A Polynomialization Algorithm for Elementary Functions and ODEs, and their Compilation into Chemical Reaction Networks

Mathieu Hemery, François Fages, Sylvain Soliman

A Polynomialization Algorithm for Elementary Functions and ODEs, and their Compilation into Chemical Reaction Networks

Mathieu Hemery, François Fages, and Sylvain Soliman

Inria Saclay Île-de-France, EP Lifeware, Palaiseau, France

Abstract. In this short paper extracted from [7], we present a polynomialization algorithm of quadratic time complexity to transform a system of elementary differential equations in polynomial differential equations (PODE). This algorithm is used as a front-end transformation in a pipeline to compile any elementary mathematical function, either of time or of some input variable, into a finite Chemical Reaction Network (CRN) which computes it. We illustrate the performance of our compiler on a benchmark of elementary functions which serve as formal specification of CRN design problems in synthetic biology, and as comparison basis with natural CRNs exhibiting similar behaviours.

1 Introduction

CRNs provide a standard formalism in chemistry and biology to describe, analyze, and also design complex molecular interaction networks given with rate functions. In the perspective of systems biology, they are a central tool to analyze the high-level functions of the cell in terms of their low-level molecular interactions. While in synthetic biology, they constitute a target programming language to implement in chemistry useful functions in either living cells or artificial devices. The interpretation of a CRN by Ordinary Differential Equations (ODE) allows us to give a precise mathematical meaning to the notion of analog computation and high-level functions computed by cells in Systems Biology:

Definition 1. *[4,5,9] A function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is* generated by a CRN *on some species y with given initial concentrations for all species, if the ODE associated to the CRN has a unique solution verifying $\forall t \geq 0\ y(t) = f(t)$.*

That first definition states that a positive real function of one positive argument is *generated* by a CRN for some given initial concentration values, if the graph of that function is given by the temporal evolution of the concentration of one molecular species in that CRN.

Definition 2. *[4,5][1] A function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is* computed by a CRN *for some input species x, output species y, and initial concentrations given for all*

[1] For the sake of simplicity of the definition given here, we omit the error control mechanism that requires one extra CRN species z verifying:
$\forall t > 1\ |y(t) - f(x(0))| \leq z(t),\ \forall t' > t\ z(t') < z(t)$ and $\lim_{t\to\infty} z(t) = 0$.

species apart from x, if for any input concentration value $x(0)$ for x, the ODE initial value problem associated to the CRN has a unique solution satisfying $\lim_{t\to\infty} y(t) = f(x(0))$.

The second definition for computed input/output functions is used in [4] to show the Turing completeness of continuous CRNs in the sense that any computable function over the real numbers can be computed by a CRN over a finite set of formal molecular species, using at most bimolecular reactions and mass action law kinetics. The proof uses a previous result of Turing completeness for functions defined by polynomial ordinary differential equation initial value problems (PIVP) [1], the dual-rail encoding of real variables by the difference of concentration between two molecular species, and a back-end quadratization transformation to restrict to elementary reactions with at most two reactants [3,6,2]. This proof gives rise to a pipeline implemented in BIOCHAM-4[2], to compile any computable real function presented by a PIVP into a finite CRN.

However in practice, it is not immediate to define a PIVP to generate or compute a desired function. In this article, we solve this problem for elementary functions over the reals, by adding to the compilation pipeline a front-end module to transform any elementary function into a PIVP which either generates or computes it. More precisely, we present a polynomialization algorithm to transform any Elementary ODE system (EODE), i.e. ODE system made of elementary differential functions, into a polynomial one (PODE). This algorithm proceeds by introducing variables for computing the non-polynomial terms of the input, eliminating such terms from the ODE by rewriting, and obtaining the ODE for the new variables by formal derivation. The derivation steps may bring new non-polynomial terms requiring new variables. We show the termination of that algorithm, with quadratic time complexity, and that only a linear number of new variables, in terms of the size of the input expression, are actually needed.

2 Polynomialization Algorithm for Elementary ODEs

The core of Alg. 1 for the polynomialization of an EODE system is the detection of the elements of the derivatives that are not polynomial, and their introduction as new variables. Then symbolic derivation and syntactic substitution allow us to compute the derivatives of the new variables and to modify the system of equations accordingly. It is worth noting that the list of substitutions has to be memorized along the way, therefore handling an algebro-differential system during the execution of the algorithm, as they may reappear during the derivation step. This typically occurs when the derivation graph harbors a cycle like: $\cos \to \sin \to \cos$. Nevertheless, a particular treatment has to be applied to the case of non-integer or negative power as they form an infinite set and may thus produce infinite chains of derivations. When adding the new variable $N = X^p$

[2] `http://lifeware.inria.fr/biocham/`. All experiments described in this paper are available at `https://lifeware.inria.fr/wiki/Main/Software#CMSB21`

Algorithm 1 Polynomialization of an EODE system

```
1: Input: A set of ODEs of the form $\{x' = f_x(x, y, \ldots), y' = f_y(x, y \ldots), \ldots\}$.
2: Output: A set of PODEs where the initial variables $x, y, \ldots$ are still present and
   accept the same solutions.
3: Initialize $Transformations \leftarrow \emptyset$ and $PolyODE \leftarrow \emptyset$
4: while ODE is not empty do
5:     take and remove $Var' = Derivative$ from $ODE$;
6:     $NewDerivative \leftarrow$ apply $Transformations$ to $Derivative$;
7:     $Terms \leftarrow$ set of maximal non-polynomial subterms of $NewDerivative$;
8:     for all $Term$ in $Terms$ do
9:         add $(Term \mapsto NewVar)$ to $Transformations$;
10:        $TermDerivative \leftarrow$ the symbolic derivative of $Term$;
11:        add $(NewVar' = TermDerivative)$ to $ODE$;
12:    $PolyDerivative \leftarrow$ apply $Transformations$ to $Derivatives$;
13:    add $(Var' = PolyDerivative)$ to $PolyODE$;
14: return $PolyODE$
```

to the system, we explicitly replace the expression X^{p-1} by N/X in the derivatives, hence the algorithm introduces the new variable $1/X$. This makes the final PODE non analytic in $X = 0$ which is linked to the fact that exponentiation apart from the polynomial case is actually not analytic in 0.

Lemma 1. *For any finite set F of formally differentiable functions over the reals, Alg. 1 terminates if $\forall f \in F, f' \in \overline{F}$ where $\overline{F}$ is the algebra of F over $\mathbb{R}$.*

Proposition 1. *Alg. 1 terminates on elementary functions over the reals, with quadratic time complexity and at most a linear number of new variables.*

Table 1 gives some performance figures about the complete compilation pipeline in terms of total computation time and size of the synthesized CRNs on a benchmark of functions considered in [6] for the quadratization problem which is the most expensive step. It is worth noting that neither the polynomialization nor the quadratization are unique, even when imposing optimality in the number of introduced variables. For example, the two CRN of hill4 compiled in our benchmark with the two options for quadratization (heuristics `fastSAT` or `sat_species` optimization) are different but both have the same number of species and reactions. The Hill5 CRN is one synthetic analog of the natural MAPK CRN since it computes a similar input/output stiff sigmoid [8], yet using 22 species and 33 reactions. However, the input of the MAPK CRN is a catalyst not consumed by the downward reactions, whereas in our CRN synthesis scheme, the input is generally consumed. The MAPK network thus illustrates an interesting case of *online* analog computation which is currently not specifically considered in our framework.

Acknowledgments: support by ANR-17-CE40-0036 and ANR-20-CE48-0002 grants and discussions with Olivier Bournez, Gleb Pogudin and Amaury Pouly.

Function	`fastSAT` time (ms)	`fastSAT` number of species	`fastSAT` number of reactions	`sat_species` time (ms)	`sat_species` number of species	`sat_species` number of reactions
Hill1	80	4	5	85	3	3
Hill2	90	6	10	82	5	8
Hill3	100	6	10	115	6	12
Hill4	100	7	13	162	7	13
Hill5	110	8	16	550	7	11
Hill10	160	13	31	timeout		
Hill20	380	23	61	timeout		
Logistic	80	3	5	85	3	5
Double exp.	80	3	4	85	3	4
Gaussian	85	3	4	85	3	4
Logit	95	4	7	100	4	6

Table 1. Performance results on the benchmark of CRN design problems of [6].

References

1. O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317–335, 2007.
2. A. Bychkov and G. Pogudin. Optimal monomial quadratization for ode systems. In *Proceedings of the IWOCA 2021 - 32nd International Workshop on Combinatorial Algorithms*, July 2021.
3. D. C. Carothers, G. E. Parker, J. S. Sochacki, and P. G. Warne. Some properties of solutions to polynomial systems of differential equations. *Electronic Journal of Differential Equations*, 2005(40):1–17, 2005.
4. F. Fages, G. Le Guludec, O. Bournez, and A. Pouly. Strong Turing Completeness of Continuous Chemical Reaction Networks and Compilation of Mixed Analog-Digital Programs. In *CMSB'17: Proceedings of the fiveteen international conference on Computational Methods in Systems Biology*, volume 10545 of *Lecture Notes in Computer Science*, pages 108–127. Springer-Verlag, Sept. 2017.
5. D. Graça and J. Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, 2003.
6. M. Hemery, F. Fages, and S. Soliman. On the complexity of quadratization for polynomial differential equations. In *CMSB'20: Proceedings of the eighteenth international conference on Computational Methods in Systems Biology*, Lecture Notes in BioInformatics. Springer-Verlag, Sept. 2020.
7. M. Hemery, F. Fages, and S. Soliman. Compiling elementary mathematical functions into finite chemical reaction networks via a polynomialization algorithm for ODEs. In *CMSB'21: Proceedings of the nineteenth international conference on Computational Methods in Systems Biology*, Lecture Notes in BioInformatics. Springer-Verlag, Sept. 2021.
8. C.-Y. Huang and J. E. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *PNAS*, 93(19):10078–10083, Sept. 1996.
9. C. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics*, 20:337–354, 1941.