Stabilizing CRN

Mathieu Hemery

March 28, 2022

1 Context

Chemical Reaction Networks (CRN) are set of reactions involving chemical species. This is widely used framework to reason about interactions among large number of chemical or biological species.

In this article we will be mainly interested by the stable states of CRN, that is the fate of a CRN when it is let alone in a precise environment. The motivation for this work is the generation and compilation of CRN able to implement some kind of robust signal computation. The paradigmatic example would be the computation of a function based on the external concentration of some active chemicals. Think of a cell with A and B present in the environment. The ratio of volume between the cell and the environment is so large that the concentration of these species may be thought of as pinned inside the cell (provided that the transport is fast enough). Then, we would like to compute a function: $f(A, B)$ and encode it inside the cell as the concentration of an internal species, Y.

How can we build a CRN such that we ensure that, past a transition phase we have:

$$Y(t) \simeq f(A(t), B(t)).$$

For example, if we think of the lactose operon we might want to implement a signal function of the form:

$$\text{signal}(t) \simeq \frac{\text{lactose}(t)}{\text{glucose}(t) + \text{lactose}(t) + 1},$$

which will be high only if lactose is high and glucose is low.

For this study it would be tempting to look at a CRN only a set of polynomial equations. But we cannot discard the dynamics solely because we are only interested in the steady state, because we also want this state to be stable. We have to check not only that the system of polynomials is identically null, but also that the second derivatives are all negatives.

2 Definition

2.1 Chemical Reaction Networks

Chemical reaction networks (CRN) are a standard formalism in chemistry and biology. A CRN over a vector S of molecular species is a finite set of formal chemical reactions of the form:

$$r(S) \xrightarrow{f(S)} p(S) \tag{1}$$

composed of a multiset $r(S)$ of reactants (with multiplicity given by stoichiometrics coefficients in r), a multiset $p(S)$ of products and a rate function $f(S)$ on the quantities of reactants.

We will moreover supposed that all our reactions are described by the Mass Action Law kinetics, that is $f(S)$ is a monomial term with a positive rate and the concentration of each input species all raised to the power of ther stoechimetric coefficient. This hypothesis rely on a well stirred medium and concentration large enough so that discrete noise is not relevant.

Collision theory shows however that the probabilities of reactions involving three or more reactants are negligible. Hence from a mechanistic point of view, the restriction to reactions involving at most two reactant molecules is of practical importance. We call an elementary CRN (ECRN) a CRN with at most bimolecular reactions and mass action law kinetics. The restriction to at most bimolecular reactions leads to polynomial ODEs of degree at most 2.

In our case, we have to distinguished among the molecular species a subset of input species and a single output species (dealing with multiple output species is not problematic). When possible we will use the following convention:

- $X = \{x_i\} \quad i \in [1, m]$ is the set of inputs,
- y is the output species,
- $Z = \{z_i\} \quad j \in [1, n]$ is the set of intermediate species.

As seen, upper case letters designate vectors or set of species.

We want to study the case where the concentrations of the inputs are imposed by the environment (because they are pinned by an external concentration for example). In this case we have by definition $\forall i, \frac{dx_i}{dt} = 0$. This leads us to introduce the notion of pinned inputs:

Definition 1. *For a CRN with a set of distinguished species S^p we defined the ODE semantic with pinned species S^p by using the differential semantic for every species $s \notin S^p$ and pinning all the distinguished species, that is:*

$$\forall s \in S^p, \quad \frac{ds}{dt} \simeq 0. \tag{2}$$

And then our definition of stabilization:

Definition 2. *We say that a CRN over the set of $m+1+n$ species $\{X, y, Z\}$ with pinned inputs X of cardinality m and distinguished output y stabilize the function $f : \mathbb{R}_+^m \mapsto \mathbb{R}_+$ over the non null domain $\mathcal{D} \subset \mathbb{R}_+^{m+1+n}$, if we have:*

$$\forall (X, y, Z) \in \mathcal{D}, \lim_{t\to\infty} y(t) = f(X), \tag{3}$$

when following the ODE semantic with pinned input species X.

In this case we will also say that f is stabilized by a CRN and denotes $\mathcal{F}_s$ the set of function for which there exists a stabilizing CRN.

It could be sensible to impose that none of the species are unbounded or oscillate. But this is actually not necessary and wi will see that allowing oscillating part do not actually bring a more powerfull model of computation.

It is important to remark that while $\mathcal{D}$ is a domain over all the species, only the output and the intermediate species are dynamics under our ODE semantic (the concentrations of the inputs are supposed to be fixed), hence saying that y stabilize at $f(X)$ is sufficient as the value of X will not move during the experiment.

Intuitively, this represent the idea that $\mathcal{D}$ is the basin of attraction of a line of fixed points defined by $y = f(X)$ (or at least, that $\mathcal{D}$ is included in this basin). It would be nice if we could extend the domain $\mathcal{D}$ to the whole space but this is not possible in general.

We have asked for a non-null domain in the sense that $\mathcal{D}$ should not be reduced to an hyper surface of $\mathbb{R}^{m+1+n}$, this is a strong requirement and we should explain why we ask it and what it means. This hypothesis ensure that the fixed point is stable in all the direction and thus that there is no perturbation that drives the CRN in a wrong computation. Thus we are asking for a strong form of robustness to be sure that every "small" perturbations will be smooth down by the dynamic. At a more formal level, this means that on the fixed point, the system is not degenerate.

To gives a counter example, we may examinate the system:

$$\frac{dx}{dt} = 0 \tag{4}$$

$$\frac{dz_1}{dt} = z_1 - x \qquad z_1(t=0) = 0 \tag{5}$$

$$\frac{dz_2}{dt} = -y(z_1 - x) \qquad z_2(t=0) = 1 \tag{6}$$

$$\frac{dy}{dt} = z_2(z_1 - x) \qquad y(t=0) = 0 \tag{7}$$

that computes $y = sin(x)$, but this is done in a way where any perturbation of the variable z_2 or y will not be eliminate. In this sense, the desired function is not the only fixed point. Worse, there is infinitely many fixed point even for arbitrary small perturbations. We may also see this as encoding information in the initial state, a feature that we don't want.

As the right hand side of the equation is a polynomial and thus highly regular, the Picard-Lindelöf theorem ensure that the solution of the PIVP exist and is unique up to the first divergence. As we have asked that y has a limit at $t \to \infty$ the unicity is automatic in our formalism.

2.2 Algebraic functions

We also remind the definition of the set of algebraic functions:

Definition 3. $f : D \subset \mathbb{R}^n \mapsto \mathbb{R}$ *is said to be algebraic if there exists a polynomial* P *of* $n+1$ *variables such that:*

$$\forall X \in D, P(X, f(X)) = 0.$$

We will note $\mathcal{F}_A$ *the set of algebraic functions.*

Remark that this is far bigger than the set of algebraic expressions. (The set of functions obtain by composition of $+, -, \times, \div$ and the integer power and square root.) Actually some algebraic functions like the Bring radical (the unique real solution of the equation $y^5 + y + x = 0$) have no explicit expression.

The purpose of algebraic geometry is to study the curve defined by such polynomial and we usually assimilate a curve and its polynomial. Hence we can speak of the unit circle: $x^2 + y^2 - 1$. For this correspondance to be exact, we have to work in an algebraic close field (to be sure to *see* all the solutions, some of them can be complex) and to precise the multiplicity of the curve: $\left(x^2 + y^2 - 1\right)^2$ is the unit circle with multiplicity 2. To simplify our discourse and as we are mainly interested by the curve and the functions it defines, we will suppose that the curves are given with multiplicity one.

Given a polynomial of degree k with respect to the output y (that is the higher degree of y in the polynomial is k), for a given value of the vector X, we may expect to recover up to k different values for y corresponding to the k roots of the polynomial (these solutions may be complex). Once we have chosen one of this value, we may follow the algebraic curve defined by $P(X, y) = 0$ in each direction until we reach a monodromy point, that is a point where the curve cross itself or get back on his step and the function is thus no more unique. The whole part of a curve where we may obtain a unique solution is call *branche*. And – except for critical points – the existence of this function is ensured by the Implicit function theorem. The implicit function theorem ensure that this is a correct definition and that the function define this way exists and is unique.

In the reminder of the paper, we will be interested to compile function implicitely defined through algebraic equation and we will abuse of notation by calling a *function* the branch of the curve that goes from one singularity to another. In this case, the function is defined by giving the polynomial P and a point (X, y) used to select the branch we want to recover.

This also explain why in definition 2 we cannot expect to extand the domain $\mathcal{D}$ to the whole space. Intuitively, the polynomial may have other branches, hence defining other bassin of attraction. See figure 1 for a simple case.

Theorem 1. *The set of functions stabilized by an ECRN with the ODE semantic and Mass Action Law is the set of algebraic functions:*

$$\mathcal{F}_S = \mathcal{F}_A. \tag{8}$$

The proof of the theorem is given in section 3.

2.2.1 From $f(x)$ to P

For convenience, we implement a pre-compilation step that allows the user to use algebraic expressions in his definition of the function: `stabilize_expression(y-sqrt(x+1), y, [y=1, x=0])`

Extracting the curve from such an expression is not *a priori* easy. And may rely on complex algebraic transformations such as elimination and gröbner basis. As we do not need to boil down to a single polynomial, we can however for our purpose keep the auxiliary variables along the main expression and "compile" as a last step the set of polynomial relation instead of a single one. Computing these variables is done the same way we would compute the desired output.

Nonetheless, dealing with auxiliary variables this way may lead to strong inneficiency as some simple rewritting may sometime be enough to remove one of this auxiliary variables.

We rely on a previous algorithm to detect all non-polynomial terms in the expression and add a new polynomial to compute them while rewritting the first. For a non polynomiality of the form: $\sqrt[n]{e}$ you introduce a new variable v along with the relation: $v^n - e = 0$. And for a non-polynomiality of the form $\frac{1}{e}$, you have $ev - 1 = 0$. These are the only expressions support for now.

The clearest presentation being an example, let us suppose we want to compute the function:

$$y = \frac{2x - 1 + \sqrt{1 + 4x^2}}{2x}$$

Example: In this case, our algorithm detect two non-polynomial term: $\frac{1}{2x}$ and $\sqrt{1+4x^2}$. So we have two new relations: $2xa = 1$ and $b^2 = 1 + 4x^2$ and the rewritting of the first one: $y = a * (2x - 1 + b)$ giving the PIVP:

$$\frac{dy}{dt} = 2ax - a + ba - y$$
$$\frac{da}{dt} = 1 - 2xa$$
$$\frac{db}{dt} = 1 + 4x^2 - b^2$$

3 Proof

There is two part in the proof that may be attacked separately. First of all the very link between CRN and an algebraic branche may be proven quite simply. In a second time, the study of the domain $\mathcal{D}$ under which we have some guaranty of convergence have to be tackled.

3.1 Existence

We will start by prooving that if a CRN stabilize a function, then this function is algebraic.

To provide the intuition, let us start by studying the case where the whole ODE stabilize on a single fixed point. Thus at equilibrium, all the derivatives of the ODE semantics are polynomial and null. We have a set of polynomial equation and we can thus eliminate the Z variables in order to obtain a single polynomial on y and X.

This is possible only if the system is not degenerate, but this is the case because we ask for a domain $\mathcal{D}$ of non null measure. Thus proving that the function $y = f(X)$ is algebraic.

If we want to define completely the computed function, we also have to select the branche by indicating a point of f, the current stable state will be enough for this.

(If we fix X, and choose a point that is stable, then if the set of equation is degenerated, we do not have a unique fixed point but a degenerated set of fixed points and thus you are dependant upon your initial conditions meaning that there exist a perturbation of y, Z at $t = 0$ such that you do not converged to the desired $f(X)$.)

A more complex case would be the case where some of the variables are unbounded or oscillate. In this case we cannot rely on the polynomial system to construct our polynomial. However, in [?] Carothers & al. give a caracterization of the functions solution of a PODE: f is solution of a PODE if and only if there exist a polynomial Q of $k+1$ variables such that:

$$Q\left(f, f^{(1)}, \ldots, f^{(k)}\right) = 0.$$

For our proof, the idea is to consider that y, X is the solution of the PODE and use the theorem to eliminate the Z variables. This gives us a polynomial Q upon X and the k first derivatives of y that defines our solution. Note that the derivatives of X are null and thus do not appear in the polynomial. Now we take the limit when $t \to \infty$ and by assumption, we have all the derivatives of y that vanishe, leaving only a polynomial in y and X as desired.

It is possible that this polynomial is degenerate when taking the limit $t \to \infty$ for example in the case of the CRN that "computes" the cosine given in equations 4 and following, where the species z_1 will converge to x and halt the couple fo reactions governing y and z_2 so that it halts on the desired value. We can compute a polynomial upon the three first derivative giving something like (up to my computation errors):

$$yy^{(3)} = {y^{(1)}}^2 + y^{(1)}y^{(2)} - 2yy^{(1)} - 3yy^{(2)}$$

that on the limit reduce to $0 = 0$. But this is not a surprise as this fixed point is not isolated and that we can actually find any value of y that stabilize the PODE for any value of x. The computation rely on the intial state to provide an acurate computation something that our model prohibit. So this CRN doesn't compute in our model and it is not surprising that we cannot find the desired polynomial.

Now let us take an algebraic function f and denotes P its «simplest» generating polynomial and (X^0, y^0) a point on the branch we want to stabilize. By simplest here, we will ask that the branch of interest is irreducible and have multiplicity 1. The first requirement (irreducibility) is mainly for convenience while the second (multiplicity 1) is essential, otherwise it would not be possible to stabilize the desired branch as the polynomial would not change its sign across the graph of f, a crucial property for the stability. Nonetheless, it is neither a restriction on the set of function as the graph V is a good caracterization of the polynomial up to the multiplicity. As we are interested only in the algebraic curve (or more precisely function) we can simply set all the multiplicity to 1. This correspond to the polynomial of lowest possible degree.

First, we must ensure that we only deal with positive variable as the concentration of a molecular species is always positive. This can be corrected by a dual rail encoding: replacing the troubling variable X by its positive and negative parts: $X = X^+ - X^-$ and adding to the polynomial a term of the form $(X^- X^+)^2$ that ensure that one of the two variables will be null. In terms of CRN, this is equivalent to adding a bidegradation: $X^+ + X^- \to \emptyset$.

We now have to check that we are able to construct an ECRN that implement this polynomial. For this we will show that we are able to add and substract monomials and form products to construct these monomials, eventually by introducing new species as intermediary variables.

To construct a species that stabilize on a monomial $M = r\prod_i X_i$, we have to separate the cases according to the degree of M. If M is of degree 0, then this is simply a basal production: $\emptyset \to M$. If M is of degree 1: $X_1 \to M$ and if M is of degree 2: $X_1 + X_2 \to M$. In all these cases, the kinetic rate of the MAL is r and we also need to add an auto degradation $M \to \emptyset$.

Now if M is of order $n > 2$, we have to split M among two intermediary variables N_1 and N_2 of order strictly smaller than n. When this is done, then $M = rN_1N_2$ may be construct as an order 2 monomial as described above.

It is important to note that outside of the reactions that define them, the species only acts as modifiers and their concentration is thus unaffected by the other reactions.

To construct a species that stabilizes on the sum of several monomial, we simply have to add a catalyzed production for each of these monomials and a single auto-degradation. Hence $S = \sum_i M_i$ gives: $M_i \to M_i + S$ for each i and $S \to \emptyset$. To substracts element in a sum, you simply add them to the negative part of the species!

To end up the proof, we have to show that we can gather every element in the derivative of the output positive and negative parts in a way that ensure that the desired branch is stable. To gather all the element of the polynomial, we start with the two bare derivatives: $\frac{dY^+}{dt} = -\text{fast}Y^+Y^- \quad \frac{dY^-}{dt} = -\text{fast}Y^+Y^-$, and add the different monomials one by one, there is three cases to consider:

1. If the monomial is positive: we simply add it to the derivative of Y^+, it will correspond to a synthesis term,
2. if the monomial is negative and contain Y^+: we can still add it to the derivative of Y^+ as it will be a (eventually catalyzed) degradation,
3. otherwise, we switch its sign and add it to the derivative of Y^-, it will correspond to a production of the negative part.

The last step is to ensure that the branch of the algebraic curve corresponding to our desired function is stable. This correponds to choosing between the $\frac{dy}{dt} = P$ and $\frac{dy}{dt} = -P$ implementation, and this is easily checked by looking at the second derivative of y at the point (X^0, y^0) and choosing the sign such that this second derivative is negative, hence enforcing the stability of the desired branch.

And this is actually sufficient to prove the theorem. The main problem is that we have no idea of the domain D over which the CRN will stabilize f other than that it covers a non empty neighbourhood of each point of f. (This is obtain by the implicit function theorem.) To obtain a more complete evaluation of D we have to look at the constructed CRN in details.

3.2 Stability

Very intuitively, we expect the CRN to be stable for the whole domain that is delimitate on its side by the monodromy points/curve and above/under by the other branch of the algebraic curve. Indeed, the flow for the output variable will change its sign when crossing the curve as shown in figure 1 and we cannot ensure what goes on when crossing the monodromy boundary as those are points where the implicit function theorem fail, we thus cannot even define our function passed these points.

On our formal level, the stability of our CRN is the same as the one of the PIVP, for a given generating polynomial $P(X, y)$ we will note Z the auxiliary variables introduce during the compilation to obtain a quadrativ PODE (that is an elementary CRN). The resulting PODE is of the form:

$$\frac{dy}{dt} = sP'(X, y, Z_j) \tag{9}$$

$$\frac{dz_j}{dt} = f_j(X, y, Z) - z_j \tag{10}$$

where $s \in \{-1, +1\}$, P' is the rewriting of P that uses the new variables and, importantly, f_j defines z_j and does not depand on it.

To examinate the stability of this differential equation, we will use the notion of Lyapunov stability and pose:

$$V(X, y, Z) = P'(X, y, Z)^2 + \sum_j \left(f_j(X, y, Z) - z_j\right)^2. \tag{11}$$

Obviously, the function V is positive definite and equal 0 when the variables of Z equals their definition and we are on a root of P'. Importantly, when the variables Z are well defined we have $P'(X, y, Z) = P(X, y)$, and the roots of P' are those of P. So V is null on the roots of P.

Now we want to look at the derivative of V:

$$\frac{d}{dt}V = 2\frac{d}{dy}P'(X,y,Z)sP'(X,y,Z)^2 \tag{12}$$

$$+2\frac{d}{dz_j}P'(X,y,Z)\left(f_j(X,y,Z)-z_j\right) \tag{13}$$

$$+\sum_j 2\frac{df_j}{dy}\left(f_j(X,y,Z)-z_j\right)sP'(X,y,Z) \tag{14}$$

$$+\sum_j 2\frac{df_j}{dz_j}\left(f_j(X,y,Z)-z_j\right)^2. \tag{15}$$

The last line is equally null because as f_j do not rely on z_j we have $\frac{df_j}{dz_j} = 0$. If we suppose we are near a stable point, we have $f_j(X, y, Z) \simeq z_j$ which cancel the second and third lines so that we are only interested in the sign of $s\frac{dP'(X,y,Z)}{dy} = s\frac{dP(X,y)}{dy}$. And we can choose s so that this quantity is negative over the branch of interest. Then by the Lyapunov theorem, V is a definite positive function that cancel on our point and its derivative is negative around that point making it a stable equilibrium point, at least locally.

This also indicates that we cannot hope for a larger bassin of attraction than the larger interval containing our solution and for which $\frac{dP}{dy}$ keeps a constant sign. Moreover, the inspection of the second and third lines indicates the potential troublemakers: $\frac{dP'}{dz_j}$ and $\frac{df_j}{dy}$.

4 Examples

We present in a first example how we can go from the CRN to the algebraic curve. And then several examples of our compilation pipeline.

4.1 Michaelis-Menten kinetics

As a first example, let us look how we can go from a biological CRN to the corresponding algebraic curve.

We denote I the inactive form of the species and A its active form, the enzyme E is the forced input the concentration of which is invariant. This gives

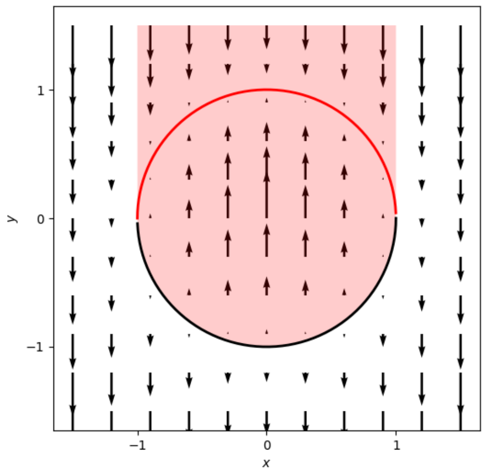


Figure 1: Flow field for the compilation of the algebraic curve $X^2 + Y^2 - 1$ defining the circle. The stabilized branch being the upper one (red), we can see that for a point inside the domain $\mathcal{D}$ in red, the flow has a single fixed point on which the polynomial is null. Outside the domain, there is no fixed point and in this case the output diverges (exponentially) to $-\infty$.

us the following ECRN:

$$I + E \to A + E \tag{16}$$
$$A \to I \tag{17}$$
$$\_ \to I \tag{18}$$
$$I \to \_ \tag{19}$$
$$A \to \_ \tag{20}$$

Notes that as in our setting, the concentration of the species E is "pinned" is this model as it only acts as an enzyme. To avoid a dependency to the initial condition, we also add the production and degradation of the species.

We have the following PODE:

$$\frac{dI}{dt} = 1 - IE + A - I (= 0) \tag{21}$$
$$\frac{dA}{dt} = IE - A - A (= 0) \tag{22}$$

Adding the two equations gives us the total concentration of the species: $I + A = 1$. Using this relation, we thus have the algebraic curve $P(E, A) = E - AE - 2A$. This gives us a michaelis menten like function: $A(E) = \frac{E}{2+E}$ and testify that it is algebraic.

4.2 Dual-rail encoding

To start the presentation of our work in the reverse direction we check that we are able to compute negative quantity using our dual-rail encoding.

In all of this examples, we will stick to our convention and denote the input(s) by x_i (or simply X if there is only one) and Y will be our output while auxiliary species are labelled az z_i.

We look at the polynomial $P(X_1, X_2, Y) = Y - (X_1 - X_2)$ as a typical case where the dual rail encoding is needed. Let us suppose that both inputs are positive to avoid cluttering our notation.

We begin by splitting the output variable: $Y = Y^+ - Y^-$. Then, it is a simple sum where negative terms ends up in the positive variable and vice-versa:

```
MA(fast) for y_m+y_p=>_.
MA(1.0) for x1=>x1+y_p.
MA(1.0) for y_m=>y_m+y_p.
MA(1.0) for x2=>x2+y_m.
MA(1.0) for y_p=>y_m+y_p.
```

in terms of ODE we have:

$$\frac{dY^+}{dt} = X_1 + Y^- - \text{fast}Y^+Y^- (= 0) \tag{23}$$
$$\frac{dY^-}{dt} = X_2 + Y^+ - \text{fast}Y^+Y^- (= 0) \tag{24}$$

and we can check that at equilibrium, the second line minus the first gives us our desired polynomial definition.

Note that we do not have to impose that the bi-degradation is faster than the other reactions as it do not disturb our main result, namely that Y will be a solution of our polynomial. In practice, the faster it is, the smaller is the concentration of the species that should be null (positive or negative parts of Y) and hence the easier it is to read the result. But our result holds whatever the (non-null) value of the parameter fast.

4.3 Third root

Let us start with the computation of the third root, the polynomial is simply: $P(x, y) = x - y^3$. But we have a monomial of order 3. We split in two and introduce $z = y^2$. Hence our PODE looks like:

$$\frac{dx}{dt} = 0 \tag{25}$$

$$\frac{dy}{dt} = x - yz \tag{26}$$

$$\frac{dz}{dt} = y^2 - z \tag{27}$$

Once given to our compiler, this ODE became the folowing CRN, we only reorder the reactions so that they appear in the same order as the monomial of the previous PODE and rename the variable `y2` into `z` to follow the convention of this article.

```
MA(1.0) for x=>x+y.
MA(1.0) for y+z=>z.
MA(1.0) for 2*y=>z+2*y.
MA(1.0) for z=>_.
```

4.4 Bring Radical

We include this part as the Bring radical is a well-known algebraic function for which there is no known explicit algebraic expression[1] and is thus a good example of an interesting algebraic function.

The corresponding polynomial is: $P(X, Y) = Y^5 + Y - X$. (once again we modify the name of the introduces species to follo the convention of this article, $z_1 = y^2$ and $z_2 = y^3$.)

```
biocham: stabilize_expression(y^5+y-x, y, [x=2, y=1]).
biocham: list_model.
MA(fast) for y_m+y_p=>_.
MA(fast) for z1_m+z1_p=>_.
```

[1]That is there is no expression of the form $f(x) = \ldots$, for convenience we switch the sign of the Bring radical (which is generally negative) to encode it as a concentration.

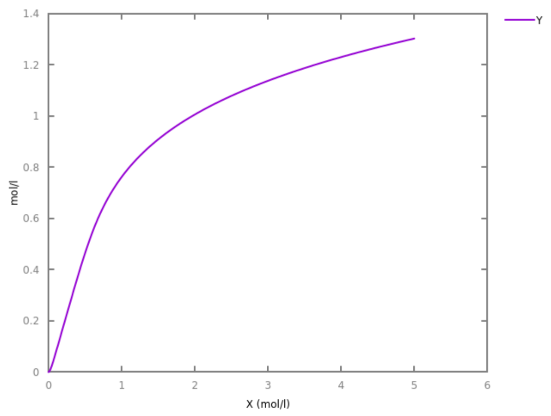


Figure 2: Dose response diagram of the CRN corresponding to the Bring radical. The concentration of X scan the range from 0 to 5 and the concentration of Y thus follow the curve of the Bring radical function.

```
MA(fast) for z2_m+z2_p=>_.
MA(1.0) for x=>x+y_p.
MA(1.0) for y_m=>y_m+y_p.
MA(1.0) for z1_m+z2_p=>z1_m+z2_p+y_p.
MA(1.0) for z1_p+z2_m=>z1_p+z2_m+y_p.
MA(1.0) for y_p=>y_m+y_p.
MA(1.0) for z1_p+z2_p=>z1_p+z2_p+y_m.
MA(1.0) for z1_m+z2_m=>z1_m+z2_m+y_m.
MA(1.0) for 2*y_p=>z1_p+2*y_p.
MA(1.0) for z1_m=>z1_m+z1_p.
MA(1.0) for 2*y_m=>z1_m+2*y_m.
MA(1.0) for z1_p=>z1_m+z1_p.
MA(1.0) for z1_p+y_p=>z1_p+z2_p+y_p.
MA(1.0) for z1_m+y_m=>z1_m+z2_p+y_m.
MA(1.0) for z2_m=>z2_m+z2_p.
MA(1.0) for z1_p+y_m=>z1_p+z2_m+y_m.
MA(1.0) for z1_m+y_p=>z1_m+z2_m+y_p.
MA(1.0) for z2_p=>z2_m+z2_p.
```

and we obtain the graph of this function as a dose-resonse diagram shown in figure 2.

Note that the quadratization step, while optimal at the quadratization step, leads to ineficiency during the negation step as the choice of encoding y^5 as $y^2 \times y^3$ make the degradation of y harder to implement which leads to the splitting of all the variable into positive and negative counterparts. It would be better to implement the two step in once but this is far from trivial to do.

4.5 Circle

A circle is defined by the polynomial equation: $P(X, Y) = X^2 + Y^2 - 1 = 0$. We choose to study in detail the circle as this is a simple case where the existence of several solution for the same input make obvious the process where we have to actively select the branch we want to stabilize as shown in figure 1. And detailed below.

If we consider Y as the output, it is tempting to use the PODE:

$$\frac{dY}{dt} = 1 - X^2 - Y^2,$$

but in this case, we have a negative monomial that does not depend of Y in its own derivative and the resulting ODE is ill-formed in biological term. We thus have to split the variable Y between its positive and negative term giving the polynomial:

$$1 - X^2 - {Y^+}^2 + 2Y^+Y^- - {Y^-}^2 = 0$$

and thus the following PODE:

$$\frac{dY^+}{dt} = 1 - {Y^+}^2 - \text{fast}Y^-Y^+ \tag{28}$$

$$\frac{dY^-}{dt} = {Y^-}^2 + X^2 - 2Y^-Y^+ - \text{fast}Y^-Y^+ \tag{29}$$

that effectively gives us the curve of a circle when we look at the dose-response diagram of the CRN as shown in figure 3.

But this is only half the circle, how to obtain the lower half? We simply have to start from the opposite polynomial:

$$-1 + X^2 + {Y^+}^2 - 2Y^+Y^- + {Y^-}^2 = 0$$

that gives this new PODE:

$$\frac{dY^+}{dt} = {Y^-}^2 + X^2 + {Y^+}^2 - 2Y^-Y^+ - \text{fast}Y^-Y^+ \tag{30}$$

$$\frac{dY^-}{dt} = 1 - \text{fast}Y^-Y^+ \tag{31}$$

5 Implementation

All this has been implemented in Biocham in the `stable_crn.pl` module. For technical reason the transformation from an algebraic relation into a CRN in slightly different than the one presented in the mathematical proof.

The core function is `stabilize_expression(+Expression, +Output, +Point)` that directly construct the CRN into the Biocham language. The `Expression` may be any algebraic expression between any number of variables (parameters

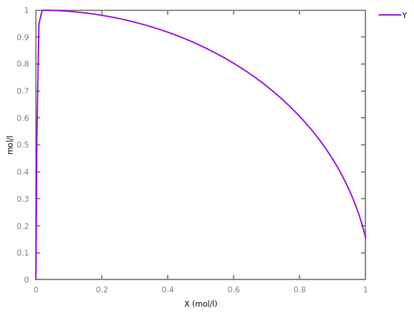
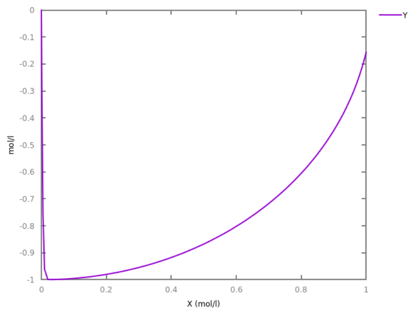
A.  **B.** 

Figure 3: Dose response diagramm of these two CRN when the concentration of X goes from 0 to 1, we can see that the value of $Y = Y^+ - Y^-$ follow the unit circle. Pannel A correspond to the PODE/CRN discuss in the beginning of the main text and pannel B correspond to the second PODE/CRN. Of course, these CRN are unstable as soon as $X > 1$.

are not supported yet). `Output` indicates the name of the Output among the variable present in the expression and `Point` is a list of value indicating the branch of the curve that we want to stabilize.

To construct the ECRN starting from the expression, we follow the following step:

- polynomialization
- quadratization
- lazy_negation and compilation

The polynomialization and quadratization are slightly different from the ones used for `gpac.pl` as the introduction of new variables do not follow the same law. In this case, we want the new variables to stabilize on their values once the inputs are pinned and not to compute it. Typically, if x is an input (hence $\frac{dx}{dt} = 0$) and we want to compute x^3 we cannot simply say that $\frac{dx^3}{dt} = 3x^2 \frac{dx}{dt} = 0$ as this will never stabilize. We have to introduce the new variable $x3 \leftarrow x^3$ and a derivative of the form: $\frac{dx3}{dt} = x2 * x - x3$. In this case we also have to introduce $x2 \leftarrow x^2$ in order to compute x^3, this may not have been the case for the quadratization of gpac.

To polynomialize the algebraic expression, we introduce new variables that will enforce relations such that we obtain a set of polynomial relations at the end.

Example – To obtain a hill function we ask for the relation: $y = \frac{x}{1+x}$ and the output y. The denominator is introduce as a new variable: $A \leftarrow \frac{1}{1+x}$ and thus the relation: $(1 + x)A = 1$. To enforce this relation, we choose the dynamics of the species A such that: $\frac{dA}{dt} = 1 - A - xA$ And this will lead to the set of reactions:

$$\begin{aligned} \emptyset &\to A \\ A &\to \emptyset \\ A + x &\to x \end{aligned}$$

and to stabilize y we add the CRN corresponding to $\frac{dy}{dt} = Ax - y$:

$$\begin{aligned} y &\to \emptyset \\ A + x &\to A + x + y \end{aligned}$$

we can concatenate the two reactions that have the same reaction rate to obtain:

$$\begin{aligned} \emptyset &\to A \\ A &\to \emptyset \\ y &\to \emptyset \\ A + x &\to x + y \end{aligned}$$

5.1 Quadratization

The idea of the quadratization is to "hack" the existing one in order to avoid duplicating code. But we have this question that we have to avoid using a variable in ints own computation (something that is not a problem in the traditional problem. For this, we introduce a synthetic variable at the end of each monomial that has exponent 1 if the variable is computed and 2 for those we have to compute.

6 Link with other work

6.1 Online computation

The link with the notion of online computable functions as defined in the thesis of A. Pouly is important to discuss.

We give back the definition as stated in the thesis: f is $\Upsilon, \Omega, \Lambda$-online computable (all these variables corresponding to functions of $\mathbb{R}_+^2$ in $\mathbb{R}_+$) if there exists a positive real δ, a polynomial $p : \mathbb{R}^{d+n} \to \mathbb{R}^d$ and an initial condition $y_0 \in \mathbb{R}^d$ such that for all inputs $x : \mathbb{R}_+ \to \mathbb{R}^n$ seen here as functions of time we have:

- there exist a unique solution y such that $y(0) = y_0$ and $y'(t) = p\left(y(t), x(t)\right)$
- the solution is bounded: $|y(t)| < \Upsilon(|x(t)|, t)$
- for two positive real a, b, if we have $|x(t) - \overline{x}| \leq e{-}\Lambda(|\overline{x}, \mu|)$ for $t \in [a, b]$ then it follows that $|y_{1,m}(t) - f(\overline{x})| \leq e^{-\mu}$ for t in $[a + \Omega(|\overline{x}, \mu|), b]$.

which we could describe as:

- There is a PIVP with variable parameters x for which y is the unique solution
- The solution is bounded by a function of x **and** t**.**
- If the input is stable around a value $\overline{x}$ with a given precision $\Lambda(\overline{x}, \mu)$ then after a given time $\Omega(\overline{x}, \mu)$, the output stabilize around $f(\overline{x})$ with a precision μ, until the input move again.

So there is several differences with the definition of stabilization:

- We have chosen to study scalar function for simplicity, apart from technical details, this is not an important difference.
- The notion of online computation rely on a unique initial condition y_0 while the definition of stabilized function is interested upon a domain of validity $\mathcal{D}$. As noted above, this requirement for a domain $\mathcal{D}$ forbid to encode information in the initial condition of the PIVP.
- As the bound on the species may grow as a function of time in the definition of online-computable function, while the output is obviously bounded, this allows for arbitrary long computation on the secondary species Z that allows a huge gain in computational power.
- Now the most important difference is that the stabilize definition simply asked for a simple (and not controlled) convergence behaviour for a fixed input while the online computation asked that when the input fluctuate near a certain value, the output fluctuate near the desired value after a known time. This is a strict requirement as this impose the PIVP to be able to filter fast variations of the input.

At the end, the model of online-computation rely on the simulation of a Turing machine that is preogressively driven to compute the desired output with more and more precision. In our case, the idea is really to enforce the attraction of the output to a particular value that is controlled by the input species.

This comparison however indicates the crucial part of our definition: the absence of divergence on any species limit the amount of computation and the robustness to initial conditions of the Z species hidden in the definition of our domain $\mathcal{D}$ precludes us to rely on some information hidden in the initial state.

6.2 Control theory

An interesting point of view upon the study of our CRN may come from control theory. In control theory, you usually have a black box function that depand upon a command variable u and an external (and possibly unknwon) function x: $f(u, x)$. The objectif is to keep this function as close as possible to a determine value: $\overline{f}$, and for this you compute the error $\epsilon = f(u, x) - \overline{f}$ and fed it to a controler $c(\epsilon)$ that will prescribed the value of u. So at the end you hope for something like:

$$\forall x(t), \quad f(c(\epsilon), x(t)) \approx \overline{f}. \tag{32}$$

Note that f and c may incorporate memory effect. The most used controller do in general incorporate a memory. The widely used PID controler is defined as:

$$c(\epsilon) = K_P \epsilon + K_I \int_0^t \epsilon(\tau) d\tau + K_D \frac{d\epsilon}{dt} \tag{33}$$

where K_P, K_I and K_D are three real constants that needed to be tuned to the desired problem.

In our case, we have a desired output $g(x)$ and an output y but we are in a case were we actually can construct the "black-box" function f so that we enforce the deisred behavior. Is there still a link between the two approach?

7 Discussion

Can we have a way of generating the polynomial from the function?