(b_x, b_y)
\end{aligned}$$

For the inductive step of the main induction, when $|\,\mathsf{f}\,| > 1$, then f is either one of $\mathsf{SUB}[\mathsf{f}, \mathbf{r}, \mathbf{s}] \in \mathcal{F}$ or $\mathsf{PREC}[\mathsf{h}] \in \mathcal{F}$.

1. For the first case assume $f = \mathsf{SUB}[\mathsf{g}, \mathsf{r}, \mathsf{s}]$. The general case follows then by same reasoning. Any normalizing derivation of $\mathsf{f}(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})$ is of the form

$$\begin{aligned}
\mathsf{f}(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l}) &\to^* \mathsf{f}(t'_1, \ldots, t'_k; t'_{k+1}, \ldots, t'_{k+l}) \\
&\to \mathsf{g}(\mathsf{r}(t'_1, \ldots, t'_k;); \mathsf{s}(t'_1, \ldots, t'_k; t'_{k+1}, \ldots, t'_{k+l})) \\
&\to^+ \underline{\mathsf{f}(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l})}
\end{aligned}$$

where $t_i \to^* t'_i$. Clearly for all t such that $\mathsf{f}(t_1, \ldots, t_k; t_{k+1}, \ldots, t_{k+l}) \to^* t \to^* \mathsf{f}(t'_1, \ldots, t'_k; t'_{k+1}, \ldots, t'_{k+l})$ we have

$$|t| \leq |f| + \Sigma_{i=1}^{k+l} b_i = |\,\mathsf{g}\,| + |\,\mathsf{r}\,| + |\,\mathsf{s}\,| + 1 + \Sigma_{i=1}^{k+l} b_i$$

Let $t_r = \mathsf{r}(t'_1, \ldots, t'_k;)$ and $t_s = \mathsf{s}(t'_1, \ldots, t'_k; t'_{k+1}, \ldots, t'_{k+l})$. Since $t'_i \to^* s_i$ implies $t_i \to^* s_i$ we have $\max\{|s_i| \mid t'_i \to^* s_i\} \leq b_i$ by the assumption on t_i. Therefore and since $|\,\mathsf{f}\,| > |\,\mathsf{r}\,|, |\,\mathsf{s}\,|$ induction hypothesis for t_r, t_s is applicable, so

$$\begin{aligned}
\max\{|t'_r| \mid t_r \to^* t'_r\} &\leq p_{\mathsf{r}}(b_1, \ldots, b_k) \\
\max\{|t'_s| \mid t_s \to^* t'_s\} &\leq p_{\mathsf{s}}(b_1, \ldots, b_{k+l})
\end{aligned}$$

for monotone polynomials p_{r} and p_{s}. Also, since $|\,\mathsf{f}\,| > |\,\mathsf{g}\,|$ and by the previous observation, induction hypothesis is applicable for g. Thus we may assume there exists a monotone polynomial p_h such that

$$\max\{|t| \mid g(t_r; t_s) \to^* t\} \leq p_{\mathsf{g}}(p_{\mathsf{r}}(b_1, \ldots, b_k), p_{\mathsf{s}}(b_1, \ldots, b_{k+l}))$$

Observe that $|r|+\Sigma_{i=1}^{k} b_i \leq p_{\mathsf{r}}(b_1,\ldots,b_k)$ and $|s|+\Sigma_{i=1}^{k+l} b_i \leq p_{\mathsf{s}}(b_1,\ldots,b_{k+l})$. Assuming otherwise one easily derives a contradiction. By similar reasoning one obtains

$$|\,\mathsf{g}\,| + |\,\mathsf{r}\,| + |\,\mathsf{s}\,| + \Sigma_{i=1}^{k} b_i + \Sigma_{i=1}^{k+l} b_i \leq p_{\mathsf{g}}(p_{\mathsf{r}}(b_1,\ldots,b_k), p_{\mathsf{s}}(b_1,\ldots,b_{k+l}))$$

Thus setting

$$p_{\mathsf{f}}(b_1,\ldots,b_{k+l}) := p_{\mathsf{g}}(p_{\mathsf{r}}(b_1,\ldots,b_k), p_{\mathsf{s}}(b_1,\ldots,b_{k+l})) + 1$$

finishes the proof of this case.

2. Finally, assume $f = \mathsf{PREC}[\mathsf{h}] \in \mathcal{F}^{3,2}$. Again the general case where $\mathsf{f} \in \mathcal{F}$ follows by same reasoning. Assume a normalizing derivation starting from $\mathsf{f}(t_{z'}, t_b, t_x; t_y, t_a)$ where the intermediate subterms $t_{z'}, t_b, t_x; t_y, t_a$ satisfy the assumptions of the lemma with bounds $b_{z'}, b_b, b_x; b_y, b_a$. The first, second and last argument to f have to be normalized first. Hence the considered derivation begins by

$$\mathsf{f}(t_{z'}, t_b, t_x; t_y, t_a) \to^* \mathsf{f}(\underline{z'}, \underline{b}, t_x^{(1)}; t_y^{(1)}, \underline{a})$$

Then for $z', b, a \in \mathbb{W}$ it holds that $|\underline{z'}| \leq b_{z'}$ since $t_{z'} \to^* \underline{z'}$ and by same reasoning, $|\underline{b}| \leq b_b$, $|t_x^{(1)}| \leq b_x$, $|t_y^{(1)}| \leq b_y$ and $|\underline{a}| \leq b_a$. Without loss of generality this sequence continues by one application of rule 21

$$\begin{aligned}\mathsf{f}(\underline{z'}, \underline{b}, t_x^{(1)}; t_y^{(1)}, \underline{a}) &\to \mathsf{f}(\mathsf{P}(;\underline{z'}), \mathsf{S}(\underline{b};), t_x^{(1)}; t_y^{(1)}, \mathsf{h}(\underline{b}, t_x^{(1)}; t_y^{(1)}, \underline{a})) \\ &\to^+ \mathsf{f}(\underline{z}, \underline{b'}, t_x^{(2)}; t_y^{(2)}, \underline{h(\underline{b}, t_x; t_y, \underline{a})})\end{aligned}$$

where $\mathsf{h}(\underline{b}, t_x^{(1)}; t_y^{(1)}, \underline{a}) \to^* \underline{\mathsf{h}(\underline{b}, t_x; t_y, \underline{a})}$ by confluence of $\mathcal{S}$. By Lemma 3.3 and definition of the algebra π, $\underline{\mathsf{h}(\underline{b}, t_x; t_y, \underline{a})} = \underline{\mathsf{h}(\underline{b}, t_x; t_y, \mathsf{f}(\underline{\epsilon}, \underline{b}, t_x; t_y, \underline{a}))} = \underline{\mathsf{f}(\underline{0}, \underline{b}, t_x; t_y, \underline{a})}$.

Consequently, by definition of π and iterated application of Lemma 3.3, for $c \in \mathbb{W}$ the c-th application of rule 21 in the considered derivation is of shape

$$\begin{aligned}&\mathsf{f}(\underline{z' \dot{-} c}, \underline{b + c}, t_x^{(3)}; t_y^{(3)}, \underline{\mathsf{f}(\underline{c}, \underline{b}, t_x; t_y, \underline{a})}) \\ &\quad\to \mathsf{f}(\mathsf{P}(;\underline{z' \dot{-} c}), \mathsf{S}(\underline{b + c};), t_x^{(3)}; t_y^{(3)}, h(\underline{b + c}, t_x^{(3)}; t_y^{(3)}, \underline{\mathsf{f}(\underline{c}, \underline{b}, t_x; t_y, \underline{a})})) \\ &\quad\to^+ \mathsf{f}(\underline{z' \dot{-} c'}, \underline{b + c'}, t_x^{(4)}; t_y^{(4)}, \underline{h(b + c, t_x; t_y, \mathsf{f}(\underline{c}, \underline{b}, t_x; t_y, \underline{a}))}) \\ &\quad= \mathsf{f}(\underline{z' \dot{-} c'}, \underline{b + c'}, t_x^{(4)}; t_y^{(4)}, \underline{\mathsf{f}(\underline{c'}, \underline{b}, t_x; t_y, \underline{a})})\end{aligned}$$

where the equality follows again by Lemma 3.3 and π. Likewise the last application of rule 21 is of shape

$$\begin{aligned}&\mathsf{f}(\underline{z' \dot{-} z}, \underline{b + z}, t_x^{(5)}; t_y^{(5)}, \underline{\mathsf{f}(\underline{z}, \underline{b}, t_x; t_y, \underline{a})}) \\ &\quad\to \mathsf{f}(\mathsf{P}(;\underline{z' \dot{-} z}), \mathsf{S}(\underline{b + z};), t_x^{(5)}; t_y^{(5)}, h(\underline{b + z}, t_x^{(5)}; t_y^{(5)}, \underline{\mathsf{f}(\underline{z}, \underline{b}, t_x; t_y, \underline{a})})) \\ &\quad\to^+ \mathsf{f}(\underline{\epsilon}, \underline{b + z'}, t_x^{(6)}; t_y^{(6)}, \underline{\mathsf{f}(\underline{z'}, \underline{b}, t_x; t_y, \underline{a})}) \\ &\quad\to \underline{\mathsf{f}(\underline{z'}, \underline{b}, t_x; t_y, \underline{a})}\end{aligned}$$

Any occuring term, except the last, is of shape $\mathsf{f}(t_1, t_2, t_3; t_4, t_5)$ for some terms $t_1, \ldots, t_5$. Since the last term is embedded in the previous term, we can construct a polynomial p_{f} satisfying the lemma by summing up appropriate length bounds for the intermediate subterms and add the length of f. Assume $c \leq z$. As we have already shown there exists a monotone polynomial p_{P} for P satisfying the lemma. Since $|\underline{z' \dot{-} c}| < |\underline{z'}| + 1 = |\,\mathsf{P}(;\underline{z'})|$ and $|\underline{z'}| \leq b_{z'}$ the first argument to f is bounded in length by $p_{\mathsf{P}}(b_{z'})$. Similar, since $|\underline{b + c}| \leq |\underline{b + z'}| \leq |\underline{b}| + |\underline{z'}|$ the second argument is at most $p_{\mathsf{S}}(b_{z'} + b_b)$ in length. As for all j, $t_x^{(j)}$ derives from t_x the length of $t_x^{(j)}$ is bounded by b_x. Similar $t_y^{(j)}$ is bounded in length by b_y. By Lemma 4.10 there exists a monotone polynomial q_f such that

$$|\underline{\mathsf{f}(\underline{c}, \underline{b}, t_x; t_y, \underline{a})}| = |\underline{\mathsf{f}(\underline{c}, \underline{b}, \underline{t_x}; \underline{t_y}, \underline{a})}| \leq q_{\mathsf{f}}(|\underline{c}|, |\underline{b}|, |\underline{t_x}|, |\underline{t_y}|, |\underline{a}|)$$

By monotonicity of p_f, for any $c \leq z'$ we obtain

$$|\underline{\mathsf{f}(\underline{c}, \underline{b}, t_x; t_y, \underline{a})}| \leq q_{\mathsf{f}}(b_{z'}, b_b, b_x, b_y, b_a)$$

Since $|\,\mathsf{f}\,| > |\,\mathsf{g}\,|$, induction hypothesis is applicable for h. Therefore when $h(\underline{b + c}, t_x^{(j)}; t_y^{(j)}, \underline{\mathsf{f}(c, b, x; y, a)}) \to^* t$ then t is bounded by a monotone polynomial p_h satisfying the lemma. Thus when $c \leq z'$, by similar reasoning as before the length of the last argument to f and the unique normal form is bounded by

$$p_a(b_{z'}, b_b, b_x, b_y, b_a) := p_h\big(b_{z'} + b_b, b_x, b_y, q_{\mathsf{f}}(b_{z'}, b_b, b_x, b_y, b_a)\big)$$

Putting things together the lemma follows by choosing the polynomial p_f such that

$$\begin{aligned} p_{\mathsf{f}}(b_{z'}, b_b, b_x, b_y, b_a) := |\,\mathsf{f}\,| + p_{\mathsf{P}}(b_{z'}) + p_{\mathsf{S}}(b_{z'} + b_b) + b_x + b_y \\ + p_a(b_{z'}, b_b, b_x, b_y, b_a) \end{aligned}$$

□

Corollary 4.12. *Assume that $f \in \mathrm{PS}$ and $m_1, \ldots, m_k, m_1, \ldots, m_l \in \mathbb{W}$. Then there exists a polynomial p_f satisfying*

$$\max\{|s| \mid \mathsf{f}(\underline{m}_1, \ldots, \underline{m}_k; \underline{n}_1, \ldots, \underline{n}_l) \to^* s\} \leq p_{\mathsf{f}}(|m_1|, \ldots, |m_k|, |n_1|, \ldots, |n_l|)$$

Proof. Since for every $m \in \mathbb{W}$, $\underline{m}$ is a normal form with respect to $\mathcal{S}$ we obtain $\max\{|t| \mid \underline{m} \to^* t\} = |\underline{m}| = |m|$. Hence the claim follows by choosing the polynomial obtained from Theorem 4.11. □

Theorem 4.13. *For every* $\mathrm{f} \in \mathrm{PS}$ *any $\mathcal{S}$-reduction yields an algorithm for* f *running in polynomial space.*

Proof. We have already observed that for every function f ∈ PS there exists a function symbol $\mathsf{f} \in \mathcal{F}$ such that any normalizing reduction

$$\mathsf{f}(\underline{n}_1, \ldots, \underline{n}_k; \underline{m}_1, \ldots, \underline{m}_l) = t_0 \to_{\mathcal{S}} t_1 \to_{\mathcal{S}} \ldots \to_{\mathcal{S}} t_n = \underline{a}$$

results in a numeral $\underline{a}$ satisfying $a = \mathrm{f}(m_1, \ldots, m_k; n_1, \ldots, n_l)$ for $a \in \mathbb{W}$. Thus any $\mathcal{S}$-reduction yields an algorithm for the functions in PS. By Corollary 4.12, the length of every ground term t_i in the above rewrite sequence is bounded by a polynomial p_f in the length of $m_1, \ldots, m_k, n_1, \ldots, n_l \in \mathbb{W}$.

By folklore, there exists an algorithm that rewrites every ground term into its reduct which runs in polynomial time and hence in polynomial space. It is easy to see that based on this, we can construct an algorithm with the desired space bounding property that iteratively calculates $t_1, t_2, \ldots, \underline{a}$ and finally returns a. □

5. Conclusion

Following [7, 8] we have shown how to obtain from the function theoretic characterization FPSPACE a term rewriting characterization $\mathcal{S}$ of the polyspace computable functions.

The here presented results mainly differ from [7, 8] in the following cases. Instead of referring to the resource-bound free characterization given in [7] we have proven in Appendix B and C directly that the class PS coincides with FPSPACE. Only a indirect proof of this fact is given in [8]. It is obvious to see that the recursion scheme of generalized safe primitive recursion closes PS under safe recursion on notation. We present a proof of this in Appendix A. As opposed to [8] we therefore dropped the second scheme from the definition of PS. This results in a simplified term rewriting characterization. In [8] termination of the term rewrite characterization has been shown using complex polynomial interpretations. We provide a simplified and very natural proof of this fact by means of semantic labeling. The main theorem of [8] states that for all function symbols $\mathsf{f} \in \mathcal{F}$, every term in a rewrite sequence starting from $\mathsf{f}(\underline{m}_1, \ldots, \underline{m}_k, \underline{n}_1, \ldots, \underline{n}_l)$ is bounded in length by a polynomial in the length of $\mathsf{f}(\underline{m}_1, \ldots, \underline{m}_k, \underline{n}_1, \ldots, \underline{n}_l)$, provided that $\underline{m}_1, \ldots, \underline{m}_k, \underline{n}_1, \ldots, \underline{n}_l$ are numerals. This amounts to the conclusion given in Corollary 4.12. In the proof of this theorem, Oitavem states that by induction on the definition of $\mathcal{F}$, the claim can be shown. Then only the case where $\mathsf{f} = \mathsf{PREC}[\mathsf{h}] \in \mathcal{F}$ is given. The other cases are considered trivial. However, if we consider a function symbol $\mathsf{SUB}[\mathsf{g}, \mathbf{r}, \mathbf{s}]$, denoting a function defined by safe composition, we have to assume an arbitrary derivation of the form

$$\begin{aligned}\mathsf{f}(\underline{m}_1, \ldots, \underline{m}_k; \underline{n}_1, \ldots, \underline{n}_l) &\to \mathsf{g}(\mathbf{r}(\underline{n}_1, \ldots, \underline{n}_k;); \mathbf{s}(\underline{m}_1, \ldots, \underline{m}_k; \underline{n}_1, \ldots, \underline{n}_l)) \\ &\to^* \mathsf{g}(\mathbf{t_r}; \mathbf{t_s}) \\ &\to^* \underline{\mathsf{f}(\underline{m}_1, \ldots, \underline{m}_k; \underline{n}_1, \ldots, \underline{n}_l)}\end{aligned}$$

Since terms $\mathbf{t_r}, \mathbf{t_s}$ are not necessarily numerals, it is not clear how induction hypothesis can be applied to conclude the claim. We have rectified this

by reformulating the theorem for arbitrary terms $\mathsf{f}(r_1, \ldots, r_k; s_1, \ldots, s_l)$, provided that terms occurring in derivations of $r_1, \ldots, r_k, s_1, \ldots, s_l$ are bounded in length. Finally, Oitavem points out the importance of the bound established for functions in PS: for all $\mathrm{f} \in \mathrm{PS}$ there exists a monotone polynomial p_f such that $|\mathrm{f}(\mathbf{x}; \mathbf{y})| \leq \max\{p_f(|\mathbf{x}|), \max|\mathbf{y}|\}$ (cf. Lemma C.1). It is stated that this bound is essential for the proof of the main theorem, a bound like $|\mathrm{f}(\mathbf{x}; \mathbf{y})| \leq p_f(|\mathbf{x}|) + |\mathbf{y}|$ is inadmissible. This is not the case, as we have shown in the proof of Theorem 4.11.

A. Closure Properties of PS

In this section we show that PS is closed under safe primitive recursion and safe recursion on notation. Although this is not mandatory, having these closure properties at hand helps in presenting the proofs in the remaining sections. For this, let us prove the following lemma first.

Lemma A.1. *Assume that* f *is defined by generalized safe recursion on* h*, i.e.* $f = PREC[h]$*. Then for all* $b, \mathbf{x}, \mathbf{y}, a, z \in \mathbb{W}$*,*

$$\mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) = \mathrm{h}(b + z, \mathrm{f}(z, b, \mathbf{x}; \mathbf{y}, a))$$

More general, for all $c \in \mathbb{W}$ *such that* $z \geq c$ *it holds that*

$$\mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) = \mathrm{f}(z' \dot{-} c', b + c', \mathbf{x}; \mathbf{y}, h(b + c, \mathrm{f}(c, b, \mathbf{x}; \mathbf{y}, a)))$$

Proof. We proof the general case

$$z \geq c \Rightarrow \mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) = \mathrm{f}(z' \dot{-} c', b + c', \mathbf{x}; \mathbf{y}, h(b + c, \mathrm{f}(c, b, \mathbf{x}; \mathbf{y}, a)))$$

by induction on c. This implies

$$\begin{aligned}
\mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) &= \mathrm{f}(z' \dot{-} z', b + z', \mathbf{x}; \mathbf{y}, h(b + z, \mathrm{f}(z, b, \mathbf{x}; \mathbf{y}, a))) \\
&= \mathrm{f}(\epsilon, b + z', \mathbf{x}; \mathbf{y}, h(b + z, \mathrm{f}(z, b, \mathbf{x}; \mathbf{y}, a))) \\
&= h(b + z, \mathrm{f}(z, b, \mathbf{x}; \mathbf{y}, a))
\end{aligned}$$

If $c = \epsilon$ then then $z' \dot{-} \epsilon' = z$, $b + \epsilon' = b'$ and $b + \epsilon = b$ so

$$\begin{aligned}
f(z, b', \mathbf{x}; \mathbf{y}; \mathrm{h}(b, \mathbf{x}; \mathbf{y}, f(\epsilon, b, \mathbf{x}; \mathbf{y}, a))) &= f(z, b', \mathbf{x}; \mathbf{y}, \mathrm{h}(b, \mathbf{x}; \mathbf{y}, a)) \\
&= f(z', b, \mathbf{x}; \mathbf{y}, a)
\end{aligned}$$

For the inductive step, assume the lemma has been shown for c. Instantiating z with c results then in

$$\begin{aligned}
f(c', b, \mathbf{x}; \mathbf{y}, a) &= f(c' \dot{-} c', b + c', \mathbf{x}; \mathbf{y}; \mathrm{h}(b + c, f(c, b, \mathbf{x}; \mathbf{y}, a))) \\
&= \mathrm{h}(b + c, f(c, b, \mathbf{x}; \mathbf{y}, a))
\end{aligned}$$

and therefore, for $z \geq c'$

$$\begin{aligned}
\mathrm{f}(z', b, \mathbf{x}; \mathbf{y}, a) &= \mathrm{f}(z' \dot{-} c', b + c', \mathbf{x}; \mathbf{y}; \mathrm{h}(b + c, \mathrm{f}(c, b, \mathbf{x}; \mathbf{y}, a))) \\
&= \mathrm{f}(z' \dot{-} c', b + c', \mathbf{x}; \mathbf{y}; \mathrm{f}(c', b, \mathbf{x}; \mathbf{y}, a)) \\
&= \mathrm{f}(z' \dot{-} c'', b + c'', \mathbf{x}; \mathbf{y}; \mathrm{h}(b + c', \mathrm{f}(c', b, \mathbf{x}; \mathbf{y}, a)))
\end{aligned}$$

where the first equation is due to induction hypothesis, the second equation follows from the the previous observation and the last by unfolding the definition. This concludes the claim. □

Lemma A.2. PS *is closed under safe primitive recursion.*

Proof. Assume that f is defined via the equations

$$\begin{aligned} \mathrm{f}(\epsilon, \mathbf{x}; \mathbf{y}) &= \mathrm{g}(\mathbf{x}; \mathbf{y}) \\ \mathrm{f}(z', \mathbf{x}; \mathbf{y}) &= \mathrm{h}(z, \mathbf{x}; \mathbf{y}, \mathrm{f}(z, \mathbf{x}; \mathbf{y})) \end{aligned}$$

where $g, h \in \mathrm{PS}$. We show that for all $z, \mathbf{x}, \mathbf{y} \in \mathbb{W}$ we have $\mathrm{f}(z, \mathbf{x}; \mathbf{y}) = \hat{\mathrm{f}}(z, \mathbf{x}; \mathbf{y})$ where $\hat{\mathrm{f}}$ is defined as follows:

$$\hat{\mathrm{f}}(z, \mathbf{x}; \mathbf{y}) = PREC[h](z, \epsilon, \mathbf{x}; \mathbf{y}, \mathrm{g}(\mathbf{x}; \mathbf{y}))$$

To prove the claim, we proceed by induction on z. If $z = \epsilon$ then

$$\hat{\mathrm{f}}(\epsilon, \mathbf{x}; \mathbf{y}) = PREC[h](\epsilon, \epsilon, \mathbf{x}; \mathbf{y}, \mathrm{g}(\mathbf{x}; \mathbf{y})) = \mathrm{g}(\mathbf{x}; \mathbf{y})$$

as wished. Else assume $z > \epsilon$. Then

$$\begin{aligned} \mathrm{f}(z', \mathbf{x}; \mathbf{y}) &= \mathrm{h}(z, \mathbf{x}; \mathbf{y}, \mathrm{f}(z, \mathbf{x}; \mathbf{y})) \\ &= \mathrm{h}(z, \mathbf{x}; \mathbf{y}, \hat{\mathrm{f}}(z, \mathbf{x}; \mathbf{y})) \\ &= \mathrm{h}(z, \mathbf{x}; \mathbf{y}, PREC[h](z, \epsilon, \mathbf{x}; \mathbf{y}, \mathrm{g}(\mathbf{x}; \mathbf{y}))) \\ &= PREC[h](z', \epsilon, \mathbf{x}; \mathbf{y}, \mathrm{g}(\mathbf{x}; \mathbf{y})) \end{aligned}$$

where the second equality follows by induction hypothesis and the last equality follows by Corollary 3.3. □

Lemma A.3. PS *is closed under safe recursion on notation.*

Proof. Assume that f is defined via the equations

$$\begin{aligned} \mathrm{f}(\epsilon, \mathbf{x}; \mathbf{y}) &= \mathrm{g}(\mathbf{x}; \mathbf{y}) \\ \mathrm{f}(zi, \mathbf{x}; \mathbf{y}) &= h_i(z, \mathbf{x}; \mathbf{y}, \mathrm{f}(z, \mathbf{x}; \mathbf{y})) \end{aligned}$$

where $g, h \in \mathrm{PS}$ and $i \in \{0, 1\}$. We show that there exists a function $\hat{\mathrm{f}} \in \mathrm{PS}$ such that for all $z, \mathbf{x}, \mathbf{y} \in \mathbb{W}$, $\mathrm{f}(z, \mathbf{x}; \mathbf{y}) = \hat{\mathrm{f}}(z, z, \mathbf{x}; \mathbf{y})$ holds.

Intuitively, $\hat{\mathrm{f}}$ may be described as follows. If the first argument of $\hat{\mathrm{f}}$ is ϵ then $\hat{\mathrm{f}}(\epsilon, z, \mathbf{x}; \mathbf{y})$ will be $\mathrm{g}(\mathbf{x}; \mathbf{y})$. Each call of $\hat{\mathrm{f}}(u, z, \mathbf{x}; \mathbf{y})$ will result in $\mathrm{f}(u, \mathbf{x}; \mathbf{y})$ in the case that $u \subseteq z$. If $u \not\subseteq z$ then the call will result in $\mathrm{f}(v, \mathbf{x}; \mathbf{y})$ where v is the first predecessor of u such that $v \subseteq z$. Therefore $\hat{\mathrm{f}}$ skips all recursion steps where the recursion value u is not a prefix of the recursion input to f.

To define $\hat{\mathrm{f}}$, some predicates and boolean operations are needed.

$$\begin{aligned} \#\mathrm{f}() &= \mathrm{C}_0(\epsilon;) & \mathrm{if}(; b, t, e) &= \mathrm{Q}(; b, \epsilon, e, t) \\ \#\mathrm{t}() &= \mathrm{C}_1(\epsilon;) & \text{empty?}(; e) &= \mathrm{Q}(; e, \#\mathrm{t}(), \#\mathrm{f}(), \#\mathrm{f}()) \\ \vee(; x, y) &= \mathrm{if}(; x, x, y) & \wedge(; x, y) &= \mathrm{if}(; x, y, x) \end{aligned}$$

We regard every numeral zi as true if $i = 1$ and false if $i = 0$, ϵ has no associated boolean value. $\#\mathrm{f}()$ returns therefore a false value and likewise $\#\mathrm{t}()$ returns a true value. The function $\vee$ performs a logical or on the given arguments, likewise $\wedge$ performs a logical and. Finally empty? returns only a true value if given ϵ.

To check if two numerals x, y equal, we first define the function $\mathrm{P}^*(x;y)$ that computes $y \dot{-} x$. Based on P^* we can then formulate predicates $\geq$ and $=$ following their intentional semantics.

$$\begin{aligned} \mathrm{P}^*(\epsilon;y) &= y & \geq(x;y) &= \mathrm{empty?}(;\mathrm{P}^*(x;y)) \\ \mathrm{P}^*(x';y) &= \mathrm{P}(;\mathrm{P}^*(x;y)) & =(x,y;) &= \wedge(;\geq(x;y),\geq(y;x)) \end{aligned}$$

Here P^* is defined by safe primitive recursion and is according to Lemma A.2 in PS. Next we need to define, for $x \neq \epsilon$, a predicate $\subseteq(x;y)$ that checks if $x \subseteq y$, i.e. there exists some z such that $xz = y$. In other words, x is a prefix of y if there exists some $n \leq |y|$ such that deleting n rightmost bits of y results in x.

$$\begin{aligned} \mathrm{D}^*(\epsilon;x) &= x & \subseteq'(\epsilon,x,y;) &= \#\mathrm{f}() \\ \mathrm{D}^*(y';x) &= \mathrm{D}(;\mathrm{D}^*(y;x)) & \subseteq'(z',x,y;) &= \vee(;=(x,\mathrm{D}^*(z';y);),\subseteq'(z,x,y;)) \\ & & \subseteq(x,y;) &= \subseteq'(y,x,y;) \end{aligned}$$

Again both functions D^* and $\subseteq$ are defined by safe primitive recursion. They are described as follows. $\mathrm{D}^*(z;y)$ performs z deletions on y. $\subseteq(x,y;)$ may be rephrased as $\bigvee_{z\leq y} x = \mathrm{D}^z(;y)$ where the index means iteration.

Finally we need the function I that extracts the last bit of a given numeral:

$$I(;z) = \mathrm{if}(;z,\mathrm{C}_0(\epsilon),\mathrm{C}_1(\epsilon))$$

The function $\hat{\mathrm{f}}$ is then defined by safe primitive recursion as follows:

$$\begin{aligned} \hat{\mathrm{f}}(\epsilon,z,\mathbf{x};\mathbf{y}) &= \mathrm{g}(\mathbf{x};\mathbf{y}) \\ \hat{\mathrm{f}}(c',z,\mathbf{x};\mathbf{y}) &= \hat{h}(c,z,\mathbf{x};\mathbf{y},\hat{\mathrm{f}}(c,z,\mathbf{x};\mathbf{y})) \\ \hat{h}(c,z,\mathbf{x};\mathbf{y},a) &= \mathrm{if}(;\subseteq(\mathrm{S}(c;),z), \\ &\qquad H(\mathrm{D}(;\mathrm{S}(c;)),I(;\mathrm{S}(c;)),\mathbf{x};\mathbf{y},a) \\ &\qquad a) \\ H(z,i,\mathbf{x};\mathbf{y},a) &= \mathrm{if}(;i,h_1(z,\mathbf{x};\mathbf{y},a),h_0(z,\mathbf{x};\mathbf{y},a)) \end{aligned}$$

By Lemma A.2, $\hat{\mathrm{f}}$ in PS. To prove that $\mathrm{f}(z,\mathbf{x};\mathbf{y}) = \hat{\mathrm{f}}(z,z,\mathbf{x};\mathbf{y})$ we fix $z, \mathbf{x}, \mathbf{y}$ and show by induction on u that $\hat{\mathrm{f}}(u,z,\mathbf{x};\mathbf{y}) = \mathrm{f}(v,\mathbf{x};\mathbf{y})$ where v is either u or the first predecessor of u that is a prefix of z. This implies then that $\mathrm{f}(z,\mathbf{x};\mathbf{y}) = \hat{\mathrm{f}}(z,z,\mathbf{x};\mathbf{y})$.

Assume $u = \epsilon$. Therefore $\hat{\mathrm{f}}(\epsilon,\mathbf{x};\mathbf{y}) = \mathrm{g}(\mathbf{x};\mathbf{y}) = \mathrm{f}(\epsilon,\mathbf{x};\mathbf{y})$. For the inductive step we distinguish two cases:

- Case $u' = ci \subseteq z$ for $i \in \{0,1\}$: Therefore

$$\begin{aligned} \hat{\mathrm{f}}(u',z,\mathbf{x};\mathbf{y}) &= \hat{h}(u,z,\mathbf{x};\mathbf{y},\hat{\mathrm{f}}(u,z,\mathbf{x};\mathbf{y})) \\ &= \hat{h}(u,z,\mathbf{x};\mathbf{y},\mathrm{f}(c,\mathbf{x};\mathbf{y})) \\ &= H(\mathrm{D}(;\mathrm{S}(u;)),I(;\mathrm{S}(u;)),\mathbf{x};\mathbf{y},\mathrm{f}(c,\mathbf{x};\mathbf{y})) \\ &= H(c,\mathrm{C}_i(\epsilon),\mathbf{x};\mathbf{y},\mathrm{f}(c,\mathbf{x};\mathbf{y})) \\ &= h_i(c,\mathbf{x};\mathbf{y},\mathrm{f}(c,\mathbf{x};\mathbf{y})) \\ &= \mathrm{f}(ci,\mathbf{x};\mathbf{y}) \end{aligned}$$

where in the second equation the induction hypothesis has been applied.

- Case $u' \not\sqsubseteq z$: Then $\subseteq(\mathrm{S}(;u);z)$ is a false value,

$$\begin{aligned}\hat{\mathrm{f}}(u', z, \mathbf{x}; \mathbf{y}) &= \hat{h}(u, z, \mathbf{x}; \mathbf{y}, \hat{\mathrm{f}}(u, z, \mathbf{x}; \mathbf{y})) \\ &= \hat{\mathrm{f}}(u, z, \mathbf{x}; \mathbf{y})\end{aligned}$$

and induction hypothesis is directly applicable.

□

B. PS contains FPSPACE

In this section we show that every function that is definable in FPSPACE is definable in PS with only normal argument positions. As pointed out earlier, we can not effectively use generalized safe recursion in this situation. The idea now is to construct for every function $f \in$ FPSPACE a function $\hat{\mathrm{f}}$ into PS such that $\hat{\mathrm{f}}(w; \mathbf{x}) = \mathrm{f}(\mathbf{x})$ provided that w is sufficiently large enough. Intuitively, large enough means at least as big as the maximum depth of recursion used in computing $\mathrm{f}(\mathbf{x})$. w can then be used to simulate recursion on the safe arguments of the involved functions.

Lemma B.1. *If $f \in$ FPSPACE then there exists a function $\hat{\mathrm{f}} \in$ PS and a polynomial p_f such that*

$$\forall \mathbf{x}, w \in \mathbb{W}.|w| \geq p_f(|\mathbf{x}|) \Rightarrow \mathrm{f}(\mathbf{x}) = \hat{\mathrm{f}}(w; \mathbf{x}) \qquad (*)$$

Proof. We proof this lemma by induction on the definition of FPSPACE.

1. If $f = \mathrm{E}^n$ we set $\hat{\mathrm{f}} := \mathrm{E}^{1,n}$. Likewise, if $f = \mathrm{P}^n_j$ we defining $\hat{\mathrm{f}} := \mathrm{P}^{1,n}_{j+1}$. In these cases, let $p_f(|\mathbf{x}|) = 0$ and the claim follows. If $f = \mathsf{C}_i$ then define $\hat{\mathrm{f}}_i(w, x) := \mathrm{B}_i(\epsilon, w; \mathrm{D}(; \mathrm{B}_0(\epsilon, w; x)))$ and $p_f(|x|) = |x| + 1$. Now assume that $|w| \geq p_f(|x|)$. Since $|w| \geq |x| + 1$ we have $\mathrm{D}(; \mathrm{B}_0(\epsilon, w; x)) = \mathrm{D}(; x^R 0) = x^R$. Therefore $\mathrm{B}_i(\epsilon, w; x^R) = xi$ and the claim follows.

   Next assume that $\mathrm{f}(x, y) = x \# y$. Then f can be defined in FPSPACE by bounded recursion on notation as follows:

$$\begin{aligned}\mathrm{g}(\epsilon, y) &= y \\ \mathrm{g}(xi, y) &= \mathrm{C}_0(\mathrm{g}(x, y)) \\ \mathrm{f}(\epsilon, y) &= \mathrm{C}_1(\epsilon) \\ \mathrm{f}(xi, y) &= \mathrm{g}(y, \mathrm{f}(x, y))\end{aligned}$$

   Here $\mathrm{g}(x, y)$ is bounded in length by polynomials $b_g(|x|, |y|) = |x| + |y|$ and $b_f(|x|, |y|) = |x| * |y| + 1$ $\mathrm{f}(xi, y)$. Hence we can construct bounding functions t_g, t_f for g, f and apply the same method as for bounded recursion on notation.

2. If f is defined by composition, i.e. $\mathrm{f}(\mathbf{x}) = h(g_1(\mathbf{x}), \ldots, g_m(\mathbf{x}))$. Then by the final observation in Section 2.1 there exist length-bounding monotone polynomials b_{g_i} bounding g_i. By induction hypothesis there exists $\hat{\mathrm{h}} \in \mathrm{PS}$ with polynomial p_h for h and $\hat{\mathrm{g}}_i \in \mathrm{PS}$ with polynomial p_{g_i} for every g_i satisfying the lemma. Hence we may define

$$\hat{\mathrm{f}}(w; \mathbf{x}) := \hat{\mathrm{h}}(w; \hat{\mathrm{g}}_1(w; \mathbf{x}), \ldots, \hat{\mathrm{g}}_m(w; \mathbf{x}))$$

with polynomial:

$$p_f(|\mathbf{x}|) = p_h(b_{g_1}(|\mathbf{x}|), \ldots, b_{g_m}(|\mathbf{x}|)) + \Sigma_i p_{g_i}(|\mathbf{x}|)$$

Assume that $|w| \geq p_f(|\mathbf{x}|)$. By monotonicity of the involved polynomials $|w| \geq p_{g_i}(|\mathbf{x}|)$ for all $i \in \{1, \ldots, m\}$ and further

$$\begin{aligned} |w| &\geq p_h(b_{g_1}(|\mathbf{x}|), \ldots, b_{g_m}(|\mathbf{x}|)) \\ &\geq p_h(|g_1(\mathbf{x})|, \ldots, |g_m(\mathbf{x})|) \end{aligned}$$

Therefore induction hypothesis is directly applicable and the desired result follows.

3. The interesting cases occur when $\mathrm{f}(y, \mathbf{x})$ is defined by bounded recursion on notation or bounded primitive recursion. Of course $\hat{\mathrm{f}}(w; y, \mathbf{x})$ can not be defined by recursion on y since y appears at a safe position of f. However, if w is sufficiently large enough we can use the "normal bits" of w to simulate recursion on y. For this, we introduce a parameter z that is initialized by w. Then we build a function $\hat{\mathrm{f}} \in \mathrm{PS}$ such that recursion of $\hat{\mathrm{f}}$ on z from w down to some u corresponds to recursion on y. For both recursion schemes we construct a function $\hat{\mathrm{f}}$ together with a polynomial p_f satisfying $\mathrm{f}(y, \mathbf{x}) = \hat{\mathrm{f}}(w, w; y, \mathbf{x})$ for all $y, \mathbf{x} \in \mathbb{W}$ provided that $|w| \geq p_f(|y|, |\mathbf{x}|)$. This implies then the claim of the lemma.

   Assume now that $\mathrm{f}(y, \mathbf{x})$ is defined by bounded recursion on notation, i.e.

$$\begin{aligned} \mathrm{f}(\epsilon, \mathbf{y}) &= \mathrm{g}(\mathbf{y}) \\ \mathrm{f}(xi, \mathbf{y}) &= h_i(x, \mathbf{y}, \mathrm{f}(x, \mathbf{y})) \end{aligned}$$

   for some $g, h_0, h_1 \in \mathrm{FPSPACE}$ and bounding function $t \in \mathrm{FPSPACE}$. The definition of $\hat{\mathrm{f}}$ relies on following auxiliary functions:

$$\begin{aligned} &\mathrm{D}_1^*(\epsilon; x) = x && Y(z, w; y) = \mathrm{D}_1^*(\mathrm{D}_2^*(z, w;); y) \\ &\mathrm{D}_1^*(yi; x) = \mathrm{D}(; \mathrm{D}_1^*(y; x)) && I(z, w; y) = \mathrm{Q}(; Y(z1, w; y), \epsilon, \mathrm{C}_0(\epsilon), \mathrm{C}_1(\epsilon)) \\ &\mathrm{D}_2^*(y, x;) = \mathrm{D}_2^*(y; x) \end{aligned}$$

   Here D_1^* is defined by safe recursion on notation and from Lemma A.3 we infer that $\mathrm{D}_1^* \in \mathrm{PS}$. Both $\mathrm{D}_1^*(x; y)$ and $\mathrm{D}_2^*(x, y;)$ perform $|x|$ deletion operations on y. Therefore, $Y(z, w; y)$ performs $|w| - |z|$ deletion operations on y. When z varies in length from $|w|$ to $|w| - |y|$, $Y(z, w; y)$ varies

from y to ϵ provided that $|w| \geq |y|$. Finally, for $j \in \{0,1\}$, the function I satisfies $Y(zj,w;y) = \mathrm{C}_{I(z,w;y)}(Y(z,w;y);)$.

At each recursion step of $\hat{\mathrm{f}}$, $Y(z,w;y)$ is used to produce an appropriate initial segment of y, likewise $I(z,w;y)$ is used to determine which stepping function h_i should be used. If $Y(z,w;y) = \epsilon$ then $\hat{\mathrm{f}}$ returns the result of $\mathrm{g}(\mathbf{x})$. As recursion depths with $|z|$ below $|w| - |y|$ are irrelevant, $\hat{\mathrm{f}}$ will return ϵ in this case.

By induction hypothesis we may assume that there exist functions $\hat{\mathrm{g}}$ with polynomial p_g for g and functions $\hat{\mathrm{h}}_i$ with polynomials p_{h_i} satisfying the lemma for h_i and $i \in \{0,1\}$. We define $\hat{\mathrm{f}}$ using safe recursion on notation.

$$\begin{aligned}
C(;g,t,e) &= \mathrm{Q}(;g,t,e,e)\\
H(w;i,y,\mathbf{x},a) &= \mathrm{Q}(;i,\epsilon,\hat{\mathrm{h}}_0(w;y,\mathbf{x},a),\hat{\mathrm{h}}_1(w;y,\mathbf{x},a))\\
\hat{\mathrm{f}}(\epsilon,w;y,\mathbf{x}) &= \epsilon\\
\hat{\mathrm{f}}(zi,w;y,\mathbf{x}) &= C(;Y(z1,w;y),\\
&\qquad \hat{\mathrm{g}}(w;\mathbf{x}),\\
&\qquad H(w;I(z,w;y),Y(z,w;y),\mathbf{x},\hat{\mathrm{f}}(z,w;y,\mathbf{x})))
\end{aligned}$$

As by Lemma A.3 PS is closed under safe recursion on notation $\hat{\mathrm{f}} \in \mathrm{PS}$. Finally, p_f is defined by:

$$\begin{aligned}
p_h(|y|,|\mathbf{x}|) &:= p_{h_0}(|y|,|\mathbf{x}|) + p_{h_1}(|y|,|\mathbf{x}|) \text{ and}\\
p_f(|y|,|\mathbf{x}|) &:= p_h(|y|,|\mathbf{x}|,b_f(|y|,|\mathbf{x}|)) + p_g(|\mathbf{x}|) + |y|
\end{aligned}$$

where b_f is a monotone bounding-polynomial for $f \in \mathrm{FPSPACE}$.

We fix $y,\mathbf{x}$ and w such that $|w| \geq p_f(|y|,|\mathbf{x}|)$. Hence $|w| \geq |y|$. We show by side induction on $|u|$ that for $|w| - |y| \leq |u| \leq |w|$ we have:

$$\hat{\mathrm{f}}(u,w;y,\mathbf{x}) = \mathrm{f}(Y(u,w;y),\mathbf{x})$$

As $Y(w,w;y) = y$ this implies that $\hat{\mathrm{f}}(w,w;y,\mathbf{x}) = \mathrm{f}(y,\mathbf{x})$.

Let u be such that $|w|-|y| \leq |u| \leq |w|$. If $|u| = |w|-|y|$ then $Y(u,w;y) = \epsilon$ and so $\hat{\mathrm{f}}(u,w;y,\mathbf{x}) = \hat{\mathrm{g}}(w;\mathbf{x}) = \mathrm{g}(\mathbf{x}) = \mathrm{f}(Y(u,w;y),\mathbf{x})$ as wished.

Else assume $u = zj$ for $j \in \{0,1\}$ and $|w| - |y| < |u| \leq |w|$. By monotonicity of the involved polynomials we have

$$\begin{aligned}
|w| &\geq p_f(|y|,|\mathbf{x}|)\\
&\geq p_h(|y|,|\mathbf{x}|,b_f(|y|,|\mathbf{x}|))\\
&\geq p_{h_i}(|Y(z,w;y)|,|\mathbf{x}|,b_f(|Y(z,w;y)|,|\mathbf{x}|))\\
&\geq p_{h_i}(|Y(z,w;y)|,|\mathbf{x}|,|f(Y(z,w;y),\mathbf{x})|)
\end{aligned}$$

By side induction hypothesis we may assume $\hat{\mathrm{f}}(z,w;y,\mathbf{x}) = \mathrm{f}(Y(z,w;y),\mathbf{x})$. Hence

$$\begin{aligned}
\hat{\mathrm{h}}_i(w;Y(z,w;y),\mathbf{x},\hat{\mathrm{f}}(z,w;y,\mathbf{x})) &= \hat{\mathrm{h}}_i(w;Y(z,w;y),\mathbf{x},\mathrm{f}(Y(z,w;y);y,\mathbf{x}))\\
&= h_i(Y(z,w;y),\mathbf{x},\mathrm{f}(Y(z,w;y);y,\mathbf{x}))
\end{aligned}$$

for $i \in \{0,1\}$. Here the second equation follows by main induction hypothesis on h_0 and h_1. Since $|w| \dot{-} |y| < |zj|$ implies $Y(zj,w;y) = Y(z1,w;y) \neq \epsilon$ and $Y(zj,w;y) = \mathrm{C}_{I(z,w;y)}(Y(z,w;y);)$ we finally obtain

$$\begin{aligned}
\hat{\mathrm{f}}(zj,w;y,\mathbf{x}) &= H(w;I(z,w;y),Y(z,w;y),\mathbf{x},\hat{\mathrm{f}}(z,w;y,\mathbf{x})) \\
&= \hat{\mathrm{h}}_{I(z,w;y)}(w;Y(z,w;y),\mathbf{x},\hat{\mathrm{f}}(z,w;y,\mathbf{x})) \\
&= h_i(Y(z,w;y),\mathbf{x},\mathrm{f}(Y(z,w;y);y,\mathbf{x})) \\
&= \mathrm{f}(Y(zj,w;y);y,\mathbf{x})
\end{aligned}$$

as desired.

4. Finally assume that f is defined by bounded primitive recursion, i.e.

$$\begin{aligned}
\mathrm{f}(\epsilon,\mathbf{x}) &= \mathrm{g}(\mathbf{y}) \\
\mathrm{f}(y',\mathbf{x}) &= h(y,\mathbf{x},\mathrm{f}(y,\mathbf{x}))
\end{aligned}$$

for some $g,h \in \mathrm{FPSPACE}$ and bounding function $t \in \mathrm{FPSPACE}$. We proceed similar as in the case for bounded recursion on notation. Y is this time defined as follows:

$$\begin{aligned}
\mathrm{P}_1^*(\epsilon;x) &= x & \mathrm{P}_1^*(y';x) &= \mathrm{P}_1^*(y;\mathrm{P}(;x)) \\
\mathrm{P}_2^*(y,x;) &= \mathrm{P}_1^*(y;x) & Y(z,w;y) &= \mathrm{P}_1^*(\mathrm{P}_2^*(z,w;);y)
\end{aligned}$$

Notice that $\mathrm{P}_2^*(x,y;) = \mathrm{P}_1^*(x;y) = y \dot{-} x$. Therefore $Y(z,w;y) = y \dot{-} (w \dot{-} z)$. If $z = w$ then $\mathrm{P}_2^*(z,w;) = w \dot{-} z = \epsilon$ and therefore $Y(z,w;y) = \mathrm{P}_1^*(\epsilon;y) = y$. Provided that $w \geq y$, when z decreased to $w \dot{-} y$ then $\mathrm{P}_2^*(z,w;) = w \dot{-} z = w \dot{-} (w \dot{-} y) = (w \dot{-} w) + y = y$ and so $Y(z,w;y) = \epsilon$ in this case. We define

$$\begin{aligned}
C(;g,t,e) &= \mathrm{Q}(;g,t,e,e) \\
\hat{\mathrm{f}}(\epsilon,w;y,\mathbf{x}) &= \epsilon \\
\hat{\mathrm{f}}(z',w;y,\mathbf{x}) &= C(;Y(\mathrm{S}(;z),w;y), \\
&\qquad \hat{\mathrm{g}}(w;\mathbf{x}), \\
&\qquad \hat{\mathrm{h}}(w;Y(z,w;y),\mathbf{x},\hat{\mathrm{f}}(z,w;y,\mathbf{x})))
\end{aligned}$$

where $\hat{\mathrm{g}},\hat{\mathrm{h}}$ are given by induction hypothesis on g,h. We define

$$p_f(|y|,|\mathbf{x}|) := p_h(|y|,|\mathbf{x}|,b_f(|y|,|\mathbf{x}|)) + p_g(|\mathbf{x}|) + |y| + 1$$

where b_f is the bounding polynomial of $f \in \mathrm{FPSPACE}$ and p_g, p_h are given by induction hypothesis.

Let $y,\mathbf{x},w \in \mathbb{W}$ such that $|w| \geq p_f(|y|,|\mathbf{x}|)$. Notice that $|w| \geq p_f(|y|,|\mathbf{x}|) \geq |y| + 1$ implies $w > y$. We prove by induction on u that for $w \dot{-} y \leq u \leq w$ we have:

$$\hat{\mathrm{f}}(u,w;y,\mathbf{x}) = \mathrm{f}(Y(u,w;y),\mathbf{x})$$

If $u = w\dot{-}y$ then $Y(u, w; y) = y\dot{-}(w\dot{-}(w\dot{-}y)) = \epsilon$ and so $\hat{\mathrm{f}}(u, w; y, \mathbf{x}) = \hat{\mathrm{g}}(w; \mathbf{x}) = \mathrm{g}(\mathbf{x}) = \mathrm{f}(Y(u, w; y), \mathbf{x})$ by main induction hypothesis on g.

For the inductive step, assume $w\dot{-}y < u \leq w$. By side induction hypothesis we have $\hat{\mathrm{f}}(z, w; y, \mathbf{x}) = \mathrm{f}(Y(z, w; y), \mathbf{x})$ where $z' = u$. Further

$$\begin{aligned} |w| &\geq p_f(|y|, |\mathbf{x}|) \\ &\geq p_h(|Y(z, w; y)|, |\mathbf{x}|, b_f(|Y(z, w; y)|, |\mathbf{x}|)) \\ &\geq p_h(|Y(z, w; y)|, |\mathbf{x}|, |\,\mathrm{f}(Y(z, w; y), \mathbf{x})|) \end{aligned}$$

by monotonicity of p_f and p_h. Assuming $w \geq y$, $z' = u > w\dot{-}y$ implies $Y(z', w; y) = Y(\mathrm{S}(; z), w; y) \neq \epsilon$. Therefore

$$\begin{aligned} \hat{\mathrm{f}}(z', w; y, \mathbf{x}) &= \hat{\mathrm{h}}(w; Y(z, w; y), \mathbf{x}, \hat{\mathrm{f}}(z, w; y, \mathbf{x})) \\ &= \hat{\mathrm{h}}(w; Y(z, w; y), \mathbf{x}, \mathrm{f}(Y(z, w; y), \mathbf{x})) \\ &= \mathrm{h}(Y(z, w; y), \mathbf{x}, \mathrm{f}(Y(z, w; y), \mathbf{x})) \end{aligned}$$

where main induction hypothesis for $\hat{\mathrm{h}}$ and side induction hypothesis has been applied.

This completes the proof. □

Theorem B.2. *Let* $\mathrm{f}(\mathbf{x})$ *be in* FPSPACE. *Then* $\mathrm{f}(\mathbf{x};)$ *is in* PS.

Proof. Let p_f and $\hat{\mathrm{f}}$ be obtained by the previous lemma. We construct a function $b(\mathbf{x};)$ that produces a binary word w such that $|w| \geq p_f(|\mathbf{x}|)$. Then by setting

$$\mathrm{f}(\mathbf{x};) := \hat{\mathrm{f}}(b(\mathbf{x};); \mathbf{x})$$

we obtain the desired result.

Without loss of generality we may assume the existence of $c, d \in \mathbb{N}$ such that $(\Sigma_i |x_i|)^d + c \geq p_f(|\mathbf{x}|)$ for all $\mathbf{x}$. Therefore it suffice to construct b such that $|b(\mathbf{x};)| \geq (\Sigma_i |x_i|)^d + c$.

First we construct functions $\oplus_i$ for $i \geq 1$, taking i normal arguments, such that $|\oplus_i(x_1, \ldots, x_i)| = \Sigma_{j=1}^{i} |x_j|$.

$$\begin{aligned} \oplus_1(x_1;) &= x_1 \\ \oplus_2(x_1, x_2;) &= \mathrm{B}_1(x_1, x_2; x_2) \\ \oplus_{k+1}(x_1, \ldots, x_{k+1};) &= \mathrm{B}_1(\oplus_k(x_1, \ldots, x_k;), x_{k+1}; x_{k+1}) \end{aligned}$$

Next we define unary functions $\#_i$ for each $i \geq 1$.

$$\begin{aligned} \#_1(x;) &= x & \#_{k+1}(x;) &= \times(\#_k(x;), x;) \\ \#_2(x;) &= \times(x, x;) \end{aligned}$$

$\#_k(x;)$ produces a string of length $|x|^k$. Using this definition we may construct b as follows:

$$b(x_1, \ldots, x_n;) := \mathrm{C}_1^c(\#_d(\oplus_n(x_1, \ldots, x_n;););)$$

Here C_1^c denotes the function obtained by c compositions of C_1. As already stated $|\oplus_n(x_1, \ldots, x_n;)| = \Sigma_i |x_i|$ and hence $|\#_d(\oplus_n(x_1, \ldots, x_n;);)| = (\Sigma_i |x_i|)^d$. Finally $|\,\mathrm{C}_1^c(\#_d(\oplus_i(x_1, \ldots, x_i;););)| = (\Sigma_i |x_i|)^d + c$. This concludes the claim. □

C. FPSPACE contains PS

Lemma C.1. *If $f \in$ PS then there exists a monotone polynomial p_f such that*

$$\forall \mathbf{x} \forall \mathbf{y} \in \mathbb{W}.|\,\mathrm{f}(\mathbf{x};\mathbf{y})| \leq \max\{p_f(|\mathbf{x}|), \max_i |y_i|\} \qquad (**)$$

Proof. The proof is by induction on the definition on PS.

1. If f is a constant, i-concatenation, deletion, successor, predecessor, projection or modified concatenation function then we set $q_f(|\mathbf{x}|) = 1+\Sigma_i|x_i|$ and easily verify that for all $\mathbf{x}, \mathbf{y}$, $|\,\mathrm{f}(\mathbf{x};\mathbf{y})| \leq \ max\{q_f(|\mathbf{x}|), \max_i |y_i|\}$.

2. If f is defined by safe composition then we may assume by induction hypothesis the existence of monotone polynomials $q_h, \mathbf{q_r}, \mathbf{q_s}$ bounding $h, \mathbf{r}$ and $\mathbf{s}$ respectively. Then

$$\begin{aligned}
|\,\mathrm{f}(\mathbf{x};\mathbf{y})| &= |\,\mathrm{h}(\mathbf{r}(\mathbf{x};);\mathbf{s}(\mathbf{x};\mathbf{y}))| \\
&\leq \max\{q_h(\mathbf{r}(\mathbf{x};)), \max_i |s_i(\mathbf{x};\mathbf{y})|\} \\
&\leq \max\big\{q_h(\mathbf{q_r}(|\mathbf{x}|)), \max_i\ q_{s_i}(|\mathbf{x}|), \max_j |y_j|\big\} \\
&\leq \max\big\{q_h(\mathbf{q_r}(|\mathbf{x}|)) + \Sigma_i q_{s_i}(|\mathbf{x}|), \max_j |y_j|\big\}
\end{aligned}$$

   which follows by applying induction hypothesis and by monotonicity of the polynomials. Therefore we may choose q_f such that $q_f(|\mathbf{x}|) = q_h(\mathbf{q_r}(|\mathbf{x}|))+ \Sigma_i q_{s_i}(|\mathbf{x}|)$.

3. Finally assume that f is defined by generalized safe primitive recursion. By induction hypothesis we may assume that there exists monotone polynomials q_{h_0} for h_0 and q_{h_1} for h_1 such that (**) holds. We define q_f as follows:

$$q_f(|z|, |b|, |\mathbf{x}|) = q_h(|z| + |b|, |\mathbf{x}|)$$

   If $z = \epsilon$ then the result is immediate. Else assume $z = c'$ and the lemma has been established for c.

$$\begin{aligned}
|\,\mathrm{f}(c', b, \mathbf{x};\mathbf{y}, a)| &= |\,\mathrm{h}(b + c, \mathbf{x};\mathbf{y}, \mathrm{f}(c, b, \mathbf{x};\mathbf{y}, a))| \\
&\leq \max\{q_h(|b + c|, |\mathbf{x}|), \max_i |y_i|, |\,\mathrm{f}(c, b, \mathbf{x};\mathbf{y}, a)|\} \\
&\leq \max\big\{q_h(|b + c|, |\mathbf{x}|), \max_i |y_i|, \\
&\qquad \max\{q_f(|c|, |b|, |\mathbf{x}|), \max_i |y_i|, |a|\}\big\} \\
&\leq \max\{q_h(|c'| + |b|, |\mathbf{x}|), \max_i |y_i|, q_f(|c|, |b|, |\mathbf{x}|), |a|\} \\
&\leq \max\{q_f(|c'|, |b|, |\mathbf{x}|), \max_i |y_i|, |a|\}
\end{aligned}$$

   Here the first inequality follows by Corollary 3.3 and the second two inequalities follow by induction hypothesis on h and f respectively. For the fourth inequality we use that for c and $b \in \mathbb{W}$, $|b + c| \leq |c'| + |b|$ holds. Finally, the last inequality follows by monotonicity of q_f.

This completes the proof. □

Theorem C.2. *Let* $\mathrm{f}(\mathbf{x};\mathbf{y})$ *be in* PS*. Then* $\mathrm{f}(\mathbf{x},\mathbf{y})$ *is in* FPSPACE*.*

Proof. This is an immediate consequence of Theorem 4.13. Alternatively the claim can be shown by induction on the definition of PS. It can be easily seen that the initial functions of PS are all polyspace computable. Also, if f is an instance of safe composition then by forgetting the distinction between safe and normal arguments f is an instance of composition and the claim follows by induction hypothesis.

Finally, assume $\mathrm{f}(z,b,\mathbf{x};\mathbf{y},a)$ is defined by generalized safe recursion on h. We define the function $\hat{\mathrm{f}}(z,b,\mathbf{x};\mathbf{y},a)$ by safe primitive recursion as follows.

$$\begin{aligned} g(b,\mathbf{x};\mathbf{y},a) &= a \\ \hat{\mathrm{f}}(\epsilon,b,\mathbf{x};\mathbf{y},a) &= g(b,\mathbf{x};\mathbf{y},a) \\ \hat{\mathrm{f}}(z',b,\mathbf{x};\mathbf{y},a) &= \hat{\mathrm{h}}(z,b,\mathbf{x};\mathbf{y},\hat{\mathrm{f}}(z,b,\mathbf{x};\mathbf{y},a)) \\ \hat{\mathrm{h}}(z,b,\mathbf{x};\mathbf{y},a) &= h(z+b,\mathbf{x};\mathbf{y},a) \end{aligned}$$

Then using Lemma 3.3 it is an easy exercise to show that for all $a,z,b,\mathbf{m},\mathbf{n} \in \mathbb{W}$ we have $\mathrm{f}(z,b,\mathbf{m};\mathbf{n},a) = \hat{\mathrm{f}}(z,b,\mathbf{m};\mathbf{n},a)$. By Lemma A.2, PS is closed under safe primitive recursion, so $\hat{\mathrm{f}} \in$ PS. Observe that using the bounding polynomial established in Lemma C.1 an appropriate bounding function t from FPSPACE can be constructed. Thus $\hat{\mathrm{f}}$ is an instance of bounded primitive recursion with bounding function t and we conclude $\mathrm{f} \in$ FPSPACE. □

References

[1] A. Weiermann A. Beckmann. A term rewriting characterization of the polytime functions and related complexity classes. *Archive for Mathematical Logic*, 36:11–30, 1996.

[2] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

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

[4] E. A. Cichon and Andreas Weiermann. Term rewriting theory for the primitive recursive functions. *Ann. Pure Appl. Logic*, 83(3):199–223, 1997.

[5] A. Cobham. The intrinsic computational difficulty of functions. In *Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, pages 24–30, 1964.

[6] W. G. Handley and S. S. Wainer. Equational derivation vs. computation. *Ann. Pure Appl. Logic*, 70(1):17–49, 1994.

[7] Isabel Oitavem. New recursive characterizations of the elementary functions and the functions computable in polynomial space. *Revista Mathematica de la Universidad Complutense de Madrid*, 10:109–125, 1997.

[8] Isabel Oitavem. Implicit characterizations of pspace. In Reinhard Kahle, Peter Schroeder-Heister, and Robert F. Stärk, editors, *Proof Theory in Computer Science*, volume 2183 of *Lecture Notes in Computer Science*, pages 170–190. Springer, 2001.

[9] Kenya Ueno. Recursion theoretic operators for function complexity classes. In Xiaotie Deng and Ding-Zhu Du, editors, *ISAAC*, volume 3827 of *Lecture Notes in Computer Science*, pages 748–756. Springer, 2005.

[10] Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.