Recursion schemes, discrete differential equations and characterization of polynomial time computation

Olivier Bournez, Arnaud Durand

Novembre 2020, ANR Difference

Introduction

- Ordinary Differential Equations (ODE) is a natural way to express properties of many systems in applied science
- Very active field of maths, abundant litterature
- We are interested here in its discrete counterpart : discrete ODE
- Built on a notion of derivative, *finite differences*, widely studied in numerical optimization and combinatorial analysis

Introduction

- Study the expressive and computational power of discrete ODE
- Appears
  - to be a convenient tool for algorithm design
  - to elegantly capture complexity notions
- Believe it help to better understand computation for both the discrete and the continuous settings

Plan

- Introduction to discrete ODE
- A short survey on recursion scheme for complexity
- ODE and complexity classes : characterizing polynomial time computation

Plan

- Introduction to discrete ODE
- A short survey on recursion scheme for complexity
- ODE and complexity classes : characterizing polynomial time computation

Discrete derivative

Definition

Let $f : \mathbb{N} \to \mathbb{Z}$, the discrete derivative (a.k.a finite difference) is defined as:

$$\Delta\mathbf{f}(x) = \mathbf{f}(x+1) - \mathbf{f}(x).$$

When $f : \mathbb{N}^p \to Z^q$, set:

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

.

Sometimes use $\mathbf{f}'(x)$ instead of $\Delta(\mathbf{f}(x))$

Discrete integral

Definition (Discrete Integral)

we write $\int_a^b \mathbf{f}(x)\delta x$ as a synonym for

$$\int_a^b \mathbf{f}(x)\delta x = \sum_{x=a}^{x=b-1} \mathbf{f}(x)$$

with the conventions: $\int_a^a \mathbf{f}(x)\delta x = 0$ and $\int_a^b \mathbf{f}(x)\delta x = -\int_b^a \mathbf{f}(x)\delta x$ when $a > b$.

It follows easily by the telescope formula that:

Theorem (Fundamental Theorem of Finite Calculus)

Let $\mathbf{F}(x)$ be some function. Then, $\int_a^b \mathbf{F}'(x)\delta x = \mathbf{F}(b) - \mathbf{F}(a)$.

Discrete integral and basics of integration

Not surprisingly, basic notions from the continous setting adapts easily:

- derivation of a composition , integration by parts, etc
- Example of the Product rule: $(\mathbf{f}(x) \cdot \mathbf{g}(x))' = \mathbf{f}(x+1) \cdot \mathbf{g}'(x) + \mathbf{f}(x)' \cdot \mathbf{g}(x)$
- Let $\mathbf{f}$ be some function, $\mathbf{C}$ some constant. Then the function

$$\mathbf{F}(x) = \mathbf{C} + \sum_{x=0}^{x-1} \mathbf{f}(x)$$

is such that $\mathbf{F}'(x) = \mathbf{f}(x)$ and $\mathbf{F}(0) = \mathbf{C}$. As expected, $\mathbf{F}$ is called a primitive of $\mathbf{f}(x)$.

Discrete Ordinary Differential Equation (ODE)

Discrete ODE: System of equations of the form, where h is some function:

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

With initial value $\mathbf{f}(0,\mathbf{y}) = \mathbf{g}(\mathbf{y})$: Initial Value Problem (IVP) or a Cauchy Problem.

Integral form:

$$\mathbf{f}(x,\mathbf{y}) = \mathbf{f}(0,\mathbf{y}) + \int_0^x \mathbf{h}(\mathbf{f}(x,\mathbf{y}), x, \mathbf{y})\delta x.$$

- Hence, a discrete ODE always have a solution $f : \mathbb{N}^p \to \mathbb{Z}^q$
- Not always true if one wants $f : \mathbb{Z}^p \to \mathbb{Z}^q$

Linear system of discrete ODE

Linear ODE: system of the form

$$\begin{cases} \mathbf{f}'(x,\mathbf{y}) = \mathbf{A}(x,\mathbf{y}) \cdot \mathbf{f}(x,\mathbf{y}) + \mathbf{B}(x,\mathbf{y}) \\ \mathbf{f}(0,\mathbf{y}) = \mathbf{G}(\mathbf{y}) \text{ (initial conditions)} \end{cases}$$

For matrices $\mathbf{A}$ and vectors $\mathbf{B}$ and $\mathbf{G}$.

- Well known and simple kind of system
- Easy to solve in the continous setting

Linear system of discrete ODE

Easy to see that solution is of the form:

$$\mathbf{f}(x,\mathbf{y}) = \left(\overline{2}^{\int_0^x \mathbf{A}(t,\mathbf{y})\delta t}\right) \cdot \mathbf{G}(\mathbf{y}) + \int_0^x \left(\overline{2}^{\int_{u+1}^x \mathbf{A}(t,\mathbf{y})\delta t}\right) \cdot \mathbf{B}(u,\mathbf{y})\delta u.$$

Or, alternatively:

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

with the conventions that $\prod_x^{x-1} \kappa(x) = 1$ and $\mathbf{B}(-1,\mathbf{y}) = \mathbf{G}(\mathbf{y})$
Computational content is clear: the solution can be computed

Bounded sum and product

Arithmetic is used freely below.
Let $g : \mathbb{N}^{p+1} \to \mathbb{N}$,

- Let $f(x, \mathbf{y}) = \sum_{z<x} g(z, \mathbf{y})$ for $x \neq 0$, and 0 for $x = 0$. Function f is the unique solution of :

$$\begin{cases} \frac{\partial f(x,\mathbf{y})}{\partial x} = g(x, \mathbf{y}) \\ f(0, \mathbf{y}) = 0 \end{cases}$$

- Let $f(x, \mathbf{y}) = \prod_{z<x} g(z, \mathbf{y})$ for $x \neq 0$, and 1 for $x = 0$. Function f is the unique solution of :

$$\begin{cases} \frac{\partial f(x,\mathbf{y})}{\partial x} = f(x, \mathbf{y}) \cdot (g(x, \mathbf{y}) - 1) \\ f(0, \mathbf{y}) = 1 \end{cases}$$

Plan

- Introduction to discrete ODE
- A short survey on recursion scheme for complexity
- ODE and complexity classes : characterizing polynomial time computation

Plan

- Introduction to discrete ODE
- A short survey on recursion scheme for complexity
- ODE and complexity classes : characterizing polynomial time computation

Primitive recursive functions

Let $p \in \mathbb{N}$, $g : \mathbb{N}^p \to \mathbb{N}$ and $h : \mathbb{N}^{p+2} \to \mathbb{N}$.
The function $f = \mathrm{REC}(g, h) : \mathbb{N}^{p+1} \to \mathbb{N}$ is defined by primitive recursion from g and h if:

$$\begin{cases} f(0, \mathbf{y}) = g(\mathbf{y}) \\ f(x+1, \mathbf{y}) = h(f(x, \mathbf{y}), x, \mathbf{y}) \end{cases}$$

- High complexity functions
- How to restrict the recursion scheme to lower complexity?

Bounded recursion

Let $g : \mathbb{N}^p \to \mathbb{N}$, $h : \mathbb{N}^{p+2} \to \mathbb{N}$ and $i : \mathbb{N}^{p+1} \to \mathbb{N}$.
The function $f = \mathrm{BR}(g, h) : \mathbb{N}^{p+1} \to \mathbb{N}$ is defined by bounded recursion from g, h and i if

$$\begin{aligned} f(0, \mathbf{y}) &= g(\mathbf{y}) \\ f(x+1, \mathbf{y}) &= h(f(x, \mathbf{y}), x, \mathbf{y}) \end{aligned}$$

under the condition that:

$$f(x, \mathbf{y}) \leq i(x, \mathbf{y}).$$

Key ingredient to capture elementary function and Grzegorczyk's hierarchy

Recursion on notation (Cobham)

Consider $\mathbf{s}_0, \mathbf{s}_1 : \mathbb{N} \to \mathbb{N}$

$$\mathbf{s}_0(x) = 2 \cdot x \text{ and } \mathbf{s}_1(x) = 2 \cdot x + 1.$$

Definition

Function f defined by bounded recursion on notations, i.e. BRN, from functions g, h_0, h_1 et k when:

$$\begin{cases} f(0, \mathbf{y}) = g(\mathbf{y}) \\ f(\mathbf{s}_0(x), \mathbf{y}) = h_0(x, \mathbf{y}, f(x, \mathbf{y})) \text{ for } x \neq 0 \\ f(\mathbf{s}_1(x), \mathbf{y}) = h_1(x, \mathbf{y}, f(x, \mathbf{y})) \\ f(x, \mathbf{y}) \leq k(x, \mathbf{y}) \end{cases}$$

Cobham's approach

$\mathscr{F}_P$ smallest subset of primitive recursive functions

- Containing basis functions :
  Function$\mathbf{0}$, projections p_i^k, successor functions $\mathbf{s}_0(x) = 2 \cdot x$ and $\mathbf{s}_1(x) = 2 \cdot x + 1$, "smash" function $x \sharp y = 2^{|x| \times |y|}$
- Closed by composition
- Closed by bounded recursion on notations

Cobham (62) : $\mathscr{F}_P$ is equal to $\mathbf{FP}$, the class of polynomial time computable functions

Why it works

$$\begin{cases} f(0,\mathbf{y}) = g(\mathbf{y}) \\ f(\mathbf{s}_0(x),\mathbf{y}) = h_0(x,\mathbf{y},f(x,\mathbf{y})) \text{ for } x \neq 0 \\ f(\mathbf{s}_1(x),\mathbf{y}) = h_1(x,\mathbf{y},f(x,\mathbf{y})) \\ f(x,\mathbf{y}) \leq k(x,\mathbf{y}) \end{cases}$$

Why it works

$$\begin{cases} f(0,\mathbf{y}) = g(\mathbf{y}) \\ f(\mathbf{s}_0(x),\mathbf{y}) = h_0(x,\mathbf{y},f(x,\mathbf{y})) \text{ for } x \neq 0 \\ f(\mathbf{s}_1(x),\mathbf{y}) = h_1(x,\mathbf{y},f(x,\mathbf{y})) \\ f(x,\mathbf{y}) \leq k(x,\mathbf{y}) \end{cases}$$

- f is defined from h_0, h_1 and k.
- If $|k(x,\mathbf{y})|$ is polynomial in $|x| + |y|$, then so is $|f(x,\mathbf{y})|$
- Hence, inner terms do not grow too fast!

Why it works

$$\begin{cases} f(0,\mathbf{y}) = g(\mathbf{y}) \\ f(\mathbf{s}_0(x),\mathbf{y}) = h_0(x,\mathbf{y},f(x,\mathbf{y})) \text{ for } x \neq 0 \\ f(\mathbf{s}_1(x),\mathbf{y}) = h_1(x,\mathbf{y},f(x,\mathbf{y})) \\ f(x,\mathbf{y}) \leq k(x,\mathbf{y}) \end{cases}$$

- $|\mathbf{s}_1(x)| = |\mathbf{s}_0(x)| = |x| + 1$
- Then the number of induction steps is in $O(|x|)$.

Going further: syntactic restriction, ramified recursion

- Cobham's work was the starting point of numerous attempts to capture complexity classes by recursion algebras
- Generalize to $\mathbf{L}, \mathsf{NC}^i, \mathsf{AC}^i$ classes
- Alternative approaches that do not require to bound the function *a priori*.
  - Predicative recursion (Bellantoni, Cook)
  - Ramified recurrence (Leivant, Leivant-Marion)
  - ....

Plan

- Introduction to discrete ODE
- A short survey on recursion scheme for complexity
- ODE and complexity classes : characterizing polynomial time computation

Plan

- Introduction to discrete ODE
- A short survey on recursion scheme for complexity
- ODE and complexity classes : characterizing polynomial time computation

Discrete ODE for elementary functions

Definition (Discrete ODE schemata)

Let $g : \mathbb{N}^p \to \mathbb{N}$ and $h : \mathbb{Z} \times \mathbb{N}^{p+1} \to \mathbb{Z}$.
Function f is defined by discrete ODE solving from g and h, denoted by $f = \mathrm{ODE}(g, h)$, if $f : \mathbb{N}^{p+1} \to \mathbb{Z}$ if f solution of:

$$\begin{cases} \frac{\partial f(x,\mathbf{y})}{\partial x} = h(f(x,\mathbf{y}), x, \mathbf{y}) \\ f(0,\mathbf{y}) = g(\mathbf{y}) \end{cases}$$

When h is linear : LI schemata.

Discrete ODE for elementary functions

What about the smallest classes of functions

- that contains $\mathbf{0}$, the projections π_i^p, the successor $\mathbf{s}$, addition $+$, subtraction $-$
- that is closed under composition and discrete linear ODE schemata LI.

Result: Corresponds to elementary functions

Remark: recall the definition of bounded sum and bounded product.

ODE for complexity classes ?

- Elementary functions are of high complexity
- But linear systems is the simplest kind of system
- What can we do (i.e. what can we restrict more) to capture smaller complexity classes and in particular the class of polynomial time computable functions $\mathbf{FP}$?

Derivation along a function

Definition ($\mathcal{L}$-ODE)

Let $\mathcal{L} : \mathbb{N}^{p+1} \to \mathbb{Z}$. We write

$$\frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \mathcal{L}} = \frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \mathcal{L}(x, \mathbf{y})} = \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}), \tag{2}$$

as a formal synonym for
$\mathbf{f}(x+1, \mathbf{y}) = \mathbf{f}(x, \mathbf{y}) + (\mathcal{L}(x+1, \mathbf{y}) - \mathcal{L}(x, \mathbf{y})) \cdot \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y})$.

Inspired by the classical formula:

$$\frac{\delta f(x, \mathbf{y})}{\delta x} = \frac{\delta \mathcal{L}(x, \mathbf{y})}{\delta x} \cdot \frac{\delta f(x, \mathbf{y})}{\delta \mathcal{L}(x, \mathbf{y})}.$$

$\mathcal{L}$-ODE

The equality

$$\frac{\delta f(x, \mathbf{y})}{\delta x} = (\mathcal{L}(x+1, \mathbf{y}) - \mathcal{L}(x, \mathbf{y})) \cdot \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y})$$

implies that the value of the derivative i.e. the variation of the function has to be considered only when

$$\mathcal{L}(x+1, \mathbf{y}) - \mathcal{L}(x, \mathbf{y}) \neq 0$$

Consequence: only as many values to consider to compute $f(x, \mathbf{y})$ as the number of times $\mathcal{L}(t, \mathbf{y})$ changes between $t = 0$ and $t = x$...

Application: if $\mathcal{L}(x, \mathbf{y}) = \ell(x)$ then only a logarithmic in x number of values

Towards capturing $\mathbf{FP}$

Deriving along the logarithm function is not sufficient to capture $\mathbf{FP}$

- It is easily seen that the solution of

$$\frac{\partial f(x)}{\partial \ell(x)} = f(x) \cdot (f(x) - 1) \tag{3}$$

  is a fast growing function (output is exponential in size)
- Idea: combine linearity and derivation along some particular function $\mathcal{L}$ i.e. systems :

$$\frac{\partial \mathbf{f}(x, \mathbf{y})}{\partial \mathcal{L}} = \mathbf{h}(\mathbf{f}(x, \mathbf{y}), x, \mathbf{y}), \tag{4}$$

  where
  - h is "linear"
  - $\mathcal{L}$ has a polylogarithmic number of values

$\mathbb{DL}$

Definition ($\mathbb{DL}$)

Let $\mathbb{DL}$ be the smallest subset of functions,

- that contains $\mathbf{0}$, $\mathbf{1}$, projections π_i^p, the length $\ell(x)$, functions $x{+}y$, $x{-}y$, $x \times y$, the sign function $\mathrm{sg}(x)$
- closed under composition (when defined) and linear length-ODE scheme:

$$\frac{\partial \mathbf{f}(x,\mathbf{y})}{\partial \ell} = \mathbf{u}(\mathbf{f}(x,\mathbf{y}),x,\mathbf{y}) \quad \text{and } \mathbf{f}(0,\mathbf{y}) = \mathbf{g}(\mathbf{y})$$

where $\mathbf{u}$ is *essentially linear* in $\mathbf{f}(x,\mathbf{y})$.

A characterization of $\mathbf{FP}$

Theorem: $\mathbb{DL} = \mathbf{FP}$

Proof of ($\subseteq$): Roughly speaking

- The derivation along $\ell(x)$ (or any $\mathcal{L}$ with polylog "jumps") permits to control the number of steps
- Linearity of the system permits to control the size of the output

Proof of ($\supseteq$): By a direct expression of a polynomial computation of a register machine.

Conclusion, questions and work in progress

- Study the expressive and computational power of discrete ODE
- Appears
  - to be a convenient tool for algorithm design
  - to elegantly capture complexity notions

- Extend the work to other classes (**FPSPACE**, $\mathbf{NP}$, circuit classes)
- smaller derivation steps and allowing errors
- Generalize to the continuous setting