Continuous time computation with restricted integration capabilities

Manuel Lameiras Campagnolo

D.M./I.S.A., Universidade Técnica de Lisboa,
Tapada da Ajuda, 1349-017, Lisboa, Portugal;
C.L.C./D.M./I.S.T., Universidade Técnica de Lisboa,
Av. Rovisco Pais, 1049-001, Lisboa, Portugal
mlc@math.isa.utl.pt

Abstract. Recursion theory on the reals, the analog counterpart of recursive function theory, is an approach to continuous-time computation inspired by the models of Classical Physics. In recursion theory on the reals, the discrete operations of standard recursion theory are replaced by operations on continuous functions such as composition and various forms of differential equations like indefinite integrals, linear differential equations and more general Cauchy problems. We define classes of real recursive functions in a manner similar to the standard recursion theory and we study their complexity. We prove both upper and lower bounds for several classes of real recursive functions, which lie inside the elementary functions, and can be characterized in terms of space complexity. In particular, we show that hierarchies of real recursive classes closed under restricted integration operations are related to the exponential space hierarchy. The results in this paper, combined with earlier results, suggest that there is a close connection between analog complexity classes and subrecursive classes, at least in the region between $\mathcal{F}$LINSPACE and the primitive recursive functions.

Key words: Continuous-time computation, recursion theory, computational complexity, exponential space hierarchy, differential equations, numerical integration.

1 Introduction

In recent years, there has been a renewed interest in analog computation, where the internal states of a computer are continuous rather than discrete. In particular there is an interest in continuous time computation, where the states of the computer evolve continuously. The main motivation for this is the belief that it is possible to use the tools of computation theory to better classify the variety of continuous dynamical systems we see in the world, or at least in its classical idealization. Previous efforts have had promising results but much remains to be done, especially with respect to the complexity of analog computation [17].

Many proposed models of analog computation can be viewed as a general framework where digital computation is simply embedded [3, 5, 15, 4, 20, 6]. This

suggests the two following questions. First, one can ask if is it possible to compute beyond the Turing model, i.e., are there functions non computable by the Turing machine which can be computed without violating the laws of Physics. Secondly, we can ask if the Turing machine is an adequate model to describe the computations that occur in our physical world. In this paper we prove some results that may be of interest with regard to the latter issue.

Functions like sin, cos and log are extremely frequent in our models of physical phenomena. Those functions are solutions of very simple differential equations. Although their computation by digital means require fairly complex algorithms, they can be computed efficiently as most common continuous mathematical functions. In this paper we claim that there is in fact a close connection between classical recursion and real recursion when we restrict the operators in both theories in a similar way, at least for the region of the complexity hierarchy we consider.

In classical recursion theory, there is a significant interest in various forms of so-called bounded recursion [8]. We will see that we can build a very natural hierarchy of continuous time computable classes of functions, where each level is closed under a bounded integration operator, and which is tightly related to the exponential space hierarchy.

In [7] we explored the computational power of a real recursive class where only linear differential equations could be solved. We showed that if such a model was allowed to sense inequalities in a differentiable way, then it could compute exactly the elementary functions, i.e., the set of functions computable in time or space bounded by a tower of exponentials of arbitrary height. It is surprising that such systems, as opposed to the highly non-linear ones that are commonly used to model physical phenomena, have so much computational power. That result brings a new insight on how digital computation implicitly reduces the complexity of our models of the physical world.

Here, we also explore an even more restricted model which, from a pool of basic functions, can only calculate integrals of functions already defined and solve up to a certain number of linear differential equations. We will show that this is still enough to reach all the levels of the exponential space hierarchy.

We first define a restricted form of integration, we call bounded integration, where the solution of the Cauchy problem is supposed to have an *a priori* bound, and we describe an analog class $\mathcal{B}_0$. We show that $\mathcal{B}_0$ contains $\mathcal{F}$LINSPACE, i.e., the class of functions computable by a deterministic Turing machine in linear space. Then, we build upon it a whole hierarchy of classes $\mathcal{B}_n$ which contains, level by level, the exponential space hierarchy, EXPSPACEF^n. We also show that functions in $\mathcal{B}_n$ are computable (in the sense of Grzegorczyk and Lacombe) in space bounded by a tower of exponentials of height n.

Next, we eliminate the explicit bounds in the definition of $\mathcal{B}_n$ and describe a hierarchy of real recursive classes $\mathcal{S}_n$ which contains $\mathcal{B}_0$ as its set of basic functions, where integration of functions already defined can be used freely and the nth level can be reached if we allow the system to solve up to n linear differential equations. We show that $\mathcal{S}_n \subset \mathcal{B}_n$ and therefore that $\mathcal{S}_n$ has the same com-

putable upper bounds as $\mathcal{B}_n$. We also show that all functions in EXPSPACEF^n have extensions to the reals in $\mathcal{S}_{n+1}$.

We end with some remarks and open questions.

2 Real recursive functions, $\mathcal{B}_0$, and upper and lower bounds

A function algebra

$$[B_1, B_2, ...; O_1, O_2, ...],$$

is the smallest set containing basic functions $\{B_1, B_2, ...\}$ and closed under certain operations $\{O_1, O_2, ...\}$, which take one or more functions in the class and create new ones. Although function algebras have been defined in the context of recursion theory on the integers, they are equally suitable to define classes of real valued recursive functions. Moore [15] proposed the first theory of recursive functions on the reals in analogy with classical recursion theory, where primitive recursion is replaced by an integration operator, and minimalization on the integers is extended to the reals. We first consider the following subclass of the real recursive functions, which is closed under composition and under a restricted form of integration named bounded integration.

Definition 1 *(The class $\mathcal{B}_0$).*

$$\mathcal{B}_0 = [0, 1, -1, +, \times, U, \theta_k; \mathrm{COMP}, \mathrm{BI}]$$

where

- $0, 1, -1$ *are constant unary functions,* $+$ *and* $\times$ *are the usual binary sum and product,* U *denotes the set of projections* $\{U_i^n(\boldsymbol{x}) = x_i, n \in \mathbb{N}, 0 < i \leq n\}$*, and* θ_k*, for* $k > 2$*,*[1] *is defined on* $\mathbb{R}$ *by* $\theta_k(x) = x^k$*, when* $x \geq 0$ *and* $\theta_k(x) = 0$*, when* $x < 0$*.*
- COMP *denotes composition, i.e. given* $f_1, \ldots, f_p$ *of arity* n *and* g *of arity* p*, then* $h(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x}))$ *is defined,*
- BI *denotes bounded integration,i.e., given functions* $f_1, \ldots, f_m$ *of arity* n*,* $g_1, \ldots, g_m$ *of arity* $n + 1 + m$*, and* b *of arity* $n + 1$*, if* $(h_1, \ldots, h_m)$ *is the unique function that satisfies the equations*

$$\boldsymbol{h}(\boldsymbol{x}, y) = \boldsymbol{f}(\boldsymbol{x}), \quad \partial_y \boldsymbol{h}(\boldsymbol{x}, y) = \boldsymbol{g}(\boldsymbol{x}, y, \boldsymbol{h}(\boldsymbol{x}, y)), \quad ||\boldsymbol{h}(\boldsymbol{x}, y)|| \leq b(\boldsymbol{x}, y)$$

on $\mathbb{R}^{n+1}$*, then* $h = h_1$ *of arity* $n + 1$*, is defined.*

In recursion theory on the reals arbitrary constants are not allowed. This is essential to establish complexity upper bounds. In fact, if we allow arbitrary reals, then the complexity of integration can be arbitrarily high [14]. We say that a constant $a \in \mathcal{B}_0$ if there is a $f \in \mathcal{B}_0$ s.t. $f(x) = a$, for all $x \in \mathbb{R}$.

[1] With this condition, all functions in $\mathcal{B}_0$ are twice continuously differentiable.

Notice that the basic functions of $\mathcal{B}_0$ are bounded by polynomials and COMP and BI preserve this property. Therefore, all functions in $\mathcal{B}_0$ are polynomially bounded and are total, i.e., they are defined on $\mathbb{R}^n$. Moreover, they are of class $C^{k-1}(\mathbb{R}^n)$.

Many usual functions like sin, cos, polynomials with coefficients in $\mathcal{B}_0$, and even a function that matches log on $[1, +\infty)$ belong to $\mathcal{B}_0$. An example of a function that isn't in this class is $\exp : \mathbb{R} \to \mathbb{R}^+$ since it is not bounded by a polynomial. Given a class of continuous functions like $\mathcal{B}_0$, it is natural to ask what is its computational complexity. Namely, we may want to know which discrete functions can be computed in the class. We may also want to determine the computational resources necessary to approximate (in the sense of computational analysis) all functions in the class.

Since we are only interested in elementary functions, we can characterize them in terms of standard space or time complexity, and consider the Turing machine as the underlying computational model. This approach differs from others, namely BSS-machines [3] or information-based complexity [21], since it focus on *effective* computability and complexity.

To compare the computational complexity of real recursive classes and standard recursive classes we have to set some conventions. On one hand, we follow a straightforward approach to associate a class of integer functions to a real recursive class. We simply consider the *discretization* of a real recursive class, i.e., the subset of functions with integer values for integer arguments. More precisely,

Definition 2 *Given a real recursive class $\mathcal{C}$, $\mathcal{F}_{\mathbb{N}}(\mathcal{C}) = \{f : \mathbb{N}^n \to \mathbb{N}$ s.t. f has an extension to the reals in $\mathcal{C}\}$. We say that $\mathcal{C}$ is closed under a certain operation O in a* weak sense *if $[f, g, \ldots \in \mathcal{F}_{\mathbb{N}}(\mathcal{C}) \Rightarrow \mathrm{O}(f, g, \ldots) \in \mathcal{F}_{\mathbb{N}}(\mathcal{C})]$.*

If $\mathcal{F}_{\mathbb{N}}(\mathcal{C})$ contains a certain complexity class $\mathcal{C}'$, then we can consider $\mathcal{C}'$ as a lower bound for $\mathcal{C}$.

On the other hand, we consider the computational complexity of real functions. We use the notion of [13], whose underlying computational model is the function-oracle Turing machine. Intuitively, the time (resp. space) complexity of f is the number of moves (resp. the amount of tape) required by a function-oracle Turing machine to approximate the value of $f(x)$ within an error bound 2^{-n}, as a function of the argument x and the precision n.

Let's briefly recall what a function-oracle Turing machine is (we give an informal description: details can be found in [12, 13]). We write $\phi \rightsquigarrow x$ if $\phi = \{\phi(n)\}$ is a computable sequence s.t. for all $n \in \mathbb{N}$, $\|\phi(n) - x\| < 2^{-n}$. For any x in the domain of f, ϕ s.t. $\phi \rightsquigarrow x$ is the oracle. The machine is a Turing machine equipped with an additional query tape, and two additional states. When the machine enters in the query state, it replaces the current string s in the query tape by the string $\phi(s)$, moves the head to the first cell of the query tape, and switches to the answer state. This is done in one step of the computation. The generalization to functions of several variables is straightforward. The input of the function-oracle TM M is a string 0^n. Let $M^{\phi}(n)$ denote its output. Then,

Definition 3 *(Computability and complexity of real computable functions) M computes f if for any x in the domain of f and for any $\phi \rightsquigarrow x$, $\{M^\phi(n)\} \rightsquigarrow f(x)$, i.e., for all n, $\|M^\phi(n) - f(x)\| < 2^{-n}$. We say that the time (resp. space) complexity of f on its domain is a certain function b if there is a function-oracle TM M which, for all n, outputs $M^\phi(n)$ in a number of steps (resp. amount of tape) bounded by $b(x, n)$.*

Intuitively, the computation of f by a function-oracle TM M with input 0^n and oracle ϕ is done in two parts (cf. [13]): (1) M computes, from the output precision 2^{-n}, the required input precision 2^{-m}; (2) M queries the oracle to get $\phi(m)$, with $\|\phi(m) - x\| < 2^{-m}$ and outputs $M(\phi(m))$ such that $\|M(\phi(m)) - f(x)\| \leq 2^{-n}$. Here, $M(\phi(m))$ denotes the output of M that runs on $\phi(m)$ *without* entering in the query state. Since we will need to evaluate the required input precision, we define the following notation.

Definition 4 *(Required input precision) Let $\phi \rightsquigarrow x$. If there is a function-oracle TM M that computes f and a function l s.t. for all x in the domain of f and all n,*

$$m \geq l(x, n) \Rightarrow \|M(\phi(m)) - f(x)\| < 2^{-n},$$

then $l(x, n)$ is called the required input precision to compute $f(x)$ with precision 2^{-n}.

For a certain function $f : \mathbb{R}^n \to \mathbb{R}$, s_f and l_f will denote, respectively, the space complexity and the required input precision for f.

Given the conventions above, we will now show lower and upper bounds for $\mathcal{B}_0$. We will first prove that all functions in $\mathcal{F}$LINSPACE have extensions to the reals in $\mathcal{B}_0$, i.e., $\mathcal{F}\text{LINSPACE} \subset \mathcal{F}_{\mathbb{N}}(\mathcal{B}_0)$. Towards that end, we will use the following function algebra for $\mathcal{F}$LINSPACE given by [18].

Lemma 5 $\mathcal{F}\text{LINSPACE} = [0, S, +, \times, U; \text{COMP}, \text{BREC}]$*, where S is the successor function,* COMP *denotes composition and* BREC *denotes bounded recursion, i.e., given f of arity n, g of arity $n + 2$ and b of arity $n + 1$, define h as the unique function of arity $n + 1$ such that, for all $\boldsymbol{x}$ and y,*

$$h(\boldsymbol{x}, 0) = f(\boldsymbol{x}), \quad h(\boldsymbol{x}, y + 1) = g(\boldsymbol{x}, y, h(\boldsymbol{x}, y)), \quad \|h(\boldsymbol{x}, y)\| \leq b(\boldsymbol{x}, y). \tag{1}$$

Lemma 6 *$\mathcal{B}_0$ is closed under bounded recursion in a weak sense.*

Proof. We will abuse notation by identifying f, g, h and b in Equation (1) with their extensions to the reals. We show that if h is defined from $f, g, b \in \mathcal{F}_{\mathbb{N}}(\mathcal{B}_0)$, then we can define in $\mathcal{B}_0$ a real recursive extension of h. Our construction is similar to [5]. We claim that the function h is the first component of the solution of the following differential equation that can be defined in $\mathcal{B}_0$.

$$\begin{aligned} \partial_t y_1 &= C_k \left(g(\boldsymbol{x}, s(t), r(y_2)) - y_1\right)^3 c_1(t) \\ \partial_t y_2 &= C_k \left(r(y_1) - y_2\right)^3 c_2(t) \end{aligned} \tag{2}$$

with $y_1(\boldsymbol{x}, 0) = y_2(\boldsymbol{x}, 0) = f(\boldsymbol{x})$. The "control" function $c_1(t)$ is $\theta_k(\sin 2\pi t)$ and $c_2(t)$ is $\theta_k(-\sin 2\pi t)$. Therefore, $c_2(t) = 0$ for $t \in [j, j + 1/2]$, and $c_1(t) = 0$ for $t \in [j + 1/2, j + 1]$, for all j integer. The functions s is defined by integration from c_2, simply by $s(0) = 0$ and $s' = d_k c_2$, where d_k is a constant in $\mathcal{B}_0$ such that $s(t) = j$, whenever $t \in [j, j + 1/2]$, for all j integer. Finally, r is defined as $r(t) = s(t + 1/4)$, so $r(t) = j$, whenever $t \in [j - 1/4, j + 1/4]$, for all j integer. Then, on $[0, 1/2]$, y_2 is hold constant and the behavior of y_1 is given by

$$\partial_t y_1 = C_k \left(g(\boldsymbol{x}, 0, f(\boldsymbol{x})) - y_1\right)^3 \sin^k 2\pi t, \tag{3}$$

with $y_1(\boldsymbol{x}, 0) = f(\boldsymbol{x})$. From Equation (1), $g(\boldsymbol{x}, 0, f(\boldsymbol{x})) = h(\boldsymbol{x}, 1)$, which we will denote by h_1. Then, the solution of Equation (3) satisfies

$$(h_1 - y_1(t))^2 = \left(C_k \textstyle\int_0^t \sin^k 2\pi t' dt' + C\right)^{-1},$$

where $C \geq 0$. If we choose C_k large enough, then $\|h_1 - y_1(\boldsymbol{x}, \frac{1}{2})\| \leq 1/4$ and therefore $r(y_1(\boldsymbol{x}, 1/2)) = h_1$. The same argument for y_2 shows that $r(y_2(\boldsymbol{x}, 1)) = h_1$. Repeating this for subsequent intervals $[j, j+1]$, for all j integer, we conclude that $r(y_1(\boldsymbol{x}, j)) = r(y_2(\boldsymbol{x}, j)) = h(\boldsymbol{x}, j)$, for all $j \in \mathbb{N}$.

To show that the solution of Equation (2) is bounded by a function in $\mathcal{B}_0$ we just note that y_1 and y_2 are monotonic in each interval $[j, j + 1/2]$ and $[j + 1/2, j + 1]$ for j integer. Since they are arbitrarily close to $h(\boldsymbol{x}, j)$ for j multiples of $1/2$ and h is provably bounded by a polynomial then both y_1 and y_2 are also bounded by a polynomial. □

Proposition 7 $\mathcal{F}$LINSPACE $\subset \mathcal{F}_{\mathbb{N}}(\mathcal{B}_0)$.

Proof. We use the recursive characterization of $\mathcal{F}$LINSPACE given by Lemma 5. $\mathcal{B}_0$ contains 0 and 1 and is closed under addition. Therefore, it also contains the successor function $S(x) = x+1$. It is also closed under multiplication. Moreover, $\mathcal{B}_0$ contains by definition the projection functions and is closed under composition. Finally, $\mathcal{B}_0$ is closed under bounded recursion in the sense of Lemma 6. Therefore, all functions in $\mathcal{F}$LINSPACE have real extensions in $\mathcal{B}_0$. □

Next we will set a bound on the space complexity of any function in $\mathcal{B}_0$. First we need to set bounds on the derivatives of functions in $\mathcal{B}_0$.

Lemma 8 *All functions in $\mathcal{B}_0$ and its first and second derivatives are bounded by polynomials.*

Proof. We already know that functions in $\mathcal{B}_0$ have polynomial bounds. We will show next that this is also the case for the derivatives. Basic functions in $\mathcal{B}_0$ obviously have first and second derivatives with polynomial bounds.

By induction hypothesis, $f_1, \ldots, f_p, g$ and their derivatives are bounded by polynomials. The first and second derivatives of the composition $g(f_1, \ldots, f_p)$ are given by the chain rule. Therefore they are also bounded by polynomials.

If h is defined by bounded integration, then its first derivative is

$$\partial_y h(\boldsymbol{x}, y) = g(\boldsymbol{x}, y, h(\boldsymbol{x}, y)),$$

which is clearly bounded by a polynomial, or is given by the solution of

$$\partial_{x_i} h(\boldsymbol{x}, 0) = \partial_{x_i} f(\boldsymbol{x}), \quad \partial_{x_i y} h(\boldsymbol{x}, y) = \partial_{x_i} g(\boldsymbol{x}, y, h(\boldsymbol{x}, y)). \tag{4}$$

By the induction hypothesis, $\partial_{x_i} f(\boldsymbol{x})$ is bounded by $p_1(\boldsymbol{x})$ and $\partial_{x_i} g(\boldsymbol{x}, y, z)$ is bounded by $p_2(\boldsymbol{x}, y, z)$, where the p_i's denote polynomials. Since h, which is in $\mathcal{B}_0$, is bounded by a polynomial, then $\partial_{x_i} g(\boldsymbol{x}, y, h(\boldsymbol{x}, y))$, as a function of $\boldsymbol{x}$ and y, is also bounded by a polynomial, say, $p_3(\boldsymbol{x}, y)$. Therefore, by differential inequality (cf. [10, p. 29]),

$$\partial_{x_i} h(\boldsymbol{x}, y) \leq p_1(\boldsymbol{x}) + \int_0^y p_3(\boldsymbol{x}, t) dt,$$

which is itself polynomial in $\boldsymbol{x}$ and y. We can use the same argument to prove that the second derivatives of h are also bounded by polynomials. □

Recall that we want to find a bound on the space complexity of $\mathcal{B}_0$. We will proceed by setting a bound for the basic functions and showing that both composition and bounded integration preserve it. For simplicity, we will consider scalar functions defined on $\mathbb{R}_0^+$. The lemma below follows directly from Definition 4 of required precision input.

Lemma 9 *(Numerical composition) Let f and g be computable by $\mathbf{f}$ and $\mathbf{g}$ and h be defined by $h(x) = g(f(x))$. Let ϕ be an oracle for x and b a bound on f. Then h is computable with precision 2^{-m} by the following algorithm, with input m.*

$\tilde{x} \leftarrow \phi(1) + 1/2$ *approximate value for x, such that $\tilde{x} \geq x$*
$N \leftarrow l_g(b(\tilde{x}), m)$ *required precision on $f(x)$ to compute $g(f(x))$*
$M \leftarrow l_f(\tilde{x}, N)$ *required precision on x to compute $f(x)$*
$h \leftarrow \mathbf{g}(\mathbf{f}(\phi(M))$ *output value*

As it is widely known, given a bound on the error, the numerical integration of ordinary differential equations by standard numerical analysis techniques (*e.g.* Euler's method) requires a number of steps exponential in the argument and in the second derivative. However, the amount of tape grows at most polynomially when the ordinary differential equation is defined in $\mathcal{B}_0$. Below, we give an algorithm to compute the solution of an O.D.E..

Lemma 10 *(Numerical integration) Let f and g be computable by $\mathbf{f}$ and $\mathbf{g}$ and $l = max\{l_f, l_g\}$. Let h be the unique solution of the initial problem $h(x, 0) = f(x)$ and $\partial_t h(x, t) = g(x, t, h(x, t))$ on $[0, y]$. Let $\phi_x \rightsquigarrow x$ and $\phi_y \rightsquigarrow y$. If b_L is a bound on the Lipschitz constant for g w.r.t. its last argument and b_1 and b_2 are, respectively, bounds on $\partial_y h$ and $\partial_{y^2} h$, then h is computable with precision 2^{-m} by the following algorithm, with input m.*[2]

[2] Without loss of generality we consider that b_L, b_1, b_2 are increasing functions and are greater than 1.

$\tilde{x} \leftarrow \phi_x(1) + 1/2$ *approximate value for x, s.t. $\tilde{x} \geq x$*
$\tilde{y} \leftarrow \phi_y(1) + 1/2$ *approximate value for y, s.t. $\tilde{y} \geq y$*
$\eta \leftarrow m + 3 + 2\tilde{y}\, b_L(\tilde{y})$
$M \leftarrow l(\tilde{x}, \eta)$ *required precision on x*
$h \leftarrow \mathbf{f}(\phi_x(M))$ *initial condition*
$N \leftarrow m + 3 + \log b_1(\tilde{y})$ *required precision on y*
$n \leftarrow \tilde{y}\, b_2(\tilde{y})\, e^{b_L(\tilde{y})\,\tilde{y}}\, 2^{m+3}$ *number of steps*
$\delta \leftarrow \phi_y(N)/n$ *step size*
$t \leftarrow 0$
for $i \leftarrow 1, \ldots, n$
    $h \leftarrow h + \delta\, \mathbf{g}(\phi_x(M), t, h)$
    $t \leftarrow t + \delta$

Proof. Notice that $\tilde{y}, \tilde{x}$ are used just to set the required input precision and the number of necessary steps. The numerical integration is carried on the interval $[0, \phi_y(N)]$, where $\|\phi_y(N) - y\| < 2^{-N}$. For convenience, we will consider in what follows $y = \tilde{y}$ or $y = \phi_y(N)$ when appropriate.

Suppose that, in the algorithm above, the arithmetical operations are performed exactly, and the non integers δ and t are represented exactly. This is guaranteed if we choose as the step size the dyadic rational just below δ. Then, the total error of the approximation is bounded by the sum of the error caused by the approximation of y by $\phi_y(N)$, the discretization errors of the numerical integration, the approximation errors both on the initial condition $f(x)$ and on the values of the $g(x, t, h)$ in each step.

By the mean value theorem, we know that $\|h(\phi_y(N)) - h(y)\| < b_1(\|\phi_y(N) - y\|)$. Since we set $N = m + 3 + \log b_1(y)$, then that error is at most 2^{-m-3}.

Let's denote the computed value of h in each iteration of Euler's method by h_i, and the value of t by t_i. The error in each step is $e_i = h(x, t_i) - h_i$. We follow [11]. Using Taylor's formula on $[t_i, t_{i+1}]$ for $h(x, t_{i+1})$,

$$e_{i+1} = e_i + \delta[g(x, t_i, h_i) - g(x, t_i, h(x, t_i)) + \mathbf{g}(\phi_x(M), t_i, h_i) - g(x, t_i, h_i)] + \tfrac{1}{2}\delta^2 \partial_{y^2} h(\xi),$$

where $\xi \in (t_i, t_{i+1})$. Using the fact that $\|g(x, t_i, h_i) - g(x, t_i, h(x, t_i))\| \leq b_L(y)\|h_i - h(x, t_i)\|$ and $\partial_{y^2} h(\xi) \leq b_2(y)$, we obtain

$$\|e_{i+1}\| \leq (1 + \delta\, b_L(y))\|e_i\| + \tfrac{1}{2}\delta^2\, b_2(y) + \delta\|\mathbf{g}(\phi_x(M), t_i, h_i) - g(x, t_i, h_i)\|.$$

The accumulated error is then bounded by (cf. [11, (1.13)])

$$\|e_n\| \leq e^{n\delta b_L(y)} \left(\|e_0\| + \delta\, b_2(y) + \|\mathbf{g}(\phi_x(M), t, h_i) - g(x, t_i, h_i)\|\right).$$

where e_0 is the error on the initial condition. Since $f(x)$ is computed with precision $\eta = m + 3 + 2y\, b_L(y)$, then $\|e_0\| < 2^{-(m+3+2y\, b_L(y))}$. Moreover, $n = y\, b_2(y)\, e^{b_L(y)\, y}\, 2^{m+3}$ and $\delta \approx y/n$. Furthermore, the error on $g(x, t_i, h_i)$ depends only on the precision on its first argument and is also bounded by $2^{-\eta}$. Therefore, the three terms on the right hand side of the equation above have the following bounds

$$\frac{e^{y\, b_L(y)}}{2^{m+3+2y\, b_L(y)}}, \frac{y b_2(y) e^{y\, b_L(y)}}{y\, b_2(y)\, e^{y\, b_L(y)}\, 2^{m+3}}, \frac{e^{y\, b_L(y)}}{2^{m+3+2y\, b_L(y)}} \leq 2^{-m-3}.$$

Then, the total error on the approximation of $h(x,y)$ is less than 2^{-m}, as claimed. □

Finally, we set space complexity bounds for $\mathcal{B}_0$. The complexity is linear in terms of the precision, and polynomial in terms of the argument of the function.

Proposition 11 *If $f \in \mathcal{B}_0$, then its space complexity is $s_f(x,n) \in x^{O(1)} + O(n)$.*

Proof. We will also show that if $f \in \mathcal{B}_0$, then $l_f(x,n) \in x^{O(1)} + O(n)$. We proceed by induction, starting with the basic functions of $\mathcal{B}_0$, which are $0, 1, -1, +, \times, \theta_k, U$. We will prove our claim for $\times$ and θ_k. The other cases are left to the reader.

($\times$) The binary product can be computed with the following sequence of operations and input n:

$$\begin{array}{l} \tilde{x} \leftarrow \phi_x(1) + 1/2; \\ \tilde{y} \leftarrow \phi_y(1) + 1/2; \\ N \leftarrow n + 1 + \log(\tilde{x} + \tilde{y}); \\ \phi_x(N)\phi_y(N). \end{array}$$

Clearly, $l_\times(\|(x,y)\|, n) \in \|(x,y)\|^{O(1)} + O(n)$. Furthermore, the error satisfies

$$\|\phi_x(N)\phi_y(N) - xy\| < 2^{-N}(\tilde{x}+\tilde{y}) + 2^{-2N} = 2^{-[n+1+\log(\tilde{x}+\tilde{y})]}(\tilde{x}+\tilde{y}) + 2^{-2[n+1+\log(\tilde{x}+\tilde{y})]}.$$

A little algebra shows that this is less than 2^{-n} as claimed.[3]

(θ_k) This basic function is computed, with input n, by:

$$\begin{array}{l} \tilde{x} \leftarrow \phi_x(1) + 1/2; \\ N \leftarrow n + 1 + k \log \tilde{x}; \\ \theta_0 \leftarrow \text{if}(\phi_x(N) > 0, \phi_x(N), 0); \\ \theta_0 \times \ldots \times \theta_0, \quad k \text{ times}. \end{array}$$

Clearly, $l_{\theta_k}(x,n) \in x^{O(1)} + O(n)$. The error is bounded by $\theta_0^{k-1}(\tilde{x})2^{-N} + \ldots + 2^{-kN}$. Again, assuming that $\tilde{x} > 1$, we can check that this is bounded by 2^{-n}.

Next, we show that the operation COMP and BI preserve the bounds on the required input precision and on the space complexity. Recall that if $f \in \mathcal{B}_0$, then there is polynomial p, such that $f < p$. Therefore, the size of $f(x)$ is linear in x.

(Composition) By induction hypothesis, suppose that $l_f(x,n), l_g(x,n) \in x^{O(1)} + O(n)$. We follow the algorithm in Lemma 9. The required precision on the input for $g \circ f$ is $l_{g\circ f} = M = l_f(\tilde{x}, l_g(p(\tilde{x}), m))$, where $f < p$, and still belongs to $x^{O(1)} + O(m)$ as claimed. Now, let's verify that the space complexity of h is in $x^{O(1)} + O(m)$. First, $f(x)$ has to be computed with precision 2^{-N}. This can be done, by induction hypothesis, in space $x^{O(1)} + O(N)$. Since $N \in x^{O(1)} + O(m)$, then $\mathbf{f}(\phi(M))$ can be computed in space $x^{O(1)} + O(m)$. Finally, $g(f(x))$ has to be computed with precision 2^{-m}. This can be done, by induction hypothesis, in

[3] We assume that $\tilde{x}, \tilde{y} > 1$.

space $f(x)^{O(1)} + O(m)$. Since $f < p$, then $\mathbf{g}(\mathbf{f}(\phi(M)))$ can also be computed in space $x^{O(1)} + O(m)$.

(Integration) Again, by induction hypothesis, suppose that $l = \max\{l_f(x,n), l_g(x,n)\} \in x^{O(1)} + O(n)$. We follow now the algorithm in Lemma 10. The required precision on the input is dominated by $M = l(x,\eta)$, with $\eta = m + 3 + 2y\, b_L(y)$. Since b_L is polynomial, M is polynomial in $\|(x,y)\|$ and linear in m as claimed. For the space complexity, notice that $f(x)$ has to be computed with precision $2^{-\eta}$. By induction hypothesis this can be done in space $x^{O(1)} + O(\eta)$ which is in $\|(x,y)\|^{O(1)} + O(m)$. Likewise, and since $t < y$ and $h < p$, where p is some polynomial, the space needed to compute $\mathbf{g}(\phi_x(M), t, h)$ is also in $\|(x,y,p(x,y))\|^{O(1)} + O(m) \in \|(x,y)\|^{O(1)} + O(m)$. Finally, we have to write on the tape n, δ, t, which sizes are of the order of $n = y\, b_2(y)\, e^{b_L(y)\, y}\, 2^{m+3}$. Since b_2 is a polynomial by Lemma 8, then the size of the number n is also polynomial in y and linear in m as required. □

3 Restricting composition and bounded integration

Now, we generalize the results of the previous section to the levels of the exponential space hierarchy, which ranges from the class of functions computable in linear space to the class of functions computable in elementary time or, equivalently, elementary space. This hierarchy is described in [16, p. 278], where its levels are denoted by EXPSPACEF^n. Notice that EXPSPACEF$^0 = \mathcal{F}$LINSPACE. Moreover, $\cup_n$EXPSPACEF$^n = \mathcal{E}$, the set of elementary functions.

Definition 12 *(The exponential space hierarchy) Let $f : \mathbb{N}^p \to \mathbb{N}$. $f \in$ EXPSPACEF^n iff, for all x, $f(x)$ is computable by a deterministic Turing machine in space $2^{[n]}(O(|x|))$, where $2^{[n]}$ is the iterated exponential defined by $2^{[0]}(m) = m$ and $2^{[n+1]}(m) = 2^{2^{[n]}(m)}$.*[4]

The hierarchy above cannot be closed under composition. Otherwise, the it would collapse for $n \geq 1$ since composing 2^x, which belongs to EXPSPACEF^1, with itself, would take us to any level of the hierarchy.

Thus, we will restrict composition and bounded integration similarly to [18]. Let the *degree* of the function f with n arguments be a vector of length n, where each component is either 0 or undefined.

We will see that, if $\deg_i f = 0$, where $\deg_i f$ denotes the ith component of the degree of f, then f grows at most polynomially w.r.t. its ith argument x_i. With the restricted operations we define, only composition with functions of degree zero is allowed, and a restricted form of bounded integration where the integrand function has to be of degree 0 w.r.t. to the solution is used.

We will sometimes use the more compact notation $\deg f(\boldsymbol{x}, \boldsymbol{y}) = (\deg_{\boldsymbol{x}} f, \deg_{\boldsymbol{y}} f)$ and $\deg_{\text{last}} f = 0$ when the last component of the degree is 0.

Next we describe the classes $\mathcal{B}_n$, which basic functions are the functions in $\emptyset$ and the iterated exponential $2^{[n]}$, and the composition and integration operators are restricted according to the remarks above.

[4] As usual, $|x|$ denotes the size of the number x.

Definition 13 *(The hierarchy $\mathcal{B}_n$) For all $n \geq 1$,*

$$\mathcal{B}_n = [\mathcal{B}_0, 2^{[n]}; \mathrm{RCOMP}, \mathrm{RBI}]$$

where

- *all functions in $\mathcal{B}_0$ have degree 0,*
- RCOMP *denotes restricted composition, i.e. given $f_1, \ldots, f_p$ of arity m and g of arity p, with for all $j = 1, \ldots, p$ and all $i = 1, \ldots, m$, $\deg g_j = 0$ or $\deg_i f_j = 0$, then define*

$$h(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x})),$$

  and $\deg_i h = 0$ iff for all $j = 1, \ldots, p$, $\deg_j g = \deg_i f_j = 0$,
- RBI *denotes restricted bounded integration, i.e., given functions $f_1, \ldots, f_m$ of arity n, $g_1, \ldots, g_m$ of arity $n + 1 + m$, with $\deg_{\boldsymbol{z}} g_i(\boldsymbol{x}, y, \boldsymbol{z}) = 0$, and b of arity $n+1$, if $(h_1, \ldots, h_m)$ is the unique function that satisfies the equations*

$$\boldsymbol{h}(\boldsymbol{x}, 0) = \boldsymbol{f}(\boldsymbol{x}), \quad \partial_y \boldsymbol{h}(\boldsymbol{x}, y) = \boldsymbol{g}(\boldsymbol{x}, y, \boldsymbol{h}(\boldsymbol{x}, y)), \quad ||\boldsymbol{h}(\boldsymbol{x}, y)|| \leq b(\boldsymbol{x}, y)$$

  on $\mathbb{R}^{n+1}$, then $h = h_1$ of arity $n + 1$, is defined, and for $i = 1, \ldots, n + 1$, $\deg_i h = 0$ iff $\deg_i f = \deg_i g = \deg_i b = 0$ (for convenience, set $\deg_{n+1} f = 0$).

Notice that the definition of $\mathcal{B}_n$ is very similar to $\mathcal{B}_0$. As a matter of fact, if a function in $\mathcal{B}_n$ is defined *without* the basic function $2^{[n]}$, then it belongs to $\mathcal{B}_0$. Moreover,

Lemma 14 *If $f \in \mathcal{B}_n$ and $\deg_x f(x) = 0$ then $f \in \mathcal{B}_0$ and, therefore, $l_f, s_f \in x^{O(1)} + O(n)$.*

Proof. Let's suppose that $f \notin \mathcal{B}_0$. So, $2^{[n]}$ has to be in the recursive description of f. Then, $\deg f \neq 0$, which leads to a contradiction. □

In this section we generalize Propositions 7 and 11 for $\mathcal{B}_n$. First we show that $\mathrm{EXPSPACEF}^n \subset \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$, for $n \geq 1$. The construction we give relies on the simulation of deterministic Turing machines. Specifically, we use the arithmetization within $\mathcal{F}$LINSPACE of TMs that compute numerical functions, described by Ritchie [18]. Consider the following simulation functions:

- $\mathrm{INIT}_z(x_1, \ldots, x_n)$ is the (Gödel) number of the instantaneous description (i.d. for short) of a TM z in its initial state which is scanning in standard position the input $(x_1, \ldots, x_n)$ on the tape,
- $\mathrm{NEXT}_z(x)$ is the number of the i.d. which is the immediate result of i.d. number x on TM number z,
- $\mathrm{OUTPUT}_z(x)$ is the number represented by the tape of a TM z in a final state with i.d. number x,
- $\mathrm{ACCEPT}_z(x)$ is true iff x is the number of an i.d. of a final state in a TM z.

Lemma 15 *(Ritchie's arithmetization of TMs) If f is elementary, then there exist functions* INIT_z, NEXT_z, OUTPUT_z *and* ACCEPT_z *in* $\mathcal{F}$LINSPACE *and an elementary function b_z s.t.*

$$f(\boldsymbol{x}) = \mathrm{OUTPUT}_z\big(H_z(\boldsymbol{x}, \mu_{y \leq b_z(\boldsymbol{x})}\mathrm{ACCEPT}_z(H_z(\boldsymbol{x}, y)))\big), \tag{5}$$

where $H_z(\boldsymbol{x}, y)$ is the number of the i.d. of the y-th step of the computation of the TM z on input $\boldsymbol{x}$. Furthermore, H_z can be defined by bounded recursion as

$$H_z(\boldsymbol{x}, 0) = \mathrm{INIT}_z(\boldsymbol{x}), H_z(\boldsymbol{x}, y+1) = \mathrm{NEXT}_z(H_z(\boldsymbol{x}, y)), H_z(\boldsymbol{x}, y) \leq b_z(\boldsymbol{x}). \tag{6}$$

Moreover, it suffices that $b_z(\boldsymbol{x})$ be a bound on the number of different i.d.s that can occur on input $\boldsymbol{x}$ for the TM z.

If $f \in \mathrm{EXPSPACEF}^n$, then there is, by definition, a k such that f is computable in space bounded by $s(\boldsymbol{x}) = 2^{[n]}(k|\boldsymbol{x}|)$ for all $\boldsymbol{x}$. Suppose that f is computed by a TM with S internal states and an alphabet with m symbols. Then, the number of possible i.d.'s is bounded by

$$b_z(\boldsymbol{x}) = m^{s(\boldsymbol{x})}\ S\ |\boldsymbol{x}| \leq 2^{[n+1]}(k'|\boldsymbol{x}|) \leq 2^{[n]}(p(\boldsymbol{x})),$$

where k' is a certain constant and p is some polynomial. Notice that $b_z(\boldsymbol{x}) \in \mathrm{EXPSPACEF}^n$.

In the following lemma, we abuse notation by writing $g \in \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$ and $\deg_{\mathrm{last}} g = 0$. This means that g has an extension to the reals in $\mathcal{B}_n$ which has degree 0 w.r.t. its last argument.

Lemma 16 *If $f, g, b \in \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$ and $\deg_{\mathrm{last}} g = 0$, then there is a $h \in \mathcal{B}_n$ s.t. $h(\boldsymbol{x}, 0) = f(\boldsymbol{x})$, $h(\boldsymbol{x}, y+1) = g(\boldsymbol{x}, y, h(\boldsymbol{x}, y))$ and $\|h(\boldsymbol{x}, y)\| \leq b(\boldsymbol{x}, y)$.*

Proof. The proof is as in Lemma 6. We only have to verify that Equation (2) can be solved with RBI and that the solution is bounded by a function in $\mathcal{B}_n$. By hypothesis, the function g in Equation (2) has degree 0 w.r.t. its last component. Let's denote be G the right hand side of Equation (2). Since G is a composition of g with functions in $\mathcal{B}_0$, then G has degree 0 w.r.t. y_1 and y_2. Therefore Equation (2) can be solved with RBI. It is straightforward to show that the solution of that equation is bounded in $\mathcal{B}_n$ using the arguments at the end of the proof of Lemma 6. □

Corollary 17 *$\mathcal{B}_n$ is closed under bounded sums in a weak sense.*

Proof. Let's suppose that $u \in \mathcal{B}_n$ and $\beta = 2^{[n]} \circ p$ is a bound on u. We wish to define $h \in \mathcal{B}_n$ such that $h(\boldsymbol{x}, y) = \sum_{t<y} u(\boldsymbol{x}, t)$ for $t, y \in \mathbb{N}$. We can do this with Lemma 16 setting $f = 0$ and $g(\boldsymbol{x}, y, z) = u(\boldsymbol{x}, y) + z$, where $\deg_{\mathrm{last}} g = 0$. Then we can define $h \in \mathcal{B}_n$ s.t. $h(\boldsymbol{x}, 0) = 0$, $h(\boldsymbol{x}, y+1) = u(\boldsymbol{x}, y) + h(\boldsymbol{x}, y)$, and $h(\boldsymbol{x}, y) \leq y\beta(\boldsymbol{x}, y)$. □

Bounded minimalization and bounded quantifiers can be derived from bounded sums even in classes below $\mathcal{F}$LINSPACE (cf. [19, p.118]). The following lemma follows directly from the fact that $\mathcal{B}_n$ is closed under bounded sums and restricted composition.

Lemma 18 *If $f, b \in \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$, with $\deg_{\text{last}} f = \deg_{\text{last}} b = 0$ and $g(\boldsymbol{x}, y) = \mu_{t \leq b(\boldsymbol{x},y)} f(\boldsymbol{x}, t)$, or $g(\boldsymbol{x}, y) = \forall_{t \leq b(\boldsymbol{x},y)}[f(\boldsymbol{x}, t) = 0]$, or $g(\boldsymbol{x}, y) = \exists_{t \leq b(\boldsymbol{x},y)}[f(\boldsymbol{x}, t) = 0]$, then $g \in \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$ and $\deg_{\text{last}} g = 0$.*

Next we generalize Proposition 7 to $\mathcal{B}_n$.

Proposition 19 *For all $n \in \mathbb{N}$, $\text{EXPSPACEF}^n \subset \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$.*

Proof. We will show that we can define Equation (5) in $\mathcal{B}_n$. First, notice that, with Lemma 16 we can define in $\mathcal{B}_n$ a function H that satisfies Equation (6). The remarks after Lemma 15 show that b_z can be defined in $\mathcal{B}_n$. Furthermore, $\deg_{\text{last}} H = 0$ since, in Equation (6), $\deg \text{NEXT} = 0$ and b_z doesn't depend on y. Since ACCEPT_z also has degree 0, then, from Lemma 18, $\mu_{y \leq b_z(\boldsymbol{x})} \text{ACCEPT}_z(H_z(\boldsymbol{x}, y))$ can be defined in $\mathcal{B}_n$, and its degree w.r.t. y is 0. Therefore, with RCOMP, we can define $H_z(\boldsymbol{x}, \mu_{y \leq b_z(\boldsymbol{x})} \text{ACCEPT}_z(H_z(\boldsymbol{x}, y)))$. Finally, we apply OUTPUT_z, which has degree 0, as in Equation (5). □

Next we give a converse result, which generalizes Proposition 11. Specifically, we show that every function in $\mathcal{B}_n$ is computable with precision 2^{-m} in space $2^{[n]}(x^{O(1)}) + O(m)$. But first, in analogy with Lemma 8, we have to set bounds on the functions in $\mathcal{B}_n$ and their derivatives.

Lemma 20 *If $f \in \mathcal{B}_n$ then there is a polynomial p s.t. f and its first and second derivatives are bounded by $2^{[n]} \circ p$.*

Proof. First, we need to show that if $\deg_i f = 0$, then f grows at most polynomially w.r.t. its ith argument. This has been already proved for basic functions in $\mathcal{B}_0$. If $h(\boldsymbol{x})$ is defined with RCOMP as $g(f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x}))$, it can only have $\deg_i h = 0$ if for all $j = 1, \ldots, p$, $\deg_j g = \deg_i f_j = 0$. Therefore, by induction hypothesis, h grows at most polynomially w.r.t. to x_i. Finally, if h is defined with RBI then $\deg_i h = 0$ only if $\deg_i b = 0$, where $h < b$. Since b grows at most polynomially w.r.t. to x_i by induction hypothesis, the same is true for h.

It is clear that any $f \in \mathcal{B}_n$ is bounded by $2^{[n]} \circ p$ for some polynomial p since the basic functions of $\mathcal{B}_n$ satisfy that bound and both RCOMP and RBI preserve it. For the derivatives, we know from Lemma 8 that basic functions in $\mathcal{B}_0$ satisfy the bound. The first and second derivative of the basic function $2^{[n]}$ are also bounded by $2^{[n]} \circ p$.

The derivatives are given by the chain rule if a function is defined by restricted composition. With RCOMP, $g(f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x}))$ can only be defined in $\mathcal{B}_n$ if for all j and i, $\deg g_j = 0$ or $\deg_i f_j = 0$. Therefore, each term in the chain rule has to be the product of a function bounded by a polynomial and a function bounded by $2^{[n]} \circ p$. So, the first derivative is bounded by $2^{[n]} \circ p$ for some polynomial p. The same argument holds for the second derivative.

For restricted bounded integration, we proceed as in the proof for $\mathcal{B}_0$. Once again, by definition of RBI, $\deg_z g(\boldsymbol{x}, y, z) = 0$ for g in Equation (4). Then $g(\boldsymbol{x}, y, h(\boldsymbol{x}, y))$, as a function of $\boldsymbol{x}$ and y, satisfies $g < 2^{[n]} \circ p$ for some p and its first and second derivatives also. Using once again differential inequalities,

adjusting the polynomials in the bound, and supposing that the result holds for the derivatives of $f, g \in \mathcal{B}_n$, we can prove that the derivatives of h defined by RBI also satisfy $h < 2^{[n]} \circ p$ for some p. This concludes the proof. □

Proposition 21 *If $f \in \mathcal{B}_n$, then its space complexity is $s(x,m) \in 2^{[n]}(x^{O(1)}) + O(m)$.*

Proof. Similarly to the proof of Proposition 11, we will also need to prove that if $f \in \mathcal{B}_n$, then $l_f(x,m) \in 2^{[n]}(x^{O(1)}) + O(m)$. We proceed by induction, starting with the basic functions of $\mathcal{B}_n$, i.e., $\mathcal{B}_0$ and $2^{[n]}$. From Proposition 11 we know that the result is true for the functions in $\mathcal{B}_0$.

Let's then consider $2^{[n]}$ and claim that $s_{2^{[n]}}(x,m), l_{2^{[n]}}(x,m) \in 2^{[n-1]}(x^{O(1)}) + O(m)$, where 2^{-m} is the precision. We first show that the exponential 2^y is computable with precision 2^{-m} in space $y^{O(1)} + O(m)$ and that the required input precision $l_{2^y} \in y^{O(1)} + O(m)$. Towards this end, we follow Lemma 10, when the differential equation to be solved is $h(0) = 1$ and $\partial_y h(y) = \log 2\, h(y)$ ($\log 2$ can be defined in $\mathcal{B}_0$). Notice that b_L is bounded by a constant, and $b_1, b_2 < 2^p$ for some polynomial p, and that the initial condition is computed exactly. Therefore, the required input precision is given by $N = m + 3 + \log b_1$, which is linear in m and polynomial in y. The largest number that occurs is $n = \tilde{y}\, b_2(\tilde{y})\, e^{b_L(\tilde{y})\,\tilde{y}}\, 2^{m+3}$. Since b_L is constant, then the size of n is linear both in y and m.

Next, we compose $g = 2^{[n]}$ with $f = 2^x$ to define $2^{[n+1]}$. We follow Lemma 9, where $f, b = 2^x$ and $g = 2^{[n]}$. Then, by induction hypothesis, $N = l_g(b(x), m) \in 2^{[n]}(x^{O(1)}) + O(m)$ and $M = l_f(x, N) \in x^{O(1)} + O(N) \in 2^{[n]}(x^{O(1)}) + O(m)$ as claimed. For the space complexity, $f(x)$ is computed with precision 2^{-N}, i.e., in space $x^{O(1)} + O(N) \in 2^{[n]}(x^{O(1)}) + O(m)$. Finally, $g(f(x))$ has to be computed with precision 2^{-m} and this can be done in space in $2^{[n-1]}(f(x)^{O(1)}) + O(m) \in 2^{[n]}(x^{O(1)}) + O(m)$ as claimed.

We still need to show that the operations RCOMP and RBI preserve the bounds on the required input precision and on the space complexity. Recall that if $f \in \mathcal{B}_n$, then there is polynomial p, such that $f, f', f'' < 2^{[n]} \circ p$.

For RCOMP, let's consider that $h = g(f(x))$, where $\deg g = 0$ or $\deg f = 0$. If $\deg f = 0$, then $f(x)$ is computable with precision 2^{-m} and $s_f, l_f \in x^{O(1)} + O(m)$ by Lemma 14. By hypothesis, $s_g, l_g \in 2^{[n]}(x^{O(1)}) + O(m)$. We can then proceed as we did above for $g = 2^{[n]}$ and $f = 2^x$, adjusting the bounds. If $\deg g = 0$, the result is proved similarly. We leave the details to the reader.

Finally, for RBI, let's suppose that $s_f, l_f, s_g, l_g \in 2^{[n]}(x^{O(1)}) + O(m)$, and consider the differential equation $h(x,0) = f(x), \partial_y h(x,y) = g(x,y,h)$. We follow Lemma 10, with $b_L < p$, since $\deg_{\text{last}} g = 0$, and $b_1, b_2 < 2^{[n]} \circ p$ for some polynomial p by Lemma 20. The required input precision is dominated by $M = l(x,\eta)$, with $\eta = m + 3 + 2y\, b_L(y)$. Since $b_L(y) \in y^{O(1)}$, then $M \in 2^{[n]}(x^{O(1)}) + y^{O(1)} + O(m)$. For the space complexity, notice that $f(x)$ has to be computed with precision $2^{-\eta}$. By induction hypothesis this can be done in space $2^{[n]}(x^{O(1)}) + O(\eta)$ which is still in $2^{[n]}(x^{O(1)}) + y^{O(1)} + O(m)$. Now, by definition of RBI, $\deg_{\text{last}} g = 0$. Therefore, from Lemma 14, $g(x,y,h)$ can be computed in space polynomial in h. Since $h < 2^{[n]} \circ p$ for some polynomial p, then the space needed

to compute $\mathbf{g}(\phi_x(M), t, h)$ is in $2^{[n]}(\|(x,y)\|^{O(1)}) + O(m)$. Finally, to write n, δ, t on the tape, which are of the order of $n = y\, b_2(y)\, e^{b_L(y)\, y}\, 2^{m+3}$, we need space in $2^{[n-1]}(y^{O(1)}) + O(m)$. This concludes the proof. □

4 Indefinite integrals and linear integration

The analog hierarchy described in the previous section is somewhat unsatisfactory since it is based on bounded integration, which imposes bounds on the solutions of the differential equations the system can solve. In fact, if the solution of the Cauchy problem in Definition 13 doesn't have a bound in the class, then it doesn't belong to the class. It would be more natural to have, instead, an integration operator which, given a pair of functions of appropriate arities and dimensions, would always define a solution for the corresponding Cauchy problem as, for instance, the linear integration operator. We would also like to eliminate the *a priori* bound given by the basic function $2^{[n]}$ in Definition 13.

Incidentally, we notice that the two modifications we suggest are similar to the ones that lead, from the original definition of $\mathcal{F}$PTIME by Cobham [9], to the predicative definition of $\mathcal{F}$PTIME by Bellantoni and Cook [2]. Cobham gave an arithmetic definition of $\mathcal{F}$PTIME, which was closed under bounded recursion on notation. Bellantoni and Cook gave an elegant and equivalent definition of $\mathcal{F}$PTIME that avoids the explicit size bounds and the initial bounding function in Cobham's definition, using a more structured form of recursion called predicative or safe. In fact, several recursive definitions of important computational complexity classes have been reworked using predicativity [1, 8]. Analogously, in the context of continuous-time computation, linear integration can be considered a more structured form of integration than the general integration operation.

We give our analog system the ability to integrate functions already defined and we allow it to solve up to n nested linear differential equations in the nth level of the hierarchy. We keep $\mathcal{B}_0$ as the pool of basic functions. Formally,

Definition 22 *(The hierarchy $\mathcal{S}_n$) For all $n \geq 1$,*

$$\mathcal{S}_n = [\mathcal{B}_0; \mathrm{RCOMP}, \mathrm{INT}, n \cdot \mathrm{LI}],$$

where

- *Basic functions have degree* 0.
- INT *denotes integral, i.e., given f of arity $n+1$, define g of arity $n+1$ as $g(\boldsymbol{x}, y) = \int_0^y f(\boldsymbol{x}, t)dt$, with $\deg_i g = 0$ iff $\deg_i f = 0$,*
- LI *denotes linear integration, i.e., if the functions $f_1, \ldots, f_m$ of arity n and $g_{11}, \ldots, g_{mm}$ of arity $n+1$ are defined, then the function $h = h_1$ of arity $n+1$, where $\boldsymbol{h} = (h_1, \ldots, h_m)$ satisfies the equations $\boldsymbol{h}(\boldsymbol{x}, 0) = \boldsymbol{f}(\boldsymbol{x})$ and $\partial_y \boldsymbol{h}(\boldsymbol{x}, y) = \boldsymbol{g}(\boldsymbol{x}, y)\boldsymbol{h}(\boldsymbol{x}, y)$, is defined.*

Notice that even if the composition $\exp(\exp(x))$ is not permitted, the iterated exponential $\exp^{[n]} = \exp \circ \ldots \circ \exp$ can be defined in $\mathcal{S}_n$. Let $u_i(\boldsymbol{x}, y) =$

$\exp^{[i]}(p(\boldsymbol{x}, y))$ for $i = 1, \ldots, n$, where p is a polynomial. Then, the functions u_i are defined by the set of linear differential equations

$$\partial_y u_1 = u_1 \cdot \partial_y p \qquad \ldots \qquad \partial_y u_n = u_n \cdot u_{n-1} \cdots u_1 \cdot \partial_y p$$

with appropriate initial conditions. Thus u_n can be defined with up to n nested applications of LI and, therefore, $\exp^{[n]} \circ p \in \mathcal{S}_n$.

Lemma 23 *$\mathcal{S}_n$ is closed under bounded sums in a weak sense.*

Proof. We define a step function F which matches f on the integers, and whose values are constant on the interval $[j, j + 1/2]$ for integer j. F can be defined as $F(t) = f(s(t))$, where s is the function in the proof of Proposition 6. The bounded sum of f is then easily defined by INT. Simply write $g(0) = 0$ and $g'(t) = c_k F(t)\, \theta_k(\sin 2\pi t)$, where c_k is an appropriate constant. Then, whenever $t \in [n - 1/2, n]$, $g(t) = \sum_{z<n} f(z)$ and, therefore, the bounded sums of f are given by the above integral on $[0, n]$. □

Then, similarly to $\mathcal{B}_n$ (cf. Lemma 18), we have the following.

Lemma 24 *If $f, b \in \mathcal{F}_{\mathbb{N}}(\mathcal{S}_n)$, with $\deg_{\text{last}} f = \deg_{\text{last}} b = 0$ and $g(\boldsymbol{x}, y) = \mu_{t \le b(\boldsymbol{x},y)} f(\boldsymbol{x}, t)$, or $g(\boldsymbol{x}, y) = \forall_{t \le b(\boldsymbol{x},y)}[f(\boldsymbol{x}, t) = 0]$, or $g(\boldsymbol{x}, y) = \exists_{t \le b(\boldsymbol{x},y)}[f(\boldsymbol{x}, t) = 0]$, then $g \in \mathcal{F}_{\mathbb{N}}(\mathcal{S}_n)$ and $\deg_{\text{last}} g = 0$.*

We wish to prove an analogue to Proposition 19. However, since we don't know if $\mathcal{S}_n$ is closed under bounded recursion, we cannot proceed as we did for $\mathcal{B}_n$. We still use the arithmetization of TM within $\mathcal{F}$LINSPACE given by Lemma 15 but we have to define the function H in Equation (5) with bounded sums instead of bounded recursion. This is a standard technique in recursion theory (cf. [19, p.11]) which relies on prime coding of the values of the function defined by bounded recursion. As a matter of fact, if h is defined by $h(\boldsymbol{x}, 0) = f(\boldsymbol{x})$, $h(\boldsymbol{x}, y + 1) = g(\boldsymbol{x}, y, h(\boldsymbol{x}, y))$, and $h(\boldsymbol{x}, y) \le b(\boldsymbol{x}, y)$, then it can be written as

$$h(\boldsymbol{x}, y) = \mu_{t \le b(\boldsymbol{x},y)}\ \exists_{m \le B(\boldsymbol{x},y)}\ [\psi(\boldsymbol{x}, y, m, t) = 0]. \tag{7}$$

$B(\boldsymbol{x}, y) = \prod_{u \le y} p(u)^{b(\boldsymbol{x},u)}$ and $\psi(\boldsymbol{x}, y, m, t) = 0$ iff $(m)_0 = f(\boldsymbol{x})$, $\forall_{v<y}[(m)_{v+1} = g(\boldsymbol{x}, y, (m)_v)]$ and $(m)_y = t$, where $(m)_u$ returns the exponent of the uth prime $p(u)$ in the prime factorization of m. Note that $(m)_u$ is clearly computable in linear space and therefore in $\mathcal{B}_0$, and the logical operations in the expression above are also [19]. The procedure above does the following: it checks for all values of m smaller than $B(\boldsymbol{x}, y)$, and stops when it finds one m that codes the sequence $\{h(\boldsymbol{x}, 0), \ldots, h(\boldsymbol{x}, y)\}$.

Proposition 25 *For all $n \in \mathbb{N}$, $\text{EXPSPACEF}^n \subset \mathcal{F}_{\mathbb{N}}(\mathcal{S}_{n+1})$.*

Proof. The function H in Lemma 15 can be defined with Equation (7) instead of the recursion in Equation (6).[5] To make things clearer, let's rewrite Equation (7) for H:

[5] To simplify notation, we will drop the indices in H_z, INIT_z, NEXT_z, and b_z.

$$H(\boldsymbol{x}, y) = \mu_{t \leq b(\boldsymbol{x})}\ \exists_{m \leq B(\boldsymbol{x})}\ [\psi(\boldsymbol{x}, y, m, t) = 0] \tag{8}$$

and $\psi(\boldsymbol{x}, y, m, t) = 0$ iff $(m)_0 = \text{INIT}(\boldsymbol{x})$, $\forall_{v<y}[(m)_{v+1} = \text{NEXT}((m)_v)]$ and $(m)_y = t$.

From Lemmas 15 and 24, and the fact that logical operators belong to $\mathcal{F}$LINSPACE, we know that ψ is in $\mathcal{F}_{\mathbb{N}}(\mathcal{B}_0)$. Therefore, $\deg \psi = 0$.

We can take for the bound b the function $2^{[n]} \circ p$ for a certain polynomial p, as noticed in the remarks following Lemma 15. Since, for all $\boldsymbol{x}$, $b(\boldsymbol{x})$ is a bound on the number N of steps of the computation, then we can set the following bound B on the prime coding function of the sequence $\{H(\boldsymbol{x}, 0), \ldots, H(\boldsymbol{x}, N)\}$:

$$2^{H(\boldsymbol{x},0)} \ldots p_N^{H(\boldsymbol{x},N)} \leq 2^{[n+1]}(p(3\boldsymbol{x})) = B(\boldsymbol{x}),$$

where p is some polynomial. Therefore, B belongs to $\mathcal{F}_{\mathbb{N}}(\mathcal{S}_{n+1})$. Let's see then how to define $f \in \mathcal{F}_{\mathbb{N}}(\mathcal{S}_{n+1})$ with Equation (5). First, define H as in Equation (8) using Lemma 24. Since b and B are independent of y then $\deg_{\text{last}} H = 0$. Now, we define with RCOMP, $F_1(\boldsymbol{x}, y) = \text{ACCEPT}(H(\boldsymbol{x}, y))$, with still $\deg_{\text{last}} F_1 = 0$. Therefore, we can apply the bounded minimization operator to F_1 to define $F_2(\boldsymbol{x}) = \mu_{y \leq b(\boldsymbol{x})} F_1(\boldsymbol{x}, y)$. Finally,

$$f(\boldsymbol{x}) = \text{OUTPUT}\big(H(\boldsymbol{x}, F_2(\boldsymbol{x}))\big).$$

We have then showed that any function in EXPSPACEF^n has an extension to the reals in $\mathcal{S}_{n+1}$, as claimed. □

Finally, we want to show that $\mathcal{S}_n \subset \mathcal{B}_n$ and, therefore, $\mathcal{S}_n$ has the space complexity upper bounds given by Proposition 21.

Proposition 26 *For $n \geq 1$, $\mathcal{S}_n \subset \mathcal{B}_n$.*

Proof. The initial functions of $\mathcal{S}_n$ belong to $\mathcal{B}_n$. We just have to show that $\mathcal{B}_n$ is closed under INT and LI. Linear integration, where $\partial_y h(\boldsymbol{x}, y) = g(\boldsymbol{x}, y)h(\boldsymbol{x}, y)$, can be defined by restricted bounded integration since the term on the right hand side has degree 0 with respect to h. The operation INT is just a particular case of linear integration. Therefore, we just have to show that the functions LI and INT define are bounded by a function in $\mathcal{B}_n$. It is easy to verify that all functions in $\mathcal{S}_n$ have bounds $2^{[n]} \circ p \in \mathcal{B}_n$, since linear integration increases those bounds at most exponentially (cf. [7]) and, by definition, only n nested uses of LI are allowed in $\mathcal{S}_n$. □

5 Final remarks

We established lower and upper bounds on the computational complexity of two hierarchies of continuous time functions. In particular for the hierarchy $\mathcal{B}_n$ we showed that: (1) for all $n \in \mathbb{N}$, EXPSPACEF$^n \subset \mathcal{F}_{\mathbb{N}}(\mathcal{B}_n)$; and (2) for all $f \in \mathcal{B}_n$, $f(x)$ can be computed with precision 2^{-m} in space in $2^{[n]}(x^{O(1)}) + O(m)$. For

$\mathcal{S}_n$ we proved a weaker lower bound, namely $\text{EXPSPACEF}^n \subset \mathcal{F}_{\mathbb{N}}(\mathcal{S}_{n+1})$, and the same upper bound. It seems surprising that the space complexity grows only linearly with the number of bits of precision. For a polynomial-time computable function f, it is conjectured that the solution of the problem $y(0) = 0$ and $y' = f(x, y)$ requires polynomial space [13]. However, we consider in this paper only functions with certain bounds on the Lipschitz constant and on the derivatives.

As in [13], we defined space complexity of real functions as a measure depending on two parameters: the argument of the function and the number of bits of precision. The corresponding *non-uniform* complexity measure for functions defined on unbounded intervals depends only on the precision. We say that the space complexity of f is bounded by $s(m)$ if $s_f(x, m) \leq s(m)$ for all $x \in [-2^m, 2^m]$. With such definition, the upper bound we proved on non-uniform space complexity of functions in $\mathcal{B}_n$ or $\mathcal{S}_n$ is $2^{[n+1]}(O(m))$.

The following questions suggest themselves.

1. Is it possible to compute any function in $\mathcal{B}_0$ in space $(\log x)^{O(1)} + O(n)$? We saw this is true for the basic functions of $\mathcal{B}_0$ and this is preserved by composition. However, we were not able to get rid of the polynomial bound on the argument of the function for the numerical integration. If the answer to this question happened to be true, then tighter upper bounds for the hierarchy $\mathcal{B}_n$ would follow.
2. Can we remove bounded integration from the definition of the hierarchy $\mathcal{S}_n$? This is related to the following open problem in recursion theory: do we still obtain the same class if we replace bounded recursion by bounded sums in the recursive definition of $\mathcal{F}$LINSPACE given by Lemma 5?
3. Can we decrease the upper bounds on the classes $\mathcal{S}_n$? We believe that this is possible since $\mathcal{S}_n$ is closed under indefinite integrals and up to a certain number of linear integrations. Therefore, simpler numerical procedures then Euler's method can be used to approximate the functions of those classes. We leave this as a problem to the reader.

Our results are about space complexity of analog classes. What can we say about time complexity? Using the standard Euler's method as in Lemma 10, the bound on the time complexity would increase with the length of the recursive description of a function in $\mathcal{B}_n$. We could do better if we consider only indefinite integrals or linear differential equations. However, our definition of $\mathcal{S}_n$ still includes bounded integration, and, therefore, we weren't able to derive time complexity bounds for the analog classes we considered.

The results in this paper, combined with previous results in [7], give analog characterizations of space complexity classes that range from $\mathcal{F}$LINSPACE to the primitive recursive functions and show that when integral systems have the ability to solve differential equations of increasing generality their power increases along that range.

Acknowledgements. A previous version of this work appeared in the author's PhD dissertation, supported by FCT via Laboratório de Modelos e Arquitecturas Computacionais and grant PRAXIS XXI/BD/18304/98, and supervised by Félix

Costa and Cris Moore. I wish to thank them both for their most valuable suggestions and remarks. I thank Félix Costa in particular for his collaboration on some proofs in this paper. I also wish to thank Geoff Ostrin, Amílcar Sernadas and Vasco Brattka for useful discussions. This work has been supported by FCT and FEDER via the Center for Logic and Computation and the project ConTComp (POCTI/MAT/45978/2002).

References

1. S. Bellantoni. Predicative recursion and the polytime hierarchy. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 15–29. Birkhäuser, 1995.
2. S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.
3. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21:1–46, 1989.
4. O. Bournez. *Complexité Algorithmique des Systèmes Dynamiques Continus et Hybrides*. PhD thesis, École Normale Supérieure de Lyon, 1999.
5. M.S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100, 1995.
6. M.L. Campagnolo, C. Moore, and J.F. Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642–660, 2000.
7. M.L. Campagnolo, C. Moore, and J.F. Costa. An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 4(18):977–1000, 2002.
8. P. Clote. Computational models and function algebras. In E.R. Griffor, editor, *Handbook of Computability Theory*, pages 589–681. Elsevier, 1999.
9. A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, pages 24–30. North Holland, 1964.
10. P. Hartman. *Ordinary Differential Equations*. Birkhäuser, 2nd edition, 1982.
11. P. Henrici. *Discrete Variable Methods in Ordinary Differential Equations*. Wiley, New York, 1962.
12. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addisson-Wesley, 1979.
13. K.-I. Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991.
14. W. Miller. Recursive function theory and numerical analysis. *Journal of the ACM*, 4:465–472, 1970.
15. C. Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162:23–44, 1996.
16. P. Odifreddi. *Classical Recursion Theory II*. Elsevier, 2000.
17. P. Orponen. A survey of continuous-time computation theory. In D.-Z. Du and K.-I Ko, editors, *Advances in Algorithms, Languages, and Complexity*, pages 209–224. Kluwer Academic Publishers, Dordrecht, 1997.
18. R.W. Ritchie. Classes of predictably computable functions. *Transactions Amer. Math. Soc.*, 106:139–173, 1963.
19. H.E. Rose. *Subrecursion: Functions and Hierarchies*. Clarendon Press, 1984.
20. H. Siegelmann. *Neural Netwoks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, 1999.
21. J. Traub and A.G. Werschulz. *Complexity and Information*. Cambridge University Press, 1998.