19

Biochemical Programs and Analog-Digital Mixed Algorithms in the Cell

In this chapter, we take an IT perspective in seeking to understand how computation is carried out in the cell to maintain itself in its environment, process signals and make the decisions that determine its fate. The continuous nature of many protein interactions leads us to consider mixed analog–digital computation models, for which recent results in the theory of analog computability and complexity establish fundamental links with classical programming. We derive from these results a compiler of behavioral specifications into biochemical reactions, which can be compared to natural circuits acquired through evolution. We illustrate this approach through the example of the *mitogen-activated protein kinase* (MAPK) signaling module, which has a function of analog–digital converter in the cell, and through the cell cycle control.

19.1. Introduction

“Mathematics is the art of assigning the same name to different things” Henri Poincaré

One of the lessons of computer science is that digital computation allows us to scale up to very large programs, unlike analog computation. Nonetheless, when considering cellular processes from the perspective of a computer scientist, even if the discrete nature of activation is indeed present in all-or-none of the genes, what strikes most is the preponderance of the gradual activation of protein complexes and the importance of the time that these transformations take within large networks of interactions. In this chapter, we strive to assume the significance of analog

Chapter written by François FAGES and Guillaume LE GULUDEC. Inria Saclay Ile de France.

computation in the cell and lay the foundation for a theory of biochemical computation in order to analyze natural circuits through the definition of tools for the specification of their operations, for the compilation of these specifications in synthetic circuits and for the comparison of their computational complexities.

19.2. Biochemical programs

19.2.1. *Syntax*

Formally, a biochemical reaction here is a rule of the form

$$a_1x_1 + a_2x_2 + \ldots \xrightarrow{f} b_1x_1 + b_2x_2$$

where the a_i and b_j are *stoichiometric coefficients* and the x_i are the chemical species in the system. f is the *kinetic function* of the reaction. At most, *elementary reactions* have two reactants and can appear in five well-known forms: binding or complexation $x + y \rightarrow z$, unbinding $z \rightarrow x + y$, transformation or transportation $x \rightarrow y$, synthesis $x \rightarrow x + y$ (or $\_ \rightarrow y$) and passive $x \rightarrow \_$ or active $x + y \rightarrow y$ degradation.

It will be assumed that reactions follow the *law of mass action*, that is to say, the kinetic function of each reaction is of the form $f = kx_1^{a_1} \ldots x_n^{a_n}$ with k constant. The reaction is then denoted in the form $R \xrightarrow{k} P$ (and we omit k if $k = 1$). The other conventional kinetic functions are obtained by means of reduction of elementary reaction systems assuming that states are quasi-stationary, for example Michaelis–Menten kinetics in $\frac{v*x}{c+x}$ for an enzymatic transformation reaction of a substrate x by an enzyme in smaller quantity, or Hill kinetics in $\frac{v*x^n}{c^n+x^n}$ for a cooperative allosteric enzyme reaction [SEG 84].

It should be noted that conservation of matter cannot be satisfied, for example in the case of synthesis or degradation reactions. Such reactions are physically impossible but ought to be interpreted as abstractions of physical reactions that overlook certain molecular species. For instance, a formal synthesis reaction $\_ \rightarrow x$ may in fact represent the transcription of a gene supposed to be active in RNA x or directly in protein x, or still protein activation $\underline{x}$ by phosphorylation reaction $\underline{x} + y \rightarrow x + y$, ignoring its non-phosphorylated inactive form $\underline{x}$ as well as the kinase enzyme y both assumed to be present. In the latter case, the degradation reaction $x \rightarrow \_$ can then actually denote the dephosphorylation reaction $x + z \rightarrow \underline{x} + z$ returning to the inactive form under the effect of a phosphatase z also ignored.

Consequently, these formal reactions do not prejudice their physical implementation.

19.2.2. *Semantics*

Such systems of reactions can be interpreted in different ways that can be linked by means of approximation [GIL 77] or abstraction [FAG 8a] relations, to form a hierarchy of semantics, especially in the computational context of abstract interpretation [COU 77].

Differential semantics associates continuous concentrations with molecular species and the system of differential equations

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_j (b_i^j - a_i^j) f_j(x_1, \dots, x_n)$$

in which the index j ranges the whole set of reaction rules of the system. A system is at *steady state* on x_i when $\frac{\mathrm{d}x_i}{\mathrm{d}t} = 0$.

Stochastic semantics associates an integer number of molecules (or concentration level) to each molecular species and a continuous-time Markov chain, in which transition probabilities are defined based on the kinetic functions through normalization, and the time of the next reaction is given by an exponential law.

Petri net semantics acts on the same states as stochastic semantics but does not consider kinetics, probabilities and continuous time (abstraction by forgetful functor). Nevertheless, the notions of Petri net invariants provide information on differential and stochastic semantics, such as conservation laws (P-invariants) and extreme flows (T-invariants) that are an essential tool for the analysis of metabolic networks.

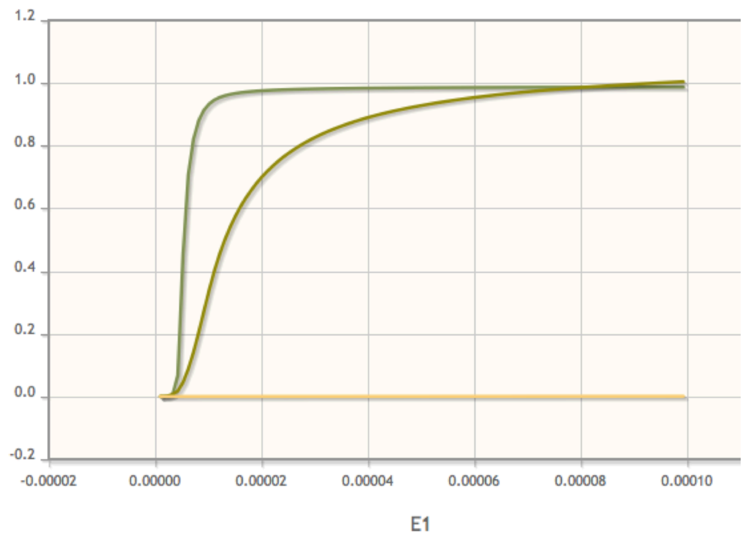
Boolean semantics yet abstract discrete states into Boolean states concerning the presence/absence of molecular species (abstraction $(> 0) : \mathbb{N} \to \{0, 1\}$, or $(> \theta)$ for a threshold θ) and associates an asynchronous Boolean transition system to the reaction system. *Model-checking* tools then make it possible to verify properties of accessibility and trajectories in large reaction systems whose kinetics is unknown [CHA 04]. They make it possible to show that if a behavior is not possible in Boolean semantics then it is also not in stochastic semantics regardless of kinetic functions [FAG 8a].

19.2.3. *Example of MAPK signaling networks*

MAPK circuits are extremely common signaling modules that can be found in several copies in eukaryotic organisms. In these signaling pathways, proteins activated by phosphorylation are themselves kinases that catalyze other phosphorylations in cascade. Thus, the MAPK cascade has three phosphorylation stages for a total of 30 elementary reactions: the input `E1` of the cascade, directly connected to the membrane receptor, catalyzes the phosphorylation of the kinase KKK of the first stage, which in turn twice phosphorylates the kinase KK of the second stage. Furthermore, in this doubly phosphorylated form the latter phosphorylates the protein K of the last stage of the cascade, which is itself doubly phosphorylated PP_K and capable of migrating inside the kernel and activating or inhibiting the transcription of certain genes.

Huang and Ferrell [HUA 96] have proposed an explanation for this structure by showing that MAPK cascades exhibit a response (at steady state) in the form of a Hill function (thus similar to the response of a chain of cooperative allosteric enzymatic reactions), that is to say that by means of denoting by (u, y) the input–output pair of the system, it was possible to approximate the dose–response diagram by an equation of the form $y(u) \approx \lambda \frac{u^\alpha}{c^\alpha + u^\alpha}$ with α of the order of 4.9 at the third level PP_K $\alpha \sim 1.7$ at the second PP_KK and $\alpha = 1$ at the first level P_KKK, which is Michaelian:

```
biocham: present(E1=3.0e-5, KKK=0.003, KK=1.2, K=1.2,
                 E2=0.0003, KKPase=0.0003, Kpase=0.12).
biocham: dose_response(E1, 1e-6, 1e-4, 500, {PP_K, PP_KK, P_KKK}).
```



The MAPK cascade therefore acts as an *analog–digital converter* that transforms a continuous signal on input (concentration of active receivers E1) into an digital signal on output (all-or-nothing activation of the protein PP_K).

19.3. Behavioral logical specifications

The time logic CTL provides a very powerful expression language for the analysis of the Boolean dynamics of a reaction system [CHA 04]. In this logical language, a propositional formula defines a set of states (the set of states that satisfy it), and modal operators define the truth value of formulas in one or all future states, on one or all the branches of the Boolean transition system associated with the reactions. Besides accessibility properties based on a set of initial states, we can express a resulting stationary state (which cannot be left), a steady state (which cannot be left), compulsory crossing points (*checkpoints*) to reach another state, possibilities of oscillations, etc.

The MAPK example verifies, for instance, accessibility properties of stationary or stable states but also oscillation properties automatically verified by *model-checking* methods.

```
biocham: generate_ctl.
reachable(steady(K))
reachable(steady(P_K))
reachable(steady(PP_K))
...
checkpoint(E1,P_KKK)
...
oscil(KKK)
oscil(KK)
oscil(K)
```

In addition, if we only consider the output activation property Kpp as the specification of the behavior, the command `biocham: reduce_model (reachable(PP_K))` determines through *model-checking* that the 15 reverse dephosphorylation reactions are not useful and can be automatically removed from the model.

The oscillation properties of the abstract Boolean semantics of MAPK do not imply that they occur in the differential semantics. It could rather be intuitively deduced that they cannot occur therein because MAPK is a cascade of reactions evolving from the input stage to the output stage without reverse reaction. It is, nevertheless, possible to obtain such oscillations in differential semantics for certain values (8%) of the initial parameters and concentrations [QIA 07]. This can be verified by expressing the condition of oscillation in first-order time logic with quantitative constraints [FAG 14] and using stochastic optimization algorithms in order to find parameter values satisfying these constraints [RIZ 11]. The counterintuitive nature of these oscillations yields from the fact that there is no negative feedback reaction, but a complexation reaction between the kinase of the upper stage and its substrate of the lower stage that creates a *negative influence* of the lower substrate toward the upper kinase (by sequestration), and thereby, a negative circuit in the *influence diagram* (and not of reaction), which is indeed a necessary condition for oscillations [THO 81, SNOU 98, FAG 08b, FAG 15].

19.4. Analog specifications

19.4.1. *Computability and analog complexity theory*

"The varied titles of Turing's published work disguise its unity of purpose. The central problem with which he started, and to which he constantly returned, is the extent and the limitations of mechanistic explanations of nature". Max Newman

The Church–Turing thesis states that there is only one single notion of computability and as a result, all methods for mechanistic computation devised so far have always proved to be codable in Turing machines. A computational meaning can thus be given to analog computations by considering the notion of computability of computational analysis, which is based on that of the computation of real numbers with arbitrary precision, yet finite, by Turing machines:

DEFINITION 19.1.– *A real number $r \in \mathbb{R}$ is computable (respectively, in polynomial time) as defined in computational analysis if there is an approximation program of r in arbitrary precision, or more specifically a Turing machine, which takes a precision $p \in \mathbb{N}$ as input and yields on output (respectively in polynomial time according to p) a rational number $r_p \in \mathbb{Q}$ such that $|r - r_p| \leq 2^{-p}$.*

DEFINITION 19.2.– *A function $f : [a, b] \to \mathbb{R}$ is computable (respectively, in polynomial time) if there is a Turing machine that computes $f(x)$ (respectively, in polynomial time) with an oracle for x.*

Shannon's *General Purpose Analog Computer* (GPAC) [SHA 41] is a model of analog computation using block circuits. We consider a set of inputs including time $t, x, y, z, \ldots$ and four types of blocks: constants, sums, products and Stieltjes integral of a variable with respect to another variable. However, in this original presentation, some circuits may not have any solution, or still have several. This problem has been solved by Graça and Costa [GRA 03] who have established a satisfactory definition for functions generable by a GPAC as a solution to initial value problems in polynomial differential equations (PIVP).

DEFINITION 19.3.– *[GRA 03] A function $f : \mathbb{R} \to \mathbb{R}$ is* GPAC-generable *if it is a component of the solution $y(t)$ of the ordinary differential equation $y'(t) = p(y(t))$ for a vector of polynomials $p \in \mathbb{R}^n[\mathbb{R}^n]$ and with initial values $y(0)$.*

For example, the GPAC `(a = integral integral -1*a)` (in which integrations are carried out with respect to time t) directly gives $y''(t) = -y(t)$ and *generates* the function $cos(t)$ with $y(0) = 1$. This class of functions exhibits a number of properties such as stability through addition, multiplication and composition, and in addition includes basic functions such as trigonometric, exponential, logarithm, etc., functions. This notion of generality has for some time been regarded as synonym of computability, which made that GPAC was a computational model less reliable than computational analysis. Nevertheless, it is possible to define a notion of computability in terms of GPAC both natural and powerful, still by means of PIVP, but by approximation of the result on a component of the system for any input:

DEFINITION 19.4.– *[GRA 03] A function $f : [a, b] \to \mathbb{R}$ is* GPAC-computable *if there are polynomial vectors $p \in \mathbb{R}^n[\mathbb{R}^n]$ and $q \in \mathbb{R}^n[\mathbb{R}]$ and a function $y : \mathbb{R}^n \to \mathbb{R}^n$ such that $y(0) = q(x)$, $y'(t) = p(y(t))$ and $\lim_{t\to\infty} y_1(t) = f(x)$.*

The calculation of f with the argument x thus involves putting the system in a state polynomially depending of x, and then allowing the system to develop according to the dynamics described by p. The result of the computation is then obtained in the first component of the system, with an accuracy proportionally increasing as time elapses.

The following theorem due to [BOU 06] thus perfectly reconciles the concepts of digital and analog computability:

THEOREM 19.1.– *[BOU 06] A function is computable from the perspective of computable analysis if and only if it is GPAC-computable.*

In addition, Pouly [POU 15] deduces thereof, for the first time, a purely analog characterization of the complexity class Ptime. We should note first that a naive definition of complexity in terms of time to wait to obtain fixed precision is not adequate. Indeed, it is always possible to contract time in the PIVPs through a change in variable. Pouly solves this problem simply by taking the *trajectory length* as measure of computation complexity:

DEFINITION 19.5.– *[POU 15] A function $f : [a, b]^n \to \mathbb{R}^m$ is said to be Ω-computable in length (with $\Omega : \mathbb{R}^2_+ \to \mathbb{R}$) if there exists $p \in \mathbb{R}^d[\mathbb{R}^d]$, $q \in \mathbb{R}^d[[a, b]^n]$ such that for all $x \in \mathrm{dom} f$, there exists $y : \mathbb{R}_+ \to \mathbb{R}^d$ such that for all $t \in \mathbb{R}_+$:*

– *$y(0) = q(x)$ and $y'(t) = p(y(t))$;*

– *for any* μ, *if* $\int_0^t ||y'(\tau)||_2 \, \mathrm{d}\tau \geq \Omega(|x|, \mu)$, *then* $|y_{1..m}(t) - f(x)| \leq e^{-\mu}$.

THEOREM 19.2.– *[POU 15] The functions* Ω*-computable in length, where* Ω *is a polynomial, are exactly functions computable in polynomial time as defined by computational analysis.*

19.4.2. *Computability and biochemical algorithmic complexity*

The previous results provide solid foundations to study biochemical analog computation. However, a biochemical system is a dynamic system on the cone $\mathbb{R}^n_+$, where the state is the datum of concentrations of *positive* values of the species in the system. The dynamic given by the law of mass action leads to a system of the form $\frac{\mathrm{d}y}{\mathrm{d}t} = p(y(t))$ with $p(y)_i = \sum_j (b_i^j - a_i^j) k_j y_1^{a_1^j} \dots y_n^{a_n^j}$. It can be seen that additional constraints appear compared to general GPACs: the components y_i must always be positive, and the monomials of p_i whose coefficient is negative must have a non-zero exponent in y_i. These constraints are necessary conditions so that there exists a set of chemical species $\mathcal{X}$ and a set of physically feasible reactions $\mathcal{R}$ for the system $(\mathcal{X}, \mathcal{R})$ to react according to the dynamic $y' = p(y)$.

Interestingly, the previous results can be generalized to positive systems and to biochemical reaction systems. The underlying idea is to encode each component y_i by the difference between two positive components y_i^+ and y_i^-.

THEOREM 19.3.– *Any GPAC can be encoded in a positive system chemically feasible of double size. If in addition the initial GPAC exhibits polynomial computational complexity, then the new system will also have polynomial complexity.*

DEMONSTRATION.– Let $p \in \mathbb{R}^n[\mathbb{R}^n]$. Each $y_i \in \mathbb{R}$ can be encoded by a couple $(y_i^+, y_i^-) \in \mathbb{R}^2_+$ such as at any moment, $y_i = y_i^+ - y_i^-$, and where the $y_i^\pm$ are subject to dynamics that correspond to a chemical system.

We define $\hat{p}_i(y_1^+, y_1^-, \dots, y_n^+, y_n^-) = p_i[y = y^+ - y^-]$, then we write $\hat{p}_i = \hat{p}_i^+ - \hat{p}_i^-$, where the monomials of $\hat{p}_i^+$ and $\hat{p}_i^-$ have positive coefficients. A new system is then defined by:

$$\forall i \leq n, \begin{cases} {y_i^+}' = \hat{p}_i^+ - f_i y_i^+ y_i^- \\ {y_i^-}' = \hat{p}_i^- - f_i y_i^+ y_i^- \\ y_i^+(0) = \max(0, y_i(0)) \\ y_i^-(0) = \max(0, -y_i(0)) \end{cases}$$

where the f_i are polynomials with positive coefficients such that $f_i \geq \max(\hat{p}_i^+, \hat{p}_i^-)$.

The additional terms $-f_i y_i^+ y_i^-$ are implemented by annihilation reaction $y^+ + y^- \xrightarrow{f_i} \_$ and make it possible to obtain a system where one of the $y_i^\pm$ still remains "small".

We note that we have $y_i^{+\prime} \leq \hat{p}_i^+(1 - y_i^+ y_i^-)$ and $y_i^{-\prime} \leq \hat{p}_i^-(1 - y_i^+ y_i^-)$, such that $(y_i^+ y_i^-)' \leq q \cdot (1 - y_i^+ y_i^-)$, where q is a positive coefficient polynomial. Since that at $t = 0$ we have $y_i^+ y_i^- = 0$, using a Grönwall inequality it can be deduced thereof that we have always: $y_i^+ y_i^- \leq 1$. It follows that $|y_i^\pm| \leq |y_i| + 1$, then $|y^\pm| \leq |y| + n$. Consequently, if the original system is upper bounded in space by a polynomial of the size of the input and time, then it is still the case for the positive system obtained by the previous construct.

Each monomial of the form $\lambda x_1^{\alpha_1} \ldots x_m^{\alpha_m}, \lambda > 0$ appearing in the right-hand side term of an equality of the form $y = p$ is implemented by the reaction $\alpha_1 x_1 + \ldots + \alpha_m x_m \xrightarrow{\lambda} y + \alpha_1 x_1 + \ldots + \alpha_m x_m$. □

It is also possible to consider elementary reactions only, more precisely having at most two reactants because any PIVP is equivalent to a quadratic PIVP:

THEOREM 19.4.– *[CAR 05] Any PIVP solution is a PIVP of degree at most equal to two.*

DEMONSTRATION.– The proof involves introducing variables for each monomial as it follows:

$$v_{i_1,\ldots,i_n} = y_1^{i_1} y_2^{i_2}, \ldots, y_n^{i_n}$$

We have $y_1 = v_{1,0,\ldots,0}$ etc. The substitution of these variables in the differential equations of the y_i' yield first-degree equations in the variables $v_{i_1,\ldots,i_n}$. The differential equations for the variables that are not y_i assume the form

$$v'_{i_1,\ldots,i_n} = \sum_{k=0}^{n} i_k * v_{i_1,\ldots,i_k-1,\ldots,i_n} * y'_k$$

namely of the second degree since the y'_k are linear combinations of the variables $v_{i_1,\ldots,i_n}$. □

These results show that differential elementary biochemical calculus has all the power of expression of PIVPs, and the Turing completeness of this computational mode can be deduced from Theorem 19.1.

THEOREM 19.5.– *Elementary biochemical reaction systems on finite universes of molecules are Turing-complete in differential semantics.*

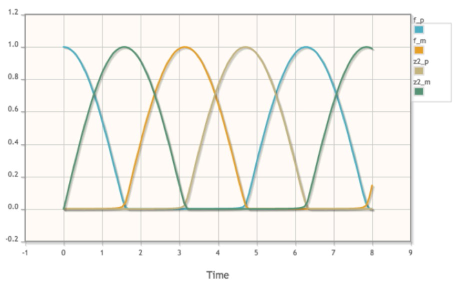
Note that this is not the case in discrete semantics of systems of reactions, where the Turing completeness requires that other mechanisms be added, such as the unbounded dynamic creation of membranes [BUS 06, BER 92] or the reactions of unbounded polymerization [CAR 10].

19.4.3. *GPAC biochemical compilation*

The proof of Theorem 19.3 shows how to biochemically implement GPACs by doubling the number of variables in positive and negative portions, and by implementing each monomial of the differential equations by a synthesis or degradation catalytic reaction according to its sign. Similarly, the proof of Theorem 19.4 shows how to only consider elementary reactions of at most two reactants by increasing the number of molecular species, more specifically by sacrificing the dimension of the system for minimizing the degrees.

These are the principles of our biochemical compiler that translates a mathematical function defined by a PIVP in a system of elementary reactions. For example, the oscillator defined by the time function $f = cos(t)$ is compiled into six elementary reactions (hereafter catalysts are denoted in brackets, `a =[c]=> b` is an abbreviation for `a+c => b+c`):

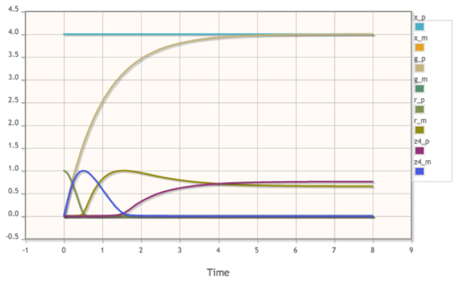
```
biocham: compile(cos,time,f).
_ =[z2_p]=> f_p.
_ =[z2_m]=> f_m.
_ =[f_m]=> z2_p.
_ =[f_p]=> z2_m.
fast*z2_m*z2_p for z2_m+z2_p=>_.
fast*f_m*f_p for f_m+f_p=>_.
present(f_p,1).

biocham: simulation(time:8).
```



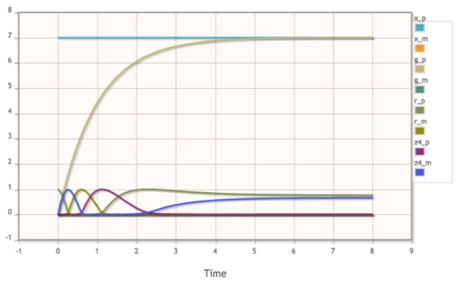
The construction of a GPAC that *calculates* the value of $f(x)$ at any point can also be achieved from a GPAC that *generates* the time function $f(t)$, by assigning the value of f at one point x_0 for which $f(x)$ does not differ following the trajectory $\gamma(t) = x + (x_0 - x)e^{-\lambda t}$, $\lambda > 0$ [POU 15]. The compilation of the GPAC that computes the function $cos(x)$ thus adds computation reactions of the result along the trajectory toward the argument, for example for $r = cos(4)$:

```
biocham: compile(cos,x,r).
_ =[g_m]=> g_p.
_ =[x_p]=> g_p.
_ =[g_p]=> g_m.
_ =[x_m]=> g_m.
_ =[g_m+z4_p]=> r_p.
_ =[g_p+z4_m]=> r_p.
_ =[x_m+z4_m]=> r_p.
_ =[x_p+z4_p]=> r_p.
_ =[g_m+z4_m]=> r_m.
_ =[g_p+z4_p]=> r_m.
_ =[x_p+z4_m]=> r_m.
_ =[x_m+z4_p]=> r_m.
_ =[g_m+r_m]=> z4_p.
_ =[g_p+r_p]=> z4_p.
_ =[x_p+r_m]=> z4_p.
_ =[x_m+r_p]=> z4_p.
_ =[g_m+r_p]=> z4_m.
_ =[g_p+r_m]=> z4_m.
_ =[x_m+r_m]=> z4_m.
_ =[x_p+r_p]=> z4_m.
fast*z4_m*z4_p for z4_m+z4_p=>_.
fast*r_m*r_p for r_m+r_p=>_.
fast*g_m*g_p for g_m+g_p=>_.
fast*x_m*x_p for x_m+x_p=>_.
present(r_p,1).
```

```
biocham: present(x_p,4).
biocham: simulation(time:8).
```



```
biocham: present(x_p,7).
biocham: simulation(time:8).
```



19.4.4. *Analog–digital converter compared to MAPK*

We have seen that the three-staged structure of the MAPK signaling circuit could be explained by the input–output function resulting thereof in all-or-nothing. However, following what has been previously exposed we can focus on directly compiling the input–output Hill function. The function $y(t) = \frac{t^\alpha}{c^\alpha + t^\alpha}$ satisfies the differential equation $y' = \frac{\alpha}{t} y(1-y)$. This function is thus easily GPAC-generable and GPAC-computable, for instance, by the following system:

$$\left\{ \begin{array}{ccc ccc} \gamma & \to & \_ & y_2 + \alpha + x + y_1 & \to & \alpha + x + y_1 + 2y_2 \\ x & \to & x + \gamma & 2y_2 + \alpha + x + y_1 & \to & \alpha + x + y_1 + y_2 \\ 2y_1 + x & \to & y_1 + x & y_2 + \alpha + \gamma + y_1 & \to & \alpha + \gamma + y_1 \\ 2y_1 + \gamma & \to & 3y_1 + \gamma & 2y_2 + \alpha + \gamma + y_1 & \to & \alpha + \gamma + y_1 + 3y_2 \end{array} \right\}$$

with the initial conditions $(\gamma, y_1, y_2)_{t=0} = (1, 1, 1/2)$. This system verifies $y_2 = \frac{x^\alpha}{1+x^\alpha}$ at steady state, and as a result constitutes a binary presence indicator: if $x \gg 1$, then $y_2 = 1$, and if $x \ll 1$, then $y_2 = 0$, the discrimination being all the greater when the value of α is significant. It should be noted that this value is given here by a fixed molecule concentration but could be more simplistically represented by means of constant kinetics.

Nonetheless, the downturn of this converter is that it creates an intermediate value in $\frac{1}{\gamma}$, which gives an exponential amplitude for $x = 0$, and thus an exponential computational complexity following the reasoning of the previous section. If we confine ourselves to take x in an interval of the form $[\varepsilon, +\infty[$, with $\varepsilon > 0$, then the complexity becomes polynomial. Moreover, if considering only the fourth order, our compiler with the command `compile(id^4/(1+id^4),x,o)` produces a system of 54 reactions, of polynomial algorithmic complexity, however, with a nonlinear computational complexity divergent component.

The 30-reaction MAPK natural circuit thus appears to be both more concise and having lower computational complexity (absence of divergence) than the reaction system produces currently following our generic principles of compilation.

19.5. Biochemical compilation of sequentiality and cell cycle

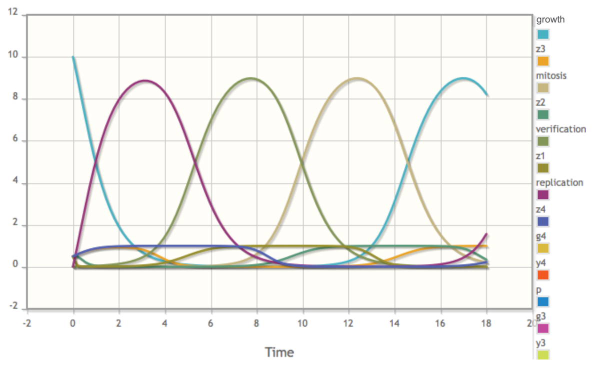
Similarly, it is possible to implement a binary absence indicator by implementing the term $\frac{c^\alpha}{c^\alpha + x^\alpha}$. This allows us to obtain a chemical implementation much better than those proposed in [SEN 11] or even [HUA 12], for which the leakage phenomena can occur: even in the relative absence of species x, the presence indicator remains in concentration sufficiently significant to catalyze certain reactions, or the opposite can be observed, the absence indicator may be in too low quantity. This is particularly visible in the implementation of sequentiality: given reactions R_i, if it is desired that R_2 be executed only when R_1 is complete, we impose an indicator of the absence of a species consumed by R_1 as a catalyst of R_2, and the same is done between R_2 and R_3. The following phenomenon can then be observed: reactions become much slower as i is large, in other words, reactions accumulate delay during their execution due to the retention of the absence indicators.

Provided with a strong enough absence indicator, it is possible to implement sequentiality, the conditional instruction as well as the loop structures of the algorithmic programming. Huang *et al.*[HUA 12] thus show how to compile small imperative programs inside a system of biochemical reactions in which molecular species are used as markers of the position of the program in a control flow graph.

For example, a minimalist specification of the cell division cycle can be output by the program

```
while true do {growth; replication; verification; mitosis;}
```

The compilation of this program into elementary reactions implements the sequentiality of the four phases of the cycle through the degradation of the markers of each of the phases, similarly to cyclin proteins in cell cycle models [GRA 09] :



19.6. Discussion

The bioinformatics vision, which is tantamount to regarding cells as machines and reactions as programs, is fertile in concepts and formal tools to understand the complexity of the networks of molecular interactions, to formally specify the biological behaviors observed or desired, to elucidate important interactions and their role in natural systems, to synthesize by means of the compilation of artificial biochemical programs and to compare them to natural circuits, not only in terms of structural complexity [BAR 6], but also, and this is new, in terms of computational complexity.

We have shown that the difficulty inherent in the analog nature of protein computations, as opposed to digital computations and discrete abstraction, that best suits to the activity of genes could assume a role at the fundamental level, through the results of equivalence between the notions of computability and complexity of analog and digital computational modes. In this direction, the analog characterization of low algorithmic complexity classes seems crucial to analyze and enforce the different ways to implement a function.

We therefore begin to understand biochemical reaction systems as programs, but not yet, for example why natural circuits have this structure and not others, what was their evolution and what are their possibilities for development. The specification language of analog computations on our biochemical compiler input can help but is still uncertain. We are able to infer a reaction system equivalent to a system of differential equations, to compile a GPAC in biochemical reactions, or a linear input–output system defined by its transfer function [CHI 15, OIS 11], and to combine these analog computations to logical conditions and control structures of conventional algorithmic programming, but these different specification languages are not unified. Although it is now possible to implant in living cells

[NIE 16, COU 15], or more safely inside non-living artificial vesicles [COU 17], synthetic biochemical programs limited to a few reactions satisfying a logical specification, the shift toward the scaling of less trivial programs requires progress of fundamental nature on reaction systems and their complexities.

In this undertaking for the decryption and reconstruction of the software of life, the next major step should be the systematic study of algorithmics of natural systems and their past or possible evolution, following the mathematical specification methods of their functions outlined here, and that can be used as part of a computational theory of evolvability [VAL 13].

19.7. Bibliography

AQ1 [BAR 6] Barabási A.-L., *Network Science*, Cambridge University Press, 2016.

[BER 92] Berry G., Boudol G., "The chemical abstract machine", *Theoretical Computer*
AQ2 *Science*, 1992.

[BOU 06] Bournez O., Campagnolo M.L., Graça D.S. *et al.*, "The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation", *International Conference on Theory and Applications of Models of Computation*, Springer, pp. 631–643, 2006.

[BUS 06] Busi N., Gorrieri R., "On the computational power of Brane Calculi", in Plotkin G. (ed.), *Transactions on Computational Systems Biology VI*, vol. 4220 of *Lecture Notes in BioInformatics*, Springer-Verlag, pp.16–43, 2006. CMSB'05 Special Issue.

[CAR 10] Cardelli L., Zavattaro L., "Turing universality of the biochemical ground form", *Mathematical Structures in Computer Science*, vol. 20, no. 1, pp. 45–73, 2010.

[CAR 05] Carothers D.C., Parker G.E., Sochacki J.S. *et al.*, "Some properties of solutions to polynomial systems of differential equations", *Electronic Journal of*
AQ3 *Differential Equations*, 2005.

[CHA 04] Chabrier-Rivier N., Chiaverini M., Danos V. *et al.*, "Modeling and querying biochemical interaction networks", *Theoretical Computer Science*, vol. 325, no.1, pp. 25–44, 2004.

[CHI 15] Chiu T.-Y., Chiang H.-J.K., Huang R.-Y. *et al.*, "Synthesizing configurable biochemical implementation of linear systems from their transfer function specifications", *PLoS ONE*, 2015.

[COU 17] Courbet A., Amar P., Fages F. *et al.*, "Computer aided design of programmable synthetic protocells performing multiplexed logic-gated micro-scale diagnostics", *In*
AQ4 *Preparation*, 2017.

[COU 15] Courbet A., Endy D., Renard E. *et al.*, "Detection of pathological biomarkers in Human clinical samples via amplifying genetic switches and logic gates", *Science Translational Medicine*, 2015.

[COU 77] Cousot P., Cousot R., "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints", *POPL'77:*

Proceedings of the 6th ACM Symposium on Principles of Programming Languages, ACM Press, New York, pp. 238–252, 1977. Los Angeles.

[FAG 15] FAGES F., GAY S., SOLIMAN S., "Inferring reaction systems from ordinary differential equations", *Theoretical Computer Science*, vol. 599, pp. 64–78, 2015.

[FAG 8a] FAGES F., SOLIMAN S., "Abstract interpretation and types for systems biology", *Theoretical Computer Science*, vol. 403, no. 1, pp. 52–70, 2008.

[FAG 08b] FAGES F., SOLIMAN S., "From reaction models to influence graphs and back: a theorem", *Proceedings of Formal Methods in Systems Biology FMSB'08*, number 5054 in *Lecture Notes in Computer Science*, Springer-Verlag, 2008.

[FAG 14] FAGES F., TRAYNARD P., "Temporal logic modeling of dynamical behaviors: first-order patterns and solvers", in DEL CERRO L.F., INOUE K. (eds), *Logical Modeling of Biological Systems*, John Wiley & Sons, Inc., chapter 8, pp. 291–323, 2014.

[GRA 09] GÉRARD C., GOLDBETER A., "Temporal self-organization of the cyclin/Cdk network driving the mammalian cell cycle", *Proceedings of the National Academy of Sciences*, vol. 106, no. 51, pp. 21643–21648, 2009.

[GIL 77] GILLESPIE D.T., "Exact stochastic simulation of coupled chemical reactions", *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[GRA 03] GRAÇA D., COSTA J., "Analog computers and recursive functions over the reals", *Journal of Complexity*, vol. 19, no. 5, pp. 644–664, 2003.

[HUA 96] HUANG C.-Y., FERRELL J.E., "Ultrasensitivity in the mitogen-activated protein kinase cascade", *Proceedings of the National Academy of Sciences*, vol. 93, no. 19, pp. 10078–10083, 1996.

[HUA 12] HUANG D.-A., JIANG J.-H., HUANG R.-Y. *et al.*, "Compiling program control flows into biochemical reactions", *ICCAD'12: IEEE/ACM International Conference on Computer-Aided Design*, ACM, San Jose, pp. 361–368, 2012.

[NIE 16] NIELSEN A.A.K., DER B.S., SHIN J. *et al.*, "Genetic circuit design automation", *Science*, 2016.

[OIS 11] OISHI K., KLAVINS E., "Biomolecular implementation of linear I/O systems", *IET SYstems Biology*, vol. 5, no. 4, pp. 252–260, 2011.

[POU 15] POULY A., Continuous models of computation: from computability to complexity, PhD Thesis, Ecole Polytechnique, 2015.

[QIA 07] QIAO L., NACHBAR R.B., KEVREKIDIS I.G. *et al.*, "Bistability and oscillations in the Huang-Ferrell model of MAPK signaling", *PLoS Computational Biology*, vol. 3, no. 9, pp. 1819–1826, 2007.

[RIZ 11] RIZK A., BATT G., FAGES F., *et al.*, "Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures", *Theoretical Computer Science*, vol. 412, no. 26, pp. 2827–2839, 2011.

[SEG 84] SEGEL L.A., *Modeling dynamic phenomena in molecular and cellular biology*, Cambridge University Press, 1984.

[SEN 11] SENUM P., RIEDEL M., "Rate-independent constructs for chemical computation", *PLOS One*, 2011.

[SHA 41] SHANNON C., "Mathematical theory of the differential analyser", *Journal of Mathematics and Physics*, vol. 20, pp. 337–354, 1941.

[SNOU 98] SNOUSSI E.H., "Necessary conditions for multistationarity and stable periodicity", *Journal of Biological Systems*, vol. 6, pp. 3–9, 1998.

[THO 81] THOMAS R., "On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations", *Springer Ser. Synergetics*, vol. 9, pp. 180–193, 1981.

[VAL 13] VALIANT L., *Probably Approximatively Correct*, Basic Books, 2013.