


A New Characterization of \mathbf{FAC}^0 via Discrete Ordinary Differential Equations

Melissa Antonelli  

HIIT & University of Helsinki, Finland

Arnaud Durand  

Université Paris Cité, France

Juha Kontinen  

University of Helsinki, Finland

Abstract

Implicit computational complexity is an active area of theoretical computer science, which aims at providing machine-independent characterizations of relevant complexity classes. One of the seminal works in this field appeared in 1965, when Cobham introduced a function algebra closed under bounded recursion on notation to capture \mathbf{FP} . Later on, several complexity classes have been characterized using *limited* recursion schemas. In this context, a new approach was recently introduced, showing that ordinary differential equations (ODEs) offer a natural tool for algorithmic design and providing a characterization of \mathbf{FP} by an ODE-schema. The overall goal of the present work is precisely that of generalizing this approach to parallel computation, obtaining an original ODE-characterization for the small circuit classes \mathbf{FAC}^0 and \mathbf{FTC}^0 .

2012 ACM Subject Classification Theory of computation \rightarrow Complexity classes; Theory of computation \rightarrow Circuit complexity

Keywords and phrases Implicit computational complexity, parallel computation, ordinary differential equations, circuit complexity

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.10

Funding *Melissa Antonelli*: Supported by HIIT and Maupertuis' SMR Programme, on behalf of the Institut Français in Helsinki, the French Embassy to Finland, the French Ministry of Higher Education, Research and Innovation in partnership with the Finnish Society for Science and Letters and the Finnish Academy of Sciences and Letters.

Arnaud Durand: Supported by Maupertuis' SMR Programme.

Juha Kontinen: Supported by the Academy of Finland grant 345634.

1 Introduction

As computability theory investigates the limits of what is algorithmically computable, complexity theory classifies functions based on the amount of resources, typically time and space, required by a machine to compute them. Taking a different point of view, implicit computational complexity (ICC) aims at providing machine-independent characterizations, which in turn have offered remarkable insights on the corresponding classes and led to relevant meta-theorems in various domains, such as database theory and constraint satisfaction.

One of the major approaches to computability and (implicit) complexity theory is constituted by the study of recursion. A foundational work in this area is due to Cobham [15], who gave a characterization of the class of poly-time computable functions \mathbf{FP} , relying on a restricted recursion mechanism, called bounded recursion on notation (BRN). This groundbreaking result has inspired several characterizations based on *limited* recursion schemas for various other classes, but also alternative implicit ways to capture \mathbf{FP} , for instance, via safe recursion [5] and ramification [25, 26].



© Melissa Antonelli, Arnaud Durand, and Juha Kontinen;
licensed under Creative Commons License CC-BY 4.0

49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024).

Editors: Rastislav Královic and Antonín Kučera; Article No. 10; pp. 10:1–10:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Cobham’s work [15], together with other early results in recursion theory [20, 6, 29, 27], have even paved the way to recursion-theoretic characterizations for small circuit classes [11, 12, 16, 1, 14, 9]. Specifically, in [11, 12], an algebra based on the so-called *concatenation recursion on notation* schema (CRN) was introduced and shown able to capture functions computable in (**Dlogtime**-uniform) \mathbf{AC}^0 (i.e. computable by families of polynomial size and constant depth circuits) [11, 12]. This result was extended to \mathbf{AC}^i and \mathbf{NC}^i due to the notions of *k*-bounded and *weak bounded recursion on notation* [12]. A few years later, a similar function algebra to capture \mathbf{TC}^0 was introduced [14]. Other characterizations for subclasses of \mathbf{NC} were independently presented in [16, 1, 9].

Related, but alternative, approaches to capture small circuit classes have also been provided in the framework of model- and proof-theory. In particular, it is well-known that there is an equivalence between \mathbf{AC}^0 and first-order logic, which naturally generalizes to extensions of the latter and larger circuit classes [23, 4, 22, 21, 28]. In [16], both a function algebra, based on so-called *upward recursion tree*, and a logical system to capture \mathbf{NC}^1 are presented. On the proof-theoretical side, in [1], together with the corresponding algebra, Allen defined a proof system *à la Buss* to capture \mathbf{NC} . Another bounded theory to capture \mathbf{NC} is introduced in [13] and then extended to several small circuit classes [14]. Theories for \mathbf{TC}^0 have been developed in [24] and, in the setting of second-order theories, in [18]. Alternative, proof-theoretical characterizations for \mathbf{NC}^1 were presented in [2], and, in the context of two-sorted theories, in [17].

A different descriptive approach to complexity, based on discrete ordinary differential equations (ODEs), was recently introduced in [10]. Informally speaking, its objective is to characterize functions computable in a given complexity class as solutions of a corresponding type of ODE. In this vein, in [10] a purely syntactic characterization of \mathbf{FP} was given by linear systems of equations deriving along a logarithmically growing function. Intuitively, the latter condition controls the number of steps, while linearity controls the growth of objects generated during the computation. Recently, this approach has also been generalized to the continuous setting [7, 8].

Although small circuit classes have been characterized in multiple ways, it is an interesting and challenging question whether they can be studied through an ODE-based approach and whether this would shed some new light on these well-known classes. Interesting, because for a descriptive approach based on ODEs to make sense and be fruitful it has to be able to cope with very subtle and restricted modes of computation. Challenging, because even simple and useful mathematical functions may not be computable (e.g. multiplication is not in \mathbf{AC}^0), thus tools at hand and the naturalness of the approach are drastically restricted. In the present paper, we investigate these questions, and show that, in fact, natural ways to introduce ODE-oriented function algebras to capture small circuit classes can be found. Our approach relies on the introduction of ODE-schemas, still using derivation along the logarithmic function and allowing for bit shifting operations through restricted forms of linear equations. Our case study focuses on the smallest classes \mathbf{AC}^0 and \mathbf{TC}^0 , but is intended as the first step towards a uniform characterization of other relevant classes in the \mathbf{AC} and \mathbf{NC} hierarchies.

Structure of the paper. The paper is divided into two main sections. In Section 2, we introduce notions and results at the basis of our ODE-style characterizations. In particular, in Section 2.1, we summarize salient aspects of the approach by [10], which we aim to generalize from the study of \mathbf{FP} to that of small circuit classes. In Section 2.2, we briefly recap basic notions in parallel complexity and recall the function algebra approach of [11, 12, 14]

to capture \mathbf{AC}^0 and \mathbf{TC}^0 . In Section 3, we present our ODE-characterizations for the mentioned classes. Specifically, in Section 3.1, we introduce restricted ODE-schemas, and, using them, in Sections 3.2 and 3.3, we define ODE-function algebras capturing the analog for functions of \mathbf{AC}^0 and \mathbf{TC}^0 , respectively. Then, in Section 3.4, we provide alternative and direct completeness proofs for both classes in the non-uniform setting, namely assuming that functions describing the circuits are given as basic functions. Finally, in Section 4, we briefly point at possible directions of future research.

2 Preliminaries

2.1 Capturing Complexity Classes via ODEs

We suppose the reader familiar with the basics of complexity theory [3, 30]. In order to introduce the approach to complexity delineated in [10], we start by cursorily recalling Cobham's result, capturing \mathbf{FP} by the following BRN schema.

► **Definition 1** (Bounded Recursion on Notation, BRN). *Given $g : \mathbb{N}^p \rightarrow \mathbb{N}$, $k : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ and $h_i : \mathbb{N}^{p+2} \rightarrow \mathbb{N}$, with $i \in \{0, 1\}$, $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ is defined by BRN from g, h_0, h_1 and k if*

$$f(0, \mathbf{y}) = g(\mathbf{y}) \quad \text{and} \quad f(s_i(x), \mathbf{y}) = h_i(f(x, \mathbf{y}), x, \mathbf{y}), \text{ for } x \neq 0$$

with $f(x, \mathbf{y}) \leq k(x, \mathbf{y})$ and $s_j(x) = 2x + j$ ($j \in \{0, 1\}$) being the binary successor functions.

The growth of f is controlled by the function k (in \mathbf{FP}), while the number of induction steps is kept under control by the application of the binary successor functions s_i . Such a schema is not fully satisfactory as it imposes an explicit bound on recursion in the form of an already known function.

As anticipated, Cobham's paper not only led to a variety of implicit characterizations for classes other than \mathbf{FP} , but also inspired alternative approaches to capture this class. Among them, the proposal by [10] has the specificity of not imposing any explicit bound on the recursion schema or assigning specific role to variables. Instead, it is based on Discrete Ordinary Differential Equation (a.k.a. Difference Equations) and combines two peculiar features: derivation along specific functions, so to control the number of computation steps, and a special syntactic form of the equation itself (here linearity), allowing to control the object size. We present the basics of the method as necessary to formulate our new results.

Recall that the *discrete derivative* of $\mathbf{f}(x)$ is defined as $\Delta \mathbf{f}(x) = \mathbf{f}(x+1) - \mathbf{f}(x)$, and that ODEs are expressions of the form:

$$\frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial x} = \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}),$$

where $\frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial x}$ stands for the derivative of $\mathbf{f}(x, \mathbf{y})$ considered as a function of x , for a fixed value for \mathbf{y} . When some initial value $\mathbf{f}(0, \mathbf{y}) = \mathbf{g}(\mathbf{y})$ is added, this is called *Initial Value Problem* (IVP). Then, in order to deal with complexity, some restrictions are needed. First, the notion of derivation has to be generalized, allowing to *derive along functions*.

► **Notation 1.** For $x \neq 0$, let $\ell(x)$ denote the length of x written in binary, i.e. $\lceil \log_2(x+1) \rceil$, and $\ell(0) = 0$. For $u \geq 0$, $\alpha(u) = 2^u - 1$ denotes the greatest integer the binary length of which is u . Also, we use $\div 2$ to denote integer division by 2, i.e. for all $x \in \mathbb{Z}$, $x \div 2 = \lfloor \frac{x}{2} \rfloor$.

► **Definition 2** (λ -ODE Schema). *Let $\mathbf{f}, \lambda : \mathbb{N}^p \rightarrow \mathbb{Z}$ and $\mathbf{h} : \mathbb{N}^{p+1} \rightarrow \mathbb{Z}$ be functions. Then,*

$$\frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \lambda} = \frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \lambda(x, \mathbf{y})} = \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}) \quad (1)$$

is a formal synonym of $\mathbf{f}(x+1, \mathbf{y}) = \mathbf{f}(x, \mathbf{y}) + (\lambda(x+1, \mathbf{y}) - \lambda(x, \mathbf{y})) \times \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y})$. When $\lambda(x, \mathbf{y}) = \ell(x)$, we call (1) a length-ODE.

10:4 A New Characterization of FAC⁰ via Discrete ODEs

Intuitively, one of the key properties of the λ -ODE schema is its dependence on the number of distinct values of λ , i.e., the value of $\mathbf{f}(x, \mathbf{y})$ changes only when the value of $\lambda(x, \mathbf{y})$ does.

The computation of solutions of λ -ODEs have been fully described in [10]. Here, we focus on the special case of $\lambda = \ell$, which is particularly relevant for our characterizations. First, observe that the value of $\ell(x)$ changes (i.e. increases by 1) when x goes from $2^t - 1$ to 2^t , i.e. from $\alpha(t)$ to $\alpha(t) + 1$. So, if \mathbf{f} is a solution of (1) with $\lambda = \ell$ and initial value $\mathbf{f}(0, \mathbf{y}) = \mathbf{g}(\mathbf{y})$, then $\mathbf{f}(1, \mathbf{y}) = \mathbf{f}(0, \mathbf{y}) + \mathbf{h}(\mathbf{f}(\alpha(0), \mathbf{y}), \alpha(0), \mathbf{y})$, and, more generally, for all x and \mathbf{y} , $\mathbf{f}(x, \mathbf{y}) = \mathbf{f}(x-1, \mathbf{y}) + \Delta(\ell(x-1)) \times \mathbf{h}(\mathbf{f}(x-1, \mathbf{y}), x-1, \mathbf{y}) = \mathbf{f}(\alpha(\ell(x)-1), \mathbf{y}) + \mathbf{h}(\mathbf{f}(\alpha(\ell(x)-1), \mathbf{y}), \alpha(\ell(x)-1), \mathbf{y})$, where $\Delta(\ell(t-1)) = \ell(t) - \ell(t-1)$. Starting from $t = x \geq 1$ and taking successive decreasing values of t , the first difference such that $\Delta(t) \neq 0$ is given by the biggest $t-1$ such that $\ell(t-1) = \ell(x) - 1$, i.e. $t-1 = \alpha(\ell(x)-1)$. Hence, by induction, it is established that:

$$\mathbf{f}(x, \mathbf{y}) = \sum_{u=-1}^{\ell(x)-1} \mathbf{h}(\mathbf{f}(\alpha(u), \mathbf{y}), \alpha(u), \mathbf{y}),$$

with $\mathbf{h}(\cdot, \alpha(-1), \mathbf{y}) = \mathbf{f}(0, \mathbf{y})$ and, as seen, $\alpha(u) = 2^u - 1$.

The second crucial novelty is the introduction of a special concept of *linearity*, utilized to control the growth of functions defined by ODEs. First, we present the notion of *degree for a polynomial expression*, which is a generalized (and slightly modified) version of the corresponding definition in [10]. Here, the degree of an expression is considered in relation to a set of variables, instead of a single one.

Let $\text{sg} : \mathbb{Z} \rightarrow \mathbb{Z}$ be the sign function over \mathbb{Z} , taking value 1 for $x > 0$ and 0 otherwise.

► **Definition 3.** A **sg**-polynomial expression $P(x_1, \dots, x_h)$ is an expression built over the signature $\{+, -, \times\}$, the **sg** function and a set of variables $X = \{x_1, \dots, x_h\}$ plus integer constants. Given a list of variables $\mathbf{x} = x_{i_1}, \dots, x_{i_m}$, for $i_1, \dots, i_m \in \{1, \dots, h\}$ the degree of a set \mathbf{x} in a **sg**-polynomial expression P , $\text{deg}(\mathbf{x}, P)$, is inductively defined as follows:

- $\text{deg}(\mathbf{x}, P) = 0$ for P constant
- $\text{deg}(\mathbf{x}, x_{i_j}) = 1$, for $x_{i_j} \in \{x_{i_1}, \dots, x_{i_m}\}$, and $\text{deg}(\mathbf{x}, x_{i_j}) = 0$, for $x_{i_j} \notin \{x_{i_1}, \dots, x_{i_m}\}$
- $\text{deg}(\mathbf{x}, P + Q) = \text{deg}(\mathbf{x}, P - Q) = \max\{\text{deg}(\mathbf{x}, P), \text{deg}(\mathbf{x}, Q)\}$
- $\text{deg}(\mathbf{x}, P \times Q) = \text{deg}(\mathbf{x}, P) + \text{deg}(\mathbf{x}, Q)$
- $\text{deg}(\mathbf{x}, \text{sg}(P)) = 0$.

A **sg**-polynomial expression P is said to be *essentially constant* in a set of variables \mathbf{x} when $\text{deg}(\mathbf{x}, P) = 0$. A **sg**-polynomial expression P is said to be *essentially linear* in a set \mathbf{x} , when $\text{deg}(\mathbf{x}, P) = 1$.

In what follows, we consider functions $\mathbf{f} : \mathbb{N}^{p+1} \rightarrow \mathbb{Z}^d$, i.e. vectors of functions $\mathbf{f} = f_1, \dots, f_d$ from \mathbb{N}^{p+1} to \mathbb{Z} , and we introduce the linear λ -ODE schema.

► **Definition 4 (Linear λ -ODE).** Given $\mathbf{g} : \mathbb{N}^p \rightarrow \mathbb{N}^d$, $\mathbf{h} : \mathbb{N}^{p+1} \rightarrow \mathbb{Z}$ and $\mathbf{u} : \mathbb{Z} \times \mathbb{N}^{p+1} \rightarrow \mathbb{Z}^d$, the function $\mathbf{f} : \mathbb{N}^{p+1} \rightarrow \mathbb{Z}^d$ is linear λ -ODE definable from \mathbf{g} , \mathbf{h} and \mathbf{u} if it is the solution of the IVP with initial value $\mathbf{f}(0, \mathbf{y}) = \mathbf{g}(\mathbf{y})$ and such that:

$$\frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \lambda} = \mathbf{u}(\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y}),$$

where \mathbf{u} is essentially linear in $\mathbf{f}(x, \mathbf{y})$. For $\lambda = \ell$, such schema is called linear length ODE.

If \mathbf{u} is *essentially linear* in $\mathbf{f}(x, \mathbf{y})$, there exist matrices \mathbf{A} and \mathbf{B} , whose coefficients are essentially constant in $\mathbf{f}(x, \mathbf{y})$ and such that:

$$\begin{aligned} \mathbf{f}(0, \mathbf{y}) &= \mathbf{g}(\mathbf{y}) \\ \frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \ell} &= \mathbf{A}(\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y}) \times \mathbf{f}(x, \mathbf{y}) + \mathbf{B}(\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y}). \end{aligned}$$

Then, for all x and \mathbf{y} ,

$$\mathbf{f}(x, \mathbf{y}) = \sum_{u=-1}^{\ell(x)-1} \left(\prod_{t=u+1}^{\ell(x)-1} \left(1 + \mathbf{A}(\mathbf{f}(\alpha(t), \mathbf{y}), \mathbf{h}(\alpha(t), \mathbf{y}), \alpha(t), \mathbf{y}) \right) \right) \times \mathbf{B}(\mathbf{f}(\alpha(u), \mathbf{y}), \mathbf{h}(\alpha(u), \mathbf{y}), \alpha(u), \mathbf{y}),$$

with the convention that $\prod_x^{x-1} \kappa(x) = 1$ and $\mathbf{B}(\cdot, \cdot, \alpha(-1), \mathbf{y}) = \mathbf{f}(0, \mathbf{y})$.

► **Example 5** (Function $2^{\ell(x)}$). The function $x \mapsto 2^{\ell(x)}$ can be seen as the solution of the IVP with initial value $f(0) = 1$ and such that $\frac{\partial f(x)}{\partial \ell} = f(x)$, i.e. where $A = 1$, $B = 0$. Indeed, the solution of this system is of the form $f(x) = \prod_{u=0}^{\ell(x)-1} 2 = 2^{\ell(x)}$. In addition, the function $(x, y) : x, y \mapsto 2^{\ell(x)} \times y$ can be captured by the same equation, with initial value $f(0, y) = y$.

One of the main results of [10] is the implicit characterization of \mathbf{FP} by the algebra made of basic functions $0, 1, \pi_i^p, \ell, +, -, \times, \mathbf{sg}$ and closed under composition (\circ) and ℓ -ODE:

$$\mathbf{LDL} = [0, 1, \pi_i^p, \ell, +, -, \times, \mathbf{sg}; \circ, \ell\text{-ODE}].$$

2.2 On Parallel Computation and Function Algebras for \mathbf{FAC}^0

Boolean circuits are vertex-labeled directed acyclic graphs whose nodes are either *input nodes* (no incoming edges), *output nodes* (no outgoing edges) or labeled with a Boolean function from the set $\{\wedge, \vee, \neg\}$. A Boolean circuit with majority gates allows in addition gates labeled by the function MAJ, that outputs 1 when the majority of its inputs are 1's. A family of circuits (C_n) is **Dlogtime**-uniform if there is a Turing machine (with a random access tape) that decides in deterministic logarithmic time the *direct connection language of the circuit*, i.e. which, given 1^n , a, b and $t \in \{\wedge, \vee, \neg\}$, decides if a is of type t and b is a predecessor of a in the circuit (and analogously for input and output nodes). When dealing with circuits, the resources of interests are *size*, i.e. the number of its gates, and *depth*, i.e. the length of the longest path from the input to the output (see [30] for more details and related results).

► **Definition 6** (Classes \mathbf{FAC}^i and \mathbf{FTC}^i). For $i \in \mathbb{N}$, the class \mathbf{AC}^i (resp., \mathbf{TC}^i) is the class of languages recognized by a **Dlogtime**-uniform family of Boolean circuits (resp. circuits including majority gates) of polynomial size and depth $O((\log n)^i)$. We denote by \mathbf{FAC}^i and \mathbf{FTC}^i the corresponding function classes.

For \mathbf{FAC}^0 , a particularly relevant recursion-theoretic characterization was provided by Clote [11, 12]. It relies on the schema of concatenation recursion on notation.

► **Definition 7** (Concatenation Recursion on Notation, CRN). A function f is defined by concatenation recursion on notation from g and h_i , with $i \in \{0, 1\}$, if for all x, \mathbf{y} :

$$f(0, \mathbf{y}) = g(\mathbf{y}) \quad \text{and} \quad f(s_i(x), \mathbf{y}) = s_{h_i(x, \mathbf{y})}(f(x, \mathbf{y})), \quad \text{for } x \neq 0.$$

Then, Clote's function algebra is defined as the class:

$$\mathcal{A}_0 = [0, \pi_i^p, s_0, s_1, \ell, \mathbf{BIT}, \#, \circ, \text{CRN}],$$

where $\mathbf{BIT}(x, y)$ returns the y^{th} value in the binary representation of x , and $x\#y = 2^{\ell(x) \times \ell(y)}$ is the smash function. This class was also proved able to capture \mathbf{FAC}^0 [11, 12].

► **Theorem 8** ([11, 12]). $\mathcal{A}_0 = \mathbf{FAC}^0$.

That $\mathcal{A}_0 \subseteq \mathbf{FAC}^0$ is proved by passing through the (function version of the) logarithmic time hierarchy \mathbf{FLH} , which is known to be equivalent to \mathbf{FAC}^0 . Then, $\mathcal{A}_0 \subseteq \mathbf{FLH}$ is established by showing that basic functions are computable in log-time and that \mathbf{FLH} is closed under composition and CRN. That $\mathbf{FLH} \subseteq \mathcal{A}_0$ is proved by the arithmetization of log-time bounded RAM. Remarkably, in [14], these results are even generalized to \mathbf{FTC}^0 , which is proved equivalent to the function algebra:

$$\mathcal{TC}_0 = [0, \pi_i^p, s_0, s_1, \ell, \text{BIT}, \times, \#, \circ, \text{CRN}].$$

3 Towards an ODE-Characterization of \mathbf{FAC}^0

In this section, we provide the first implicit characterization of \mathbf{FAC}^0 in the ODE-setting. We start by introducing the new ODE-schemas which are at the basis of our characterizations of \mathbf{FAC}^0 and \mathbf{FTC}^0 and intuitively corresponding to left- and right-shifting (Sec. 3.1). Due to them, we introduce the function algebra \mathbf{ACDL} , the defining feature of which is precisely the presence of these special ODE-schemas, and prove this class able to capture \mathbf{FAC}^0 (Sec. 3.2). This is established passing through Clote's \mathcal{A}_0 . As a byproduct, we obtain a similar ODE-characterization for \mathbf{FTC}^0 (Sec. 3.3). Finally, an alternative, direct proof of completeness is provided for both classes in a non-uniform setting (Sec. 3.4).

► **Remark 9.** Here, functions can take images in \mathbb{Z} . Accordingly, a convention for the binary representation of integers must be adopted, e.g. by assuming that, in any binary sequence, the first bit indicates the sign. Then, all arithmetic operations can be easily re-designed to handle encodings of possibly negative integers by circuits of the same size and depth.

3.1 Discrete ODE-Schemas for Shifting

We start by considering the ODE-schemas which are at the basis of our characterizations of \mathbf{FAC}^0 and \mathbf{FTC}^0 . Observe that they will sometimes include \times . This is admissible since, as we shall see, the “kind of multiplication” we consider is actually limited to special cases (namely, multiplication by 2^i), which are proved to be computable in \mathbf{FAC}^0 .

The ℓ -ODE₁ and ℓ -ODE₂ Schemas. We start with the limited ℓ -ODE₁ schema, intuitively corresponding to left shifting(s) and (possibly) adding a bit.

► **Definition 10** (ℓ -ODE₁ Schema). *Given $g : \mathbb{N}^p \rightarrow \mathbb{N}$ and $h : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$, such that h takes values in $\{0, 1\}$ only, the function $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ is defined by ℓ -ODE₁ from functions g and h when it is the solution of the IVP with initial value $f(0, \mathbf{y}) = g(\mathbf{y})$ and such that:*

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = f(x, \mathbf{y}) + h(x, \mathbf{y}).$$

The definition of the function $2^{\ell(x)}$ (and $2^{\ell(x)} \times y$) given in Example 5 is a special case of a ℓ -ODE₁ (with $h(x, \mathbf{y}) = 0$).

► **Remark 11.** An equivalent, purely-syntactical formulation of Definition 10 is obtained by substituting the explicit constraint that $h(x, \mathbf{y}) \in \{0, 1\}$ with the assumption that $h(x, \mathbf{y})$ is of the form $\text{sg}(B(h_1, \dots, h_m, x, \mathbf{y}))$, where B is an expression built over the signature $\{+, -, \div, 2, \text{sg}\}$ and calling previously defined \mathbf{FAC}^0 functions $h_1(x, \mathbf{y}), \dots, h_m(x, \mathbf{y})$.

In terms of circuits, this schema intuitively allows us to iteratively left-shifting (the binary representation of) a given number, each time possibly adding 1 to its final position. This is clarified by the proof below, establishing that \mathbf{FAC}^0 is closed under the mentioned schema.

► **Proposition 12.** *If f is defined by ℓ -ODE₁ from g and h in \mathbf{FAC}^0 , then f is in \mathbf{FAC}^0 .*

Proof Sketch. By Def. 10, for all x and \mathbf{y} : $f(x, \mathbf{y}) = \sum_{u=-1}^{\ell(x)-1} \left(\prod_{t=u+1}^{\ell(x)-1} 2 \right) \times h(\alpha(u), \mathbf{y}) = \sum_{u=-1}^{\ell(x)-1} 2^{\ell(x)-u-1} \times h(\alpha(u), \mathbf{y})$, with the convention that $\alpha(u) = 2^u - 1$, $\prod_x^{x-1} \kappa(x) = 1$ and $h(\alpha(-1), \mathbf{y}) = f(0, \mathbf{y})$. Clearly, the multiplication here involved is always by a power of 2 (which basically corresponds to left-shifting, so is computable in \mathbf{FAC}^0). Since $h(x, \mathbf{y}) \in \{0, 1\}$, the outermost sum amounts to a concatenation (which again can be computed in \mathbf{FAC}^0). ◀

Notice that this schema is not as weak as it may seem since, together with \mathbf{sg} , it is enough to express bounded quantification.

► **Remark 13.** Let $R \subseteq \mathbb{N}^{p+1}$ and h_R be its characteristic function. Then, for all x and \mathbf{y} , it holds that $(\exists z \leq \ell(x))R(z, \mathbf{y}) = \mathbf{sg}(f(x, \mathbf{y}))$, where f is such that $f(0, \mathbf{y}) = h_R(0, \mathbf{y})$ and

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = f(x, \mathbf{y}) + h_R(\ell(x+1), \mathbf{y}).$$

Clearly, $f(x, \mathbf{y})$ is an instance of ℓ -ODE₁. Intuitively, $f(x, \mathbf{y}) \neq 0$ when, for some z smaller than x , $R(z, \mathbf{y})$ is satisfied (i.e. $h_R(z, \mathbf{y}) = 1$): when such instance exists, our bounded search ends with a positive answer. Universally bounded quantification can be expressed in a similar way, considering $\mathbf{cosg}(f(x, \mathbf{y}))$, where f is defined substituting the value of h_R with its co-sign (such that, $\mathbf{cosign}(x) = 1 - \mathbf{sg}(x)$).

The more general schema ℓ -ODE₂ allows multiple left-shifting, such that each “basic operation” corresponds to shifting a given value of a number of digits determined by $\ell(k(\mathbf{y}))$.

► **Definition 14** (ℓ -ODE₂ Schema). *Given $g : \mathbb{N}^p \rightarrow \mathbb{N}$, $h : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ and $k : \mathbb{N}^p \rightarrow \mathbb{N}$, the function $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ is defined by ℓ -ODE₂ from g, h and k if it is the solution of the IVP with initial value $f(0, \mathbf{y}) = g(\mathbf{y})$ and such that:*

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = (2^{\ell(k(\mathbf{y}))} - 1) \times f(x, \mathbf{y}) + h(x, \mathbf{y}),$$

where $h(x, \mathbf{y}) \in \{0, 1\}$, and if, for some x and \mathbf{y} , $h(x, \mathbf{y}) \neq 0$, then $k(\mathbf{y}) \neq 0$.

Since this schema is introduced to characterize \mathbf{FAC}^0 , the constraint imposing $k(\mathbf{y}) \neq 0$, when at some point $h(x, \mathbf{y})$ takes value 1, is really essential. Indeed, as we shall see (Sec. 3.3), if we omit it, ℓ -ODE₂ will be too strong, as able to capture binary counting (which is not in \mathbf{FAC}^0).

Observe that ℓ -ODE₁ is a special case of ℓ -ODE₂, such that $\ell(k(\mathbf{y})) = 1$, and that, also for it, the following closure result holds.

► **Proposition 15.** *If f is defined by ℓ -ODE₂ from \mathbf{FAC}^0 -functions g, k and h , then f is in \mathbf{FAC}^0 .*

Proof Sketch. There are two cases: if $k(\mathbf{y}) \neq 0$, the proof is analogous to that of Prop. 12; if $k(\mathbf{y}) = 0$ (and $h(x, \mathbf{y}) = 0$), for all x and \mathbf{y} , $f(x, \mathbf{y}) = g(\mathbf{y})$, in \mathbf{FAC}^0 by hypothesis. ◀

The Schema ℓ -ODE₃. Let us now consider ℓ -ODE₃, intuitively corresponding to (basic) right-shifting operations.

► **Definition 16** (ℓ -ODE₃ Schema). *Given $g : \mathbb{N}^p \rightarrow \mathbb{N}$, the function $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ is defined by ℓ -ODE₃ from g if it is the solution of the IVP with initial value $f(0, \mathbf{y}) = g(\mathbf{y})$ and such that:*

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = - \left\lceil \frac{f(x, \mathbf{y})}{2} \right\rceil$$

where $\lceil \frac{z}{2} \rceil$ is a shorthand for $z - (z \div 2)$.

► **Proposition 17.** *If f is defined by ℓ -ODE₃ from g in \mathbf{FAC}^0 , then f is in \mathbf{FAC}^0 as well.*

Proof Sketch. The proof is similar to the one for Prop. 12. By Def. 16 and since $(a \div 2^b) \div 2 = a \div 2^{b+1}$, it can be shown by induction that for all x and \mathbf{y} : $f(x, \mathbf{y}) = g(\mathbf{y}) \div 2^{\ell(x)-1}$. This intuitively corresponds to right-shifting $g(\mathbf{y})$ a number of times equal to $\ell(x) - 1$ and can be easily shown computable by a constant-depth circuit. ◀

3.2 An ODE-Characterization of \mathbf{FAC}^0

We now define a new class of functions, crucially relying on the ODE-schemas just introduced:

$$\mathbf{ACDL} = [0, 1, \pi_i^p, \ell, +, -, \div 2, \mathbf{sg}, \circ, \ell\text{-ODE}_2, \ell\text{-ODE}_3].$$

Observe that all its basic functions and (restricted) schemas are natural in the context of differential equations and calculus. In \mathbf{ACDL} , multiplication is, of course, not allowed. Compared to \mathbf{LDDL} , the linear-length ODE schema is substituted by the two schemas ℓ -ODE₂ and ℓ -ODE₃, characterized by a very limited form of “multiplication” and intuitively allowing to capture left and right shifting.

In order to prove that \mathbf{ACDL} captures \mathbf{FAC}^0 we start by providing an *indirect* proof that $\mathbf{FAC}^0 \subseteq \mathbf{ACDL}$. This is established by showing that any basic function and schema defining Clote’s \mathcal{A}_0 (so its arithmetization of log-time bounded RAM) can be simulated in our setting by functions and schemas of \mathbf{ACDL} . Preliminarily, observe that some important operations “come for free” by composition. For instance, the modulo 2 operation is defined as $(x \bmod 2) = x - \lfloor \frac{x}{2} \rfloor - \lfloor \frac{x}{2} \rfloor$, while binary successor functions are expressed in our setting as $s_0(x) = x + x$ and $s_1(x) = s_0(x) + 1$ (being the constant 0 and + basic functions of \mathbf{ACDL}).

The smash function $2^{\ell(x) \times \ell(y)}$. The smash function $x \# y : x, y \mapsto 2^{\ell(x) \times \ell(y)}$ is rewritten as the solution of the IVP defined by the initial value $f(0, y) = 1$ and such that $\frac{\partial f(x, y)}{\partial \ell} = (2^{\ell(y)} - 1) \times f(x)$. This is clearly an instance of ℓ -ODE₂, such that $g(\mathbf{y}) = 1$, $k(\mathbf{y}) = y$ and $h(x, \mathbf{y}) = 0$. Recall that, since $h(x, \mathbf{y}) = 0$, even the limit case of $y = 0$ is properly captured.

The BIT function. Intuitively, the function $\text{BIT}(x, y)$ returns the y^{th} bit in the binary representation of x . In order to capture it, a series of auxiliary functions are needed:

- the *log most significant part function* $\text{msp}(x, y) : x, y \mapsto \lfloor \frac{x}{2^{\ell(x)}} \rfloor$, which can be rewritten via ℓ -ODE₃,
- the *basic conditional function* $\text{if}(x, y, z)$, returning y if $x = 0$ and z otherwise, can be rewritten in our setting by composition, using, in particular, the “shift function” $2^{\ell(x)} \times y$ (defined using ℓ -ODE₁),

- the *special bit function* $\text{bit}(x, y)$, returning 1 when the $\ell(y)^{\text{th}}$ bit of x is 1, can be rewritten in \mathbb{ACDL} due to msp ,
- the *bounded exponentiation function* $\text{bexp}(x, y)$, that, for any $y \leq \ell(x)$, returns 2^y , can be obtained relying on functions in \mathbb{ACDL} , including, in particular, msp , if and $2^{\ell(\cdot)}$ together with ODE-schemas.

Then, using bexp and bit , the desired BIT function can be rewritten in our setting by composition: $\text{BIT}(x, y) = \text{bit}(x, \text{bexp}(x, y) - 1)$.

The CRN Schema. A function f defined by CRN from g , h_0 and h_1 can be simulated in \mathbb{ACDL} via the ℓ -ODE₁ schema. Let us consider the IVP with initial value $F(0, x, \mathbf{y}) = g(\mathbf{y})$ and such that:

$$\frac{\partial F(t, x, \mathbf{y})}{\partial \ell(t)} = F(t, x, \mathbf{y}) + h(t + 1, x, \mathbf{y})$$

where $h(t, x, \mathbf{y}) \in \{0, 1\}$ is, in turn, defined as:

$$\text{if}(\text{bit}(x, 2^{\ell(x)-\ell(t)} - 1), h_0(\text{msp}(2^{\ell(x)-\ell(t)}, x), \mathbf{y}), h_1(\text{msp}(2^{\ell(x)-\ell(t)}, x), \mathbf{y})).$$

The function $F(t, x, \mathbf{y})$ is clearly an instance of ℓ -ODE₁, and $h(t, x, \mathbf{y})$ is defined by composition from functions proved to be in \mathbb{ACDL} . Then, we set $f(x, \mathbf{y}) = F(x, x, \mathbf{y})$.

We now have all the ingredients to prove our main result.

► **Theorem 18.** $\mathbb{ACDL} = \mathbf{FAC}^0$.

Proof. $\mathbb{ACDL} \subseteq \mathbf{FAC}^0$. All basic functions of \mathbb{ACDL} are computable in \mathbf{FAC}^0 . Moreover, the class is closed under composition and, by Prop. 15 and 17, under ℓ -ODE₂ and ℓ -ODE₃. $\mathbf{FAC}^0 \subseteq \mathbb{ACDL}$ since, as we just proved, functions and schemas constituting \mathcal{A}_0 have been rewritten in \mathbb{ACDL} . ◀

A careful analysis of the above definitions shows that ℓ -ODE₂ is actually used only to capture the smash $\#$ function. One obtains a class equivalent to \mathbb{ACDL} by allowing $\#$ and by replacing ℓ -ODE₂ with the simpler ℓ -ODE₁ schema.

► **Corollary 19.** $\mathbf{FAC}^0 = [0, 1, \pi_i^p, \ell, +, -, \div 2, \text{sg}, \#; \circ, \ell\text{-ODE}_1, \ell\text{-ODE}_3]$

3.3 An ODE-Characterization of \mathbf{FTC}^0

As a byproduct, an ODE-characterization for \mathbf{FTC}^0 is easily obtained, this time passing through \mathcal{TC}_0 [14]. We consider an extension of \mathbb{ACDL} endowed with the basic function \times :

$$\mathbb{TCDL} = [0, 1, \pi_i^p, \ell, +, -, \div 2, \times, \text{sg}; \circ, \ell\text{-ODE}_2, \ell\text{-ODE}_3].$$

► **Proposition 20.** *Let f be defined by ℓ -ODE₂ from functions in \mathbf{FTC}^0 . Then, f is in \mathbf{FTC}^0 .*

Proof Sketch. The proof is similar to that of Prop. 15. The main difference concerns the computation of level 0, i.e. that of the initial values $g(\mathbf{y})$ and $h(x, \mathbf{y})$, which are now expressions possibly including \times . This does not affect the overall structure of the circuit. ◀

► **Proposition 21.** *Let f be defined by ℓ -ODE₃ from functions in \mathbf{FTC}^0 . Then, f is in \mathbf{FTC}^0 .*

10:10 A New Characterization of \mathbf{FAC}^0 via Discrete ODEs

Proof Sketch. Straightforward generalization of Prop. 17, with the same provisos of Prop. 20. \blacktriangleleft

Then, the desired characterization easily follows from Propositions 20 and 21 and from [14], rewritten in our ODE-setting (see Sec. 3.2).

► **Theorem 22.** $\mathbf{TCDDL} = \mathbf{FTC}^0$.

An alternative characterization of \mathbf{FTC}^0 is obtained by considering the following class:

$$\mathbf{TCDDL}^* = [0, 1, \pi_i^p, \ell, +, -, \div 2, \mathbf{sg}; \circ, \ell\text{-ODE}_2^*, \ell\text{-ODE}_3].$$

where \mathbf{TCDDL}^* does not include \times as a basic function, but allows the generalized schema $\ell\text{-ODE}_2^*$, with no constraint over the function k :

► **Definition 23** ($\ell\text{-ODE}_2^*$ Schema). *Let $g : \mathbb{N}^p \rightarrow \mathbb{N}$, $h : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ and $k : \mathbb{N}^p \rightarrow \mathbb{N}$, where h takes values in $\{0, 1\}$. Then, the function $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ is defined by $\ell\text{-ODE}_2^*$ from g, h and k when it is the solution of the IVP with initial value $f(0, \mathbf{y}) = g(\mathbf{y})$ and such that:*

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = (2^{\ell(k(\mathbf{y}))} - 1) \times f(x, \mathbf{y}) + h(x, \mathbf{y}).$$

► **Example 24** (bcount). Observe that if $k(\mathbf{y}) = 0$, then $\ell\text{-ODE}_2^*$ is enough to express the binary counting function $\mathbf{bcount}(x)$, that outputs the sum of the bits of x . Indeed, $\mathbf{bcount}(x) = f(x, x)$ where f is the solution of the IVP with initial value $f(0, \mathbf{y}) = \mathbf{bit}(0, y)$ and such that:

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = \mathbf{bit}(x, \mathbf{y}).$$

Since such function is not in \mathbf{FAC}^0 (see [19]), this also illustrates that $\ell\text{-ODE}_2^*$ is really more expressive than $\ell\text{-ODE}_2$.

It is easy to see that $\ell\text{-ODE}_2^*$ is enough to capture majority (and to “simulate” multiplication, see [30]), which is the essential step to show that $\mathbf{TCDDL}^* = \mathbf{FTC}^0$.

This observation, together with the fact that what really makes $\ell\text{-ODE}_2^*$ more expressive than $\ell\text{-ODE}_2$ is its behavior for $k(\mathbf{y}) = 0$, leads us to an alternative characterization for \mathbf{FTC}^0 , in line with the one of Corollary 19. Let’s consider the following schema.

► **Definition 25** ($\ell\text{-ODE}_1^*$ Schema). *Let $g : \mathbb{N}^p \rightarrow \mathbb{N}$ and $h, k : \mathbb{N}^{p+1} \rightarrow \{0, 1\}$. Then the function $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ is defined by $\ell\text{-ODE}_1^*$ from g, h and k , when it is the solution of the IVP with initial value $f(0, \mathbf{y}) = g(\mathbf{y})$ and such that:*

$$\frac{\partial f(x, \mathbf{y})}{\partial \ell} = k(x, \mathbf{y}) \times f(x, \mathbf{y}) + h(x, \mathbf{y}).$$

By straightforward inspection and Example 24, it is easily seen that \mathbf{FTC}^0 is again captured by adding the basic function $\#$ and by replacing $\ell\text{-ODE}_2^*$ with the simpler schema $\ell\text{-ODE}_1^*$.

► **Corollary 26.** $\mathbf{FTC}^0 = \mathbf{TCDDL}^* = [0, 1, \pi_i^p, \ell, +, -, \div 2, \mathbf{sg}, \#; \circ, \ell\text{-ODE}_1^*, \ell\text{-ODE}_3]$.

3.4 Alternative Direct Proofs

In this section, we introduce alternative classes \mathbf{ACDL}_C and \mathbf{TCDDL}_C , (resp.) extending \mathbf{ACDL} and \mathbf{TCDDL} with new basic functions that arithmetize the circuit families of polynomial size and constant depth $(C_n)_{n \geq 0}$ used for computation. In this non-uniform context, we prove both $\mathbf{FAC}^0 \subseteq \mathbf{ACDL}_C$ (Sec. 3.4.1) and $\mathbf{FTC}^0 \subseteq \mathbf{TCDDL}_C$ (Sec. 3.4.2) *directly*, i.e. without any references to results in [11, 14], .

3.4.1 Direct Completeness for $\text{ACDL}_{\mathcal{C}}$

Let $\mathcal{C} = (C_n)_{n \geq 0}$ be a class of circuits of polynomial size n^k , for some $k \in \mathbb{N}$, and constant depth d . We assume that each circuit C_n is in a special normal form, such that it strictly alternates between \wedge and \vee (and edges are only between gates of consecutive layers): input gates are all at level 0, negation gates are all at level 1, even levels are all \wedge gates, odd levels (other than 1) are all \vee gates, and the depth d is even (so output gates are \wedge gates).

In this context we keep the basic functions of ACDL , but add a set $\mathbf{circ}_{\mathcal{C}} = \{C, L_0^{in}, L_0^-, L_e\}$ of characteristic functions associated to the following predicates. The predicate $C \subseteq \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ describes the underlying graph of the circuit: for any integers x, α, β , $(x, \alpha, \beta) \in C$ when, in $C_{\ell(x)}$, the $\alpha^{th} \leq \ell(x)^k$ gate of some level is a predecessor of the $\beta^{th} \leq \ell(x)^k$ gate of the next level (thus, in this encoding, α and β are exponentially smaller than x). The relations $L_0^{in}, L_0^-, L_e \subseteq \mathbb{N} \times \mathbb{N}$, for $e \in \{1, \dots, d\}$, describe the level of gates (and, implicitly, their type): L_0^{in} refers to input gates, L_0^- to negation gates, and L_e to \wedge and \vee gates, depending on e being odd or even. Since we aim at defining functions over integers, we assume that input gates are numbered from $n-1$ to 0 and output gates from $m-1$ to 0, with $m \leq \ell(x)^k$. By considering the functions corresponding to the given relations, we obtain the desired ODE-style family of classes (parameterized by \mathcal{C}):

$$\text{ACDL}_{\mathcal{C}} = [0, 1, \pi_i^p, \ell, +, -, \div 2, \mathbf{sg}, \mathbf{circ}_{\mathcal{C}}; \circ, \ell\text{-ODE}_2, \ell\text{-ODE}_3].$$

► **Remark 27.** If the family \mathcal{C} is **Dlogtime**-uniform, then the functions in $\mathbf{circ}_{\mathcal{C}}$ are computable in \mathcal{A}_0 . Consequently, in this case, it holds that $\text{ACDL}_{\mathcal{C}} = \text{ACDL}$.

In this non-necessarily uniform context, a completeness proof still holds.

► **Proposition 28.** *If a function $f : \mathbb{N} \rightarrow \mathbb{N}$ is computable by a family $\mathcal{C} = (C_n)_{n \geq 0}$ of polynomial size and constant-depth circuits, then it is in $\text{ACDL}_{\mathcal{C}}$.*

To prove it we need the following lemma (which is easily established relying on Remark 13).

► **Lemma 29.** *Let g and h be functions computable in $\text{ACDL}_{\mathcal{C}}$ and $k \in \mathbb{N}$. Then, the function $\min_{i \leq \ell(x_1)^k} \{g(i, \mathbf{x}) : h(i, \mathbf{x}) \triangleright j\}$, for $\triangleright \in \{<, \leq, >, \geq, =\}$ and $j \in \{0, 1\}$, is in $\text{ACDL}_{\mathcal{C}}$.*

Proof of Prop. 28. Let $\text{Eval}(t, x)$ be a function that returns the value of the t^{th} output gate of the circuit $C_{\ell(x)}$ of input x when $t \leq m-1$ (and $\text{Eval}(t, x) = 0$ for $t > m-1$). Then, the following expression defines a function f such that $f(2^{\ell(x)^k}, x)$ outputs the value of the computation of C_n (for $n = \ell(x)$) on input x :

$$\frac{\partial f(y, x)}{\partial \ell(y)} = f(y, x) + \text{Eval}(\ell(x) - \ell(y) - 1, x)$$

with $f(0, x) = 0$. Intuitively, the function above computes the successive suffixes of the output word, starting from the bits of bigger weights. Remarkably, this is an instance of the $\ell\text{-ODE}_1$ schema (indeed, $\text{Eval}(y, x) \in \{0, 1\}$). So, the given f can be rewritten in $\text{ACDL}_{\mathcal{C}}$.

It remains to describe how the function $\text{Eval}(t, x)$ is computed. Again, we assume that C_n has depth d , is in the normal form described above, and d is even. Concretely, we start by defining a special (bounded) minimum operator function such that, given $k \in \mathbb{N}$ and two functions g and h , with $h(t, \mathbf{x}) \in \{0, 1\}$ for $t \in \mathbb{N}$ and $\mathbf{x} = x_1, \dots, x_h$,

$$\min_{i \leq \ell(x_1)^k} \{g(i, \mathbf{x}) : h(i, \mathbf{x}) \triangleright 0\},$$

with $\triangleright \in \{<, \leq, >, \geq, =\}$ and $j \in \{0, 1\}$. Intuitively, given $i \in \{0, \dots, \ell(x_1)^k\}$, this function computes the minimum of the values of $g(i, \mathbf{x})$, for i and \mathbf{x} such that $h(i, \mathbf{x}) \triangleright j$.

The inductive definition of Eval relies on those of the $d+1$ functions $\text{Eval}_0, \dots, \text{Eval}_d$, with $\text{Eval}_d = \text{Eval}$:

10:12 A New Characterization of \mathbf{FAC}^0 via Discrete ODEs

- $\text{Eval}_0(t, x)$ is equal to $\text{BIT}(t, x)$ if $L_0^{in}(t, x)$ holds and to $1 - \text{BIT}(t, x)$ if $L_0^-(t, x)$ does. For t not corresponding to gate index, $\text{Eval}_0(t, x)$ is set to an arbitrary value, say 0. Recall that, since BIT can be rewritten in \mathbf{ACDDL} (Sec. 3.2), Eval_0 is in $\mathbf{ACDDL}_{\mathcal{C}}$ as well.
- $\text{Eval}_{2e}(y, x)$ is equal to $\min_{i \leq \ell(x)^k} \{\text{Eval}_{2e-1}(i, x) : C(x, i, t) = 1\}$, for $L_{2e}(t, x)$ (i.e., if t is the index of a gate at this level). The evaluation for the i^{th} gate of the level $2e$ (a \wedge -gate) is the minimum of the evaluations of its predecessor gates of level $2e - 1$. As seen, \min is in \mathbf{ACDDL} (Lemma 29), so Eval_{2e} can also be rewritten in this class.
- Similarly, $\text{Eval}_{2e+1}(y, x)$ is the $1 - \min_{i \leq \ell(x)^k} \{1 - \text{Eval}_{2e}(i, x) : C(x, i, t) = 1\}$. The evaluation for the t^{th} gate of level $2e + 1$ (a \vee -gate) is the maximum among evaluations of its predecessor gates of level $2e$. As for Eval_{2e} , Eval_{2e+1} can be rewritten in \mathbf{ACDDL} . ◀

► **Remark 30.** The following converse to Proposition 28 also holds (by inspecting the proofs of Proposition 15 and 17): If a function $f : \mathbb{N} \rightarrow \mathbb{N}$ is in $\mathbf{ACDDL}_{\mathcal{C}}$ for some family \mathcal{C} of polynomial size and constant-depth circuits, then there exists a family \mathcal{C}' of polynomial size and constant-depth circuits that computes it.

3.4.2 Direct Completeness for \mathbf{FTC}^0

Let us now consider the similar, direct characterization for \mathbf{FTC}^0 . Suppose that C_n strictly alternates between \wedge, \vee and MAJ gates, and that input gates and their negation are all at level 0, \vee gates are at levels $3e + 1$, \wedge gates at levels $3e + 2$, and MAJ gates at levels $3e$. Accordingly, in this case L_e describes the level of a gate of a type not limited to \wedge, \vee , but including MAJ. Then, the desired family of classes is defined as:

$$\mathbf{TCDDL}_{\mathcal{C}} = [0, 1, \pi_i^p, \ell, +, -, \div 2, \mathbf{sg}, \mathbf{circ}_{\mathcal{C}}; \circ, \ell\text{-ODE}_2^*, \ell\text{-ODE}_3]$$

where, with a slight abuse of notation, we use $\mathbf{circ}_{\mathcal{C}}$ to denote the function corresponding to a (set of) relation(s), this time including the *extended* L_e . The proof that $\mathbf{FTC}^0 \subseteq \mathbf{TCDDL}_{\mathcal{C}}$ is similar to the one from Section 3.4.1.

► **Proposition 31.** *If a function $f : \mathbb{N} \rightarrow \mathbb{N}$ is computable by a family $\mathcal{C} = (C_n)_{n \geq 0}$ of polynomial-size and constant-depth circuits including MAJ gates, then it is in $\mathbf{TCDDL}_{\mathcal{C}}$.*

Proof Sketch. The functions f and Eval are globally defined as before. Modifications only affect the definition of Eval_d and, in particular, the inductive levels corresponding to MAJ. Specifically, it is obtained as follows:

- for a given function h and integer k , $\mathbf{bcount}_h(t, x) = \sum_{i \leq \ell(x)^k} h(i, t, x)$. Notice that this function is in $\mathbf{TCDDL}_{\mathcal{C}}$, since it can be rewritten as an instance of $\ell\text{-ODE}_2^*$, and h is in \mathbf{TCDDL}^* (due to Lemma 29),
- for any i such that $L_{3e-1}(i)$ and $3e - 1 < d$: $v_{3e-1}^0(i, t, x) = \mathbf{sg}(C(x, i, t))$ and $v_{3e-1}^1(i, t, x) = \text{if}(C(x, i, t), \text{Eval}_{3e-1}(i, x), 0)$. The value of $v_{3e-1}^0(i, t, x)$ is 1 when i is a predecessor of gate t , and $v_{3e-1}^1(i, t, x)$ is 1 when, in addition, the value of gate i , on input x , is 1,
- for t such that $L_{3e}(t)$, $\text{Eval}_{3e}(t, x)$ is defined as $\mathbf{sg}(\mathbf{bcount}_{v_{3e-1}^0}(t, x) - 2 \times \mathbf{bcount}_{v_{3e-1}^1}(t, x))$. The function outputs 1 when more than half of the inputs of gate t are 1. ◀

4 Conclusion

We have presented new characterizations for \mathbf{FAC}^0 and \mathbf{FTC}^0 through the prism of discrete differential equations. Although the use of classical arithmetical functions is intrinsically limited by the low computational power of these classes, the ODEs used are surprisingly

natural restrictions of linear ODEs. More generally, this work is intended as the first step of a project aiming at characterizing other relevant classes, starting with FAC^k and FNC^k . Another challenging direction of future research would be to develop logical and proof-theoretical counterparts to ODE-style algebras, e.g. by defining *natural* rule systems (oriented by the ODE design) to syntactically characterize the corresponding classes.

References

- 1 B. Allen. Arithmetizing uniform NC. *Ann. Pure Appl. Logic*, 53:1–50, 1991.
- 2 T. Arai. A bounded arithmetic AID for Frege systems. *Ann. Pure Appl. Logic*, 103:155–199, 2000.
- 3 S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2007.
- 4 D.A.M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1 . *J. of Comput. and Syst. Sc.*, 41:274–306, 1990.
- 5 S. Bellantoni and S. Cook. A new recursion-theoretic characterization of poly-time functions. *Comput. Complex.*, 2:97–110, 1992.
- 6 J.H. Bennett. *On Spectra*. PhD thesis, Princeton University, 1962.
- 7 M. Blanc and O. Bournez. A characterisation of polynomial time computable functions from the integers to the reals using discrete ordinary differential equations. In *Proc. MCU*, pages 58–74, 2022.
- 8 M. Blanc and O. Bournez. A characterisation of functions computable in polynomial time and space over the reals with discrete ordinary differential equations: simulation of Turing machines with analytic discrete ODEs. In *Proc. MFCS*, pages 21:1–21:15, 2023.
- 9 G. Bonfante, R. Kahle, J.-Y. Marion, and I. Oitavem. Two function algebras defining functions in NC^k boolean circuits. *Inf. and Comput.*, 2016.
- 10 O. Bournez and A. Durand. A characterization of functions over the integers computable in polynomial time using discrete differential equations. *Comput. Complex.*, 32(7), 2023.
- 11 P.G. Clote. A sequential characterization of the parallel complexity class NC. Technical report, Boston College, 1988.
- 12 P.G. Clote. Sequential, machine-independent characterizations of the parallel complexity classes AlogTIME , AC^k , NC^k and NC. In Buss S.R. and Scott P.J., editors, *Feasible Mathematics: A Mathematical Sciences Institute Workshop, Ithaca, New York, June 1989*, Progress in Computer Science and Applied Logic, pages 49–69. Birkhäuser, Boston, MA, 1990.
- 13 P.G. Clote and G. Takeuti. Bounded arithmetic for NC, AlogTIME , L and NL. *Ann. Pure and Appl. Logic*, 56:73–117, 1992.
- 14 P.G. Clote and G. Takeuti. First order bounded arithmetic and small complexity classes. In P.G. Clote and J.B. Remmel, editors, *Feasible Mathematics II*, pages 154–218. Birkhäuser, 1995.
- 15 A. Cobham. The intrinsic computational difficulty of functions. In *Logic, Methodology and Philosophy of Science: Proc. 1964 International Congress*, pages 24–30. Amsterdam: North-Holland, 1965.
- 16 K.J. Compton and C. Laflamme. An algebra and a logic for NC^1 . *Inf. Comput.*, 87(1/2):240–262, 1990.
- 17 S. Cook and T. Morioka. Quantified propositional calculus and a second-order theory for NC^1 . *Arch. Math. Logic*, 44:711–749, 2005.
- 18 S. Cook and P. Nguyen. Theories for TC^0 and other small circuit classes. *Log. Meth. Comput. Sci.*, 2:1–40, 2006.
- 19 M.L. Furst, J.B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. In *Proc. 22nd Annual Symposium on Foundations of Computer Science*, pages 260–270, 1981.
- 20 A. Grzegorzcyk. Some classes of recursive functions. *Rozprawy Matematyczne*, 4, 1953.
- 21 Y. Gurevich and H. Lewis. A logic for constant-depth circuit. *Inf. Control*, 61:65–74, 1984.

- 22 L. Hella, J. Kontinen, and K. Luosto. Regular representations of uniform \mathbf{TC}^0 . [arXiv:2309.06926](#).
- 23 N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16:760–778, 1987.
- 24 J. Johannsen. A bounded arithmetic theory for constant depth threshold circuits. In *GÖDEL '96*, volume 6 of *Springer Lecture Notes in Logic*, pages 224–234. Hájek, P., 1996.
- 25 D. Leivant. Ramified recurrence and computational complexity I: Word recurrence and poly-time. In P.G. Clote and J.B. Remmel, editors, *Feasible Mathematics II*, Progress in Computer Science and Applied Logic, pages 320–343. Birkhäuser, 1994.
- 26 D. Leivant and Y.-Y. Marion. Lambda calculus characterizations of poly-time. *Fundam. Inform.*, 19(1,2):167–184, 1993.
- 27 J.C. Lind. *Computing in Logarithmic Space*. PhD thesis, Massachusetts Institute of Technology, 1974.
- 28 S. Lindell. A purely logical characterization of circuit uniformity. In *7th Structure in Complexity Theory Conf.*, pages 185–192, 1992.
- 29 R.W. Ritchie. Classes of predicability computable functions. *Trans. Am. Math. Soc.*, 106:139–173, 1963.
- 30 H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer, 1999.

A Proofs from Section 3

A.1 The Schemas $\ell\text{-ODE}_1$ and $\ell\text{-ODE}_2$

Proof of Proposition 12. By Definition 10, for all x and \mathbf{y} :

$$\begin{aligned} f(x, \mathbf{y}) &= \sum_{u=-1}^{\ell(x)-1} \left(\prod_{t=u+1}^{\ell(x)-1} 2 \right) \times h(\alpha(u), \mathbf{y}) \\ &= \sum_{u=-1}^{\ell(x)-1} 2^{\ell(x)-u-1} \times h(\alpha(u), \mathbf{y}) \end{aligned}$$

with the convention that $\alpha(u) = 2^u - 1$, $\prod_x^{x-1} \kappa(x) = 1$ and $h(\alpha(-1), \mathbf{y}) = f(0, \mathbf{y})$. Notice that the given multiplication is always by a power of 2 decreasing for each increasing value of u , which basically corresponds to left-shifting (which can be computed in \mathbf{FAC}^0). Hence, since by Definition 10, $h(x, \mathbf{y}) \in \{0, 1\}$, the outermost sum amounts to a concatenation (which again can be computed in \mathbf{FAC}^0).

Concretely, for any inputs x and \mathbf{y} , the desired polynomial-sized and constant-depth circuit to compute $f(x, \mathbf{y})$ is defined as follows:

- In parallel compute the values of $g(\mathbf{y})$ and of each $h(\alpha(u), \mathbf{y})$, for any $u \in \{0, \dots, \ell(x) - 1\}$. For hypothesis, g and h are computable in \mathbf{FAC}^0 , and, since there are $\ell(x) + 1$ initial values to be computed, the whole desired computation can be done in polynomial size and constant depth.
- In one step, (left-)shift the value of $h(\alpha(-1), \mathbf{y}) = g(\mathbf{y})$ by padding $\ell(x)$ zeros on the right and, for $u \geq 0$, (left-)shift each value $h(\alpha(u), \mathbf{y})$ by padding on the right $\ell(x) - u - 1$ zeros (this corresponds to multiply by $2^{\ell(x)-u-1}$) and padding on the left $u + \ell(g(\mathbf{y}))$ zeros.
- Compute, bit-by-bit, the disjunction of all values computed above. Clearly, this is done in constant depth. ◀

Proof of Proposition 15. There are two main cases to be taken into account. If $k(\mathbf{y}) \neq 0$, the proof is similar to that of Proposition 12. Indeed, for all x and \mathbf{y} :

$$\begin{aligned} f(x, \mathbf{y}) &= \sum_{u=-1}^{\ell(x)-1} \left(\prod_{t=u+1}^{\ell(x)-1} 2^{\ell(k(\mathbf{y}))} \right) \times h(\alpha(u), \mathbf{y}) \\ &= \sum_{u=-1}^{\ell(x)-1} 2^{\ell(k(\mathbf{y})) \times (\ell(x)-u-1)} \times h(\alpha(u), \mathbf{y}) \end{aligned}$$

with the convention that $\alpha(u) = 2^u - 1$, $\prod_x^{x-1} \kappa(x) = 1$ and $h(\alpha(-1), \mathbf{y}) = f(0, \mathbf{y})$. Observe that here multiplication corresponds to a left-shifting where “basic shifting” corresponds to a left movement of $\ell(k(\mathbf{y}))$ digits. As for the basic case, it can be easily shown that this operation can be implemented by a constant-depth circuit. Then, analogously to Proposition 12, the outermost iterated sum amounts to concatenation (as, by construction, $h(\alpha(u), \mathbf{y}) \in \{0, 1\}$).

Concretely, we can construct a constant-depth circuit generalizing the procedure defined for the special case of ℓ -ODE₁:

- In parallel, compute the values of $g(\mathbf{y})$ and $h(\alpha(u), \mathbf{y})$, for each $u \in \{0, \dots, \ell(x) - 1\}$. This can be done in constant depth by hypothesis.
- In one step, shift the value of $g(\mathbf{y})$ by padding $\ell(k(\mathbf{y})) \times \ell(x)$ zeros on the right, and, for $u \geq 0$, shift all values $h(\alpha(u), \mathbf{y})$ by padding on the right $\ell(k(\mathbf{y})) \times (\ell(x) - u - 1)$ zeros (i.e. multiplying by $2^{\ell(k(\mathbf{y})) \times (\ell(x) - u - 1)}$) and by padding on the left $\ell(g(\mathbf{y})) + \ell(k(\mathbf{y})) \times (u + 1) - 1$ zeros.
- Compute, bit-by-bit, the disjunction of all the values above.

In the special case of $k(\mathbf{y}) = 0$ and $h(x, \mathbf{y}) = 0$, it holds that for all x and \mathbf{y} , $f(x, \mathbf{y}) = g(\mathbf{y})$. This is clearly computable in \mathbf{FAC}^0 , as corresponding to compute $g(\mathbf{y})$, which is computable in constant depth by hypothesis. ◀

A.2 The Schema ℓ -ODE₃

For $x > 0$, the equation characterizing Definition 16 can be re-written as:

$$f(x, \mathbf{y}) = f(x - 1, \mathbf{y}) - \Delta\ell(x - 1) \times \left\lceil \frac{f(x - 1, \mathbf{y})}{2} \right\rceil,$$

where, as seen, $\Delta\ell(x - 1) = \ell(x) - \ell(x - 1)$. Observe that also in this case we are using \times with a slight abuse of notation: indeed, we are dealing with “bit multiplication” and multiplying a number by 0 or 1 can be easily done in \mathbf{FAC}^0 (and easily rewritten in our setting using the basic conditional function if , which, in turn, can be defined in \mathbf{ACDL} , see Sec. 3.3). In other words,

$$\begin{aligned} f(x, \mathbf{y}) &= \begin{cases} f(x - 1, \mathbf{y}) & \text{if } \ell(x) = \ell(x - 1) \\ f(x - 1, \mathbf{y}) - \left\lceil \frac{f(x - 1, \mathbf{y})}{2} \right\rceil & \text{otherwise} \end{cases} \\ &= \begin{cases} f(x - 1, \mathbf{y}) & \text{if } \ell(x) = \ell(x - 1) \\ \left\lceil \frac{f(x - 1, \mathbf{y})}{2} \right\rceil & \text{otherwise} \end{cases} \\ &= \begin{cases} f(x - 1, \mathbf{y}) & \text{if } \ell(x) = \ell(x - 1) \\ f(x - 1, \mathbf{y}) \div 2 & \text{otherwise.} \end{cases} \end{aligned}$$

10:16 A New Characterization of FAC^0 via Discrete ODEs

A bit more formally,

$$f(x, \mathbf{y}) = \left\lfloor \frac{f(\beta(\ell(x) - 1), \mathbf{y})}{2} \right\rfloor = \left\lfloor \frac{f(2^{\ell(x)-1} - 1, \mathbf{y})}{2} \right\rfloor = f(2^{\ell(x)-1} - 1, \mathbf{y}) \div 2$$

where $\beta(\ell(z)) = 2^{\ell(z)} - 1$ is the greatest integer the length of which is $\ell(z)$, i.e. here, $2^{\ell(x)-1} - 1$ is the greatest integer the length of which is $\ell(x) - 1$. Hence, starting with $x > 0$, there are $\ell(x) - 1$ jumps of values.

Proof of Proposition 17. By Definition 16 (as clarified by the remarks above), and, since $(a \div 2^b) \div 2 = a \div 2^{b+1}$, it is easily shown by induction that, for all x and \mathbf{y} :

$$\begin{aligned} f(x, \mathbf{y}) &= \left\lfloor \prod_{u=1}^{\ell(x)-1} \frac{g(\mathbf{y})}{2} \right\rfloor \\ &= \left\lfloor \frac{g(\mathbf{y})}{2^{\ell(x)-1}} \right\rfloor \\ &= g(\mathbf{y}) \div 2^{\ell(x)-1}. \end{aligned}$$

This corresponds to right-shifting $g(\mathbf{y})$ a number of times equal to $\ell(x) - 1$, which can be easily implemented by a constant-depth circuit. \blacktriangleleft

A.3 Rewriting the Function BIT in ACDL

First, the *log most significant part function* $\text{msp}(x, y) : x, y \mapsto \left\lfloor \frac{y}{2^{\ell(x)}} \right\rfloor$ can be rewritten as the solution of the IVP:

$$\begin{aligned} f(0, y) &= y \\ \frac{\partial f(x, y)}{\partial \ell} &= - \left\lfloor \frac{f(x, y)}{2} \right\rfloor \end{aligned}$$

which is clearly an instance of $\ell\text{-ODE}_3$.

Second, we introduce the basic conditional function:

$$\text{if}(x, y, z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{otherwise.} \end{cases}$$

Notice that this function is also crucial to rewrite the CRN schema. As seen, the “shift function” $2^{\ell(x)} \times y$ can be easily rewritten via $\ell\text{-ODE}_1$. Thus, $\text{if}(x, y, z)$ is simulated in our setting by composition from shift, addition and subtraction:

$$\text{if}(x, y, z) = (2^{\ell(1-\text{sg}(x))} \times y - y) + (2^{\ell(\text{sg}(x))} \times z - z).$$

Indeed, as desired, if $x = 0$, then $\text{sg}(x) = 0$ and $\ell(\text{sg}(x)) = 0$, so that $\text{if}(0, y, z) = (2^{\ell(1)} \times y - y) + (2^{\ell(0)} \times z - z) = y$; similarly, for $x \neq 0$, $\text{if}(0, y, z) = (2^{\ell(0)} \times y - y) + (2^{\ell(1)} \times z - z) = z$. Generalizing this definition we can capture a more general conditional function below:

$$\text{cond}(x, v, y, z) = \begin{cases} y & \text{if } x < v \\ z & \text{otherwise} \end{cases}$$

Then, we consider the special bit function $\text{bit}(x, y)$ returning 1 when the $\ell(y)^{\text{th}}$ bit of x is 1. This can be rewritten in ACDL due to msp :

$$\text{bit}(x, y) = \text{msp}(y, x) - 2 \times \text{msp}(2y + 1, x).$$

Finally, we introduce the function $\text{bexp}(x, y)$, that, for any $y \leq \ell(x)$, returns 2^i . We start by defining $f_{aux}(t, x, i)$ by the ℓ -ODE₁ schema below:

$$\begin{aligned} f_{aux}(0, x, i) &= \text{if}(i, 1, 0) \\ \frac{\partial f_{aux}(t, x, i)}{\partial \ell(t)} &= f_{aux}(t, x, i) + h_{aux}(t, i) \end{aligned}$$

where

$$h_{aux}(t, i) = \text{if}(\ell(t) - i, 1, 0).$$

Observe that, as seen, it can be rewritten in \mathbb{ACDL} (while ℓ and subtraction are basic functions). Then, for $i \leq \ell(x)$, we obtain $f_{aux}(x, x, i) = 2^{\ell(x)-i}$. The function bexp is then defined as follows:

$$\text{bexp}(x, i) = \text{msp}(f_{aux}(x, x, i) - 1, 2^{\ell(x)}) = \left\lfloor \frac{2^{\ell(x)}}{2^{\ell(x)-i}} \right\rfloor = 2^i,$$

as the length of $f_{aux}(x, x, i) - 1$ is $\ell(x) - i$. Clearly, the function bexp is also in \mathbb{ACDL} as all the functions involved in its definitions (namely, msp , f_{aux} and $2^{\ell(\cdot)}$) are in \mathbb{ACDL} .

We conclude by showing that, due to bexp and bit , the desired BIT function can be rewritten in our setting by composition:

$$\text{BIT}(x, y) = \text{bit}(x, \text{bexp}(x, y) - 1).$$

Observe that alternative proofs are possible, but the one proposed here, and based on the introduction of bexp , not only has the advantage of being straightforward, but also avoids the unnatural use of function most significant part function, MSP .

A.4 On the ODE-Characterization of FTC^0

Proof of Proposition 20. The proof is similar to that of Proposition 15. The main difference concerns $g(\mathbf{y})$ and $h(x, \mathbf{y})$, which are now expressions possibly including \times .

As seen, for all x and \mathbf{y} :

$$f(x, \mathbf{y}) = \sum_{u=-1}^{\ell(x)-1} \left(\prod_{t=u+1}^{\ell(x)-1} 2^{\ell(k(\mathbf{y}))} \right) \times h(\alpha(u), \mathbf{y})$$

with the convention that $\alpha(u) = 2^u - 1$, $\prod_x^{x-1} \kappa(x) = 1$ and $h(\alpha(-1), \mathbf{y}) = f(0, \mathbf{y})$. Intuitively, the (constant-depth) circuit we are going to construct is equivalent to that of Proposition 15, but the values to be initially computed in parallel are obtained even via \times . Yet, the introduction of multiplication does not affect the overall structure of the circuit, as $h(\alpha(u), \mathbf{y}) \in \{0, 1\}$, so that the final sum again corresponds to a simple bit-concatenation (without carries).

More precisely, the desired constant-depth circuit is defined as follows:

- In parallel, compute the values of $g(\mathbf{y})$ and, for any $u = 0, \dots, \ell(x) - 1$, of $h(\alpha(u), \mathbf{y})$. Observe that this can be done in FTC^0 , but possibly not in FAC^0 , as now these arithmetic expressions may include \times .
- The value of $g(\mathbf{y})$ is shifted by padding $2^{\ell(k(\mathbf{y})) \times \ell(x)}$ zeros on the right and, for $u \geq 0$, all values $h(\alpha(u), \mathbf{y})$ are shifted by padding on the right $\ell(k(\mathbf{y}))$ zeros $\ell(x) - u - 1$ times, and by padding on the left $\ell(k(\mathbf{y}))$ zeros $\ell(g(\mathbf{y})) + u$ times.
- the disjunction of the above values is computed bit by bit. ◀

10:18 A New Characterization of \mathbf{FAC}^0 via Discrete ODEs

Proof of Proposition 22. $\mathbf{TCDL} \subseteq \mathbf{FTC}^0$. All basic functions are computable in \mathbf{FTC}^0 , and the class is closed under composition and, by Propositions 20 and 21, under ℓ -ODE₂ and ℓ -ODE₃.

$\mathbf{FTC}^0 \subseteq \mathbf{TCDL}$. By mimicking the arithmetization proof provided in [14] (see Sec. 2.2), as all the functions defining \mathcal{TC}_0 can be rewritten in \mathbf{TCDL} (this time including \times , which is basic in \mathbf{TCDL}). ◀