# A Selective Dual-Railing Technique for General-Purpose Analog Computers

Nicholas Haisler[2], Xiang Huang[1], Andrei Migunov[2(✉)], Khalid Mohammed[2], and Garrett Provence[2]

[1] Department of Computer Science, University of Illinois Springfield, Springfield, IL, USA
xhuan5@uis.edu

[2] Department of Mathematics and Computer Science, Drake University, Des Moines, IA, USA
{nicholas.haisler,andrei.migunov,khalid.mohammed, garrett.provence}@drake.edu

**Abstract.** This paper describes an efficient variation of a well-known technique for converting a *general-purpose analog computer* (GPAC) to a *chemical reaction network* (CRN). If an input system has $n$ variables, the existing *dual-railing* technique requires $2n$ variables in the resulting system. As the resulting CRNs are often fed into downstream processes which may themselves be exponential in the number of variables [7], it is frequently worthwhile to try to minimize the number of variables generated in this conversion. This paper presents a technique that results in fewer than $n$ new variables, by rewriting only those whose positive terms are 'infected' by ill-formed variables.

## 1 Introduction

Chemical Reaction Networks (CRNs) are a popular programming language for nanotechnological applications, allowing one to describe computation as a process of interactions between abstract molecular species. Moreover, CRNs have a natural *hardware language* in which they can be physically realized at the nano-scale: DNA Strand Displacement (DSD) cascades [11].

The ease of use of a 'high-level' language may depend on how comfortably one can manipulate and modify a program in order to change its behavior in an intended way. While the prospect of one day compiling high-level digital languages to CRNs or DSDs is also interesting [15], it is common and sometimes more convenient to convert other *analog* languages to CRNs.

With this in mind, ordinary differential equations (ODEs) are a well-studied and widely used model (see any standard text, e.g. [13]). In the world of analog computation, ODEs are the basis of the *general purpose analog computer* (GPAC) model, originally devised by Shannon [3,10]. Both GPACs and CRNs are Turing-complete [1,4] but the GPAC comes with the many conveniences of ODEs: tricks, substitution techniques, and many centuries of examples, in particular of the computation of numbers and functions.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
E. Formenti and L. Manzoni (Eds.): UCNC 2025, LNCS 16364, pp. 397–402, 2026.
https://doi.org/10.1007/978-3-032-15641-9_26

In this sense, we find it useful to regard CRNs as GPACs: one can express any chemical system in terms of its *mass action* kinetics, in which the concentration of each molecular species is tracked by the solution of a system of ODEs. If we want to manipulate CRNs, we might begin with a GPAC, program it to do what we want (for example, to compute $\sqrt{2}$ as the limit of one of its component functions), and then convert it to a CRN while preserving this property.

Not every GPAC is by default a CRN, as GPACs can have negative values and there cannot be negative quantities of chemical species. Moreover, in this paper we are concerned specifically with *bounded* systems whose variable values always lie within some interval $(-\beta, \beta)$ for $\beta > 0$. This is a realistic restriction that ensures that quantities do not 'overflow' to infinity, and that the time complexity of terminating computations can be meaningfully measured [2].

Section 2 reviews some formal definitions. In Sect. 3, we explore the *dual-railing* method, which transforms a GPAC into a CRN by encoding each variable as a difference of non-negative variables. In Sect. 4, we discuss a more efficient version of this method and demonstrate the improvement with an example.

## 2 Preliminaries

**Definition 1.** *A* General Purpose Analog Computer (GPAC) *is a system of ordinary differential equations of the form* $\mathbf{y}'(t) = \mathbf{p}(\mathbf{y}(t))$*, together with a set of initial values* $\mathbf{y}(\mathbf{0})$*, where* $\mathbf{p}$ *is a vector of polynomials. In this sense, a GPAC is a* polynomial initial value problem. *Such systems are always polynomial, autonomous, and first-order.*

**Definition 2.** *A* Chemical Reaction Network (CRN) *is a pair* $(S, R)$ *where* $S$ *is a set of (usually, capitalized and single-letter) names of molecular species, and* $R$ *is a set of reactions in which those species participate. Each reaction* $\rho$ *has the following form, where* $k_\rho \in \mathbb{N}$ *is its* rate constant*:*

$$\rho : a_1 X_1 + .... + a_n X_n \xrightarrow{k_\rho} b_1 Y_1 + .... + b_m Y_m.$$

**Theorem 1. (CRN-implementable Functions [5], Theorem 3.1 and 3.2).** *A function is CRN-implementable if and only if it is GPAC-implementable with the further restriction that the derivative of each component* $x_i$ *in its variable vector* $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ *is of the form*

$$x_i' = p - qx_i, \quad \text{where } p,\ q \text{ are positive polynomials.} \tag{1}$$

## 3 Naive Dual-Railing

Here we describe the well-known dual-railing method for converting a GPAC into a CRN. To *dual-rail a variable* $x$ is to split it into two variables, $x^+$ and $x^-$, ensuring that $x(t) = x^+(t) - x^-(t)$ for all $t \in [0, \infty)$, and that both $x^+$ and $x^-$ are CRN-implementable. The dual-railing method is really a collection of

related approaches that use this idea, and in examining a dual-railing approach we consider two factors: how many variables of a system are modified, and in what way are these modified. In our sense, a *naive* approach modifies every single variable even if, as we will show in the next section, this is not always necessary.

The other factor concerns the specific way of defining the new variables. One folkloric and common [9,14] option is to use *fast annihilation reactions*: as dual species $X^+$ and $X^-$ accumulate, their excess can be shed via $X^+ + X^- \xrightarrow{k} W$, where $W$ is a waste species. This introduces a term $-kx^+x^-$ in the new variables' ODEs. Another approach scales the $-x^+x^-$ term by the entire ODE $x'$ [8], in which case it is known that the resulting system is bounded [6]. It is often assumed that the fast annihilation reaction method results in a bounded system for sufficiently large $k$, but this has not yet been shown:

*Problem 1.* Given $\mathbf{x}$, is there a constant $k$ so that the dual-railed system based on annihilation reactions ($Z = k$ in Eq. (2)) is bounded?

**Construction 1.** *(Naive Dual-Railing). If* $\mathbf{x} = (x_1, x_2, ..., x_n)$ *is a GPAC consisting of* $n$ *variables with ODEs*

$$\begin{aligned} x_1' &= p_1(x_1, ..., x_n), \\ x_2' &= p_2(x_1, ..., x_n), \\ &... \\ x_n' &= p_n(x_1, ..., x_n), \end{aligned}$$

*then we define a new system* $\hat{\mathbf{x}} = (x_1^+, x_1^-, x_2^+, x_2^-, ..., x_n^+, x_n^-)$ *as follows:*

1. *For each* $i$*, define a new polynomial* $\hat{p}_i(\hat{\mathbf{x}}) = p_i(x_1^+ - x_1^-, x_2^+ - x_2^-, ..., x_n^+ - x_n^-)$ *by substituting the differences* $x_i^+ - x_i^-$ *for each* $x_i$ *everywhere in* $p_i$*.*
2. *Next, split this polynomial up into* $\hat{p}_i = \hat{P}_i^+ - \hat{P}_i^-$*, where* $\hat{P}_i^+$ *consists of all the positive terms in* $\hat{p}_i$*, and* $\hat{P}_i^-$ *consists of all the positive parts of the negative terms in* $\hat{p}_i$*.*
3. *Finally, define the CRN-implementable ODEs for the new variables by:*

$$\begin{aligned} x_i^{+'} &= \hat{P}_i{}^+ - x_i^+ \cdot x_i^- \cdot Z, \\ x_i^{-'} &= \hat{P}_i{}^- - x_i^+ \cdot x_i^- \cdot Z, \end{aligned} \tag{2}$$

*where either* $Z = \hat{P}_i{}^+ + \hat{P}_i{}^-$ *or* $Z = k$ *(where* $k$ *is a sufficiently large constant), based on the chosen approach.*

*As a result, for every original variable,* $x_i' = p_i = \hat{P}_i^+ - \hat{P}_i^- = x_i^{+'} - x_i^{-'}$*.*

## 4 Selective Dual-Railing

Next, we demonstrate an improvement based on the observation that not every variable necessarily needs to be dual-railed. As before, let $\mathbf{x} = (x_1, x_2, ..., x_n)$ be a GPAC with $n$ variables. We identify a vertex in a graph with its variable in the ODE system.

**Definition 3.** *A variable* $x_i$ *is* ill *if its equation* $x_i'$ *is* ill-formed*, i.e., fails to have the form of Eq. (1).*

**Definition 4.** *If* $x_a, x_b$ *are variables of* $\mathbf{x}$*, we say that* $x_a$ affects $x_b$ *if* $x_a$ *appears anywhere in the positive terms of* $x_b'$*, i.e. if there is a positive monomial* $T = x_a S$ *in* $x_b'$ *where* $S$ *is some monomial or a constant.*

**Definition 5.** $x_a$ can infect $x_b$ *if* $x_a$ *affects* $x_b$ *and there is a positive term in* $x_b'$ *where* $x_a$ *appears but* $x_b$ *does not.*

Note that if $x_b'$ has a positive term where $x_b$ appears, then splitting up the other variables in that term can only produce negative terms that *also* contain $x_b$, thus it does not harm CRN-implementability.

**Definition 6.** *The* infection graph *is the directed graph* $I(\mathbf{x}) = (V, E)$ *where* $V$ *is the set of variables of* $\mathbf{x}$ *and* $E = \{(x_a, x_b) \mid x_a \text{ can infect } x_b\}$.

As an example, consider the following system, noting that only one variable, $a$, has an ill-formed ODE expression (the term $-d$ does not include $a$):

$$\mathbf{x} = \begin{cases} a' = b - ab - d \\ b' = c + a - ab^2 \\ c' = bc - bc^2 + 0.1d \\ d' = c - cd \end{cases} \tag{3}$$

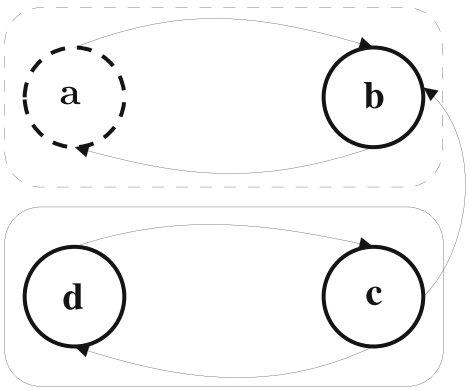



**Fig. 1.** $I(\mathbf{x})$, of Eq. 3, with one infected (dashed) and one uninfected (solid) SCC.

Construction 1 results in 8 variables. However, if we look at the strongly connected components of $I(\mathbf{x})$ in Fig. 1, we see that only two variables require rewriting: $a$ and $b$, requiring 6 variables total for the system.

Below is an algorithm that selectively dual-rails $\mathbf{x}$:

**Algorithm 1:** Selective Dual-Railing

**Input**: A GPAC $x = (x_1, x_2, \ldots, x_n)$.
**Output**: A CRN-implementable system.

Construct the infection graph $I(\mathbf{x}) = (V, E)$.
Compute the set $S(I)$ of strongly connected components (SCCs) of $I$ via Tarjan's algorithm.
Build the induced acyclic graph of SCCs, with edges inherited from $I$.
Identify the set ILL $\subseteq \{x_1, \ldots, x_n\}$ of ill-formed variables.
**foreach** $S \in S(I)$ **do**
    **if** *$S$ is reachable from an SCC $S'$ with $S' \cap$ ILL $\neq \emptyset$* **then**
        Label S infected.
Let $R \subseteq V$ be the set of variables belonging to infected SCCs.
**foreach** $x_i \in R$ **do**
    Define the polynomial $\hat{p}_i$ and the sets $\hat{P}_i^+$ and $\hat{P}_i^-$ by substituting ($y_j$, $y_j^+ - y_j^-$) for all $y_j \in R$ as in Construction 1.
    Apply dual-railing to $x_i'$ as in Eq. (2).
**foreach** $x_i \notin R$ **do**
    **foreach** $y_i \in R$ **do**
        Substitute $(y_i^+ - y_i^-)$ for all instances of $y_i$ in $x_i'$.
**return** the modified ODE system.

The values of original system variables are preserved via the relationships $x_i = (x_i^+ - x_i^-)$, as is boundedness if one uses $Z = \hat{P}_i^+ - \hat{P}_i^-$. All ill-formed variables, and those that depend on them, are dual-railed - thus, become CRN-implementable. Variables $x_i$ that are not dual-railed remain CRN-implementable: although they may undergo substitutions, by Definition 5 these cannot result in negative terms that fail to include $x_i$.

One may use, for example, Tarjan's algorithm [12] to identify strongly connected components in $I(\mathbf{x})$. Algorithm 1 introduces a number of new variables equal to the size of the union of infected SCCs of $I(\mathbf{x})$, typically fewer than $n$. The algorithm parses the input system in linear time with respect to its string representation. It performs symbolic transformations that may introduce new variables or expand terms. The total runtime is output-sensitive, and in the worst case can be exponential in the input size due to polynomial expansion. In practice, the output is typically much smaller, and the algorithm runs in time linear in the input and output sizes combined.

## 5 Conclusion

While the standard dual-railing method for bounded systems (Construction 1) results in $n$ new variables, the method of the previous section often results in

fewer. It can be worthwhile to use it when one is concerned about the number of variables and about preserving boundedness, and especially if one suspects that the initial system has some well-formed variables that are decoupled from ill-formed ones.

## References


1. Bournez, O., Campagnolo, M.L., Graça, D.S., Hainry, E.: The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation. In: Proceedings of the 3rd International Conference on Theory and Applications of Models of Computation, pp. 631–643. Springer Berlin Heidelberg (2006)
2. Bournez, O., Graça, D.S., Pouly, A.: Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. J. ACM **64**(6), 38 (2017)
3. Bush, V.: The differential analyzer. A new machine for solving differential equations. J. Franklin Inst. **212**(4), 447–488 (1931)
4. Fages, F., Le Guludec, G., Bournez, O., Pouly, A.: Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In: Feret, J., Koeppl, H. (eds.) CMSB 2017. LNCS, vol. 10545, pp. 108–127. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67471-1_7
5. Hárs, V., Tóth, J.: On the inverse problem of reaction kinetics. In: Colloquia Mathematica Societatis János Bolyai, 30: Qualitative Theory of Differential Equations, pp. 363–379 (1981)
6. Huang, X.: Chemical reaction networks: computability, complexity, and randomness. PhD thesis, Iowa State University (2020)
7. Huang, X., Huls, R.N.: Computing real numbers with large-population protocols having a continuum of equilibria. In: 28th International Conference on DNA Computing and Molecular Programming (2022)
8. Huang, X., Klinge, T.H., Lathrop, J.I.: Real-time equivalence of chemical reaction networks and analog computers. In: International Conference on DNA Computing and Molecular Programming, pp. 37–53. Springer (2019)
9. Klinge, T.H., Lathrop, J.I., Osera, P.M., Rogers, A.: Reactamole: functional reactive molecular programming. Nat. Comput. **23**(3), 477–495 (2024)
10. Shannon, C.E.: Mathematical theory of the differential analyzer. Stud. Appl. Math. **20**(1–4), 337–354 (1941)
11. Soloveichik, D., Seelig, G., Winfree, E.: DNA as a universal substrate for chemical kinetics. In: Goel, A., Simmel, F.C., Sosík, P. (eds.) DNA 2008. LNCS, vol. 5347, pp. 57–69. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03076-5_6
12. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
13. Teschl, G.: Ordinary Differential Equations and Dynamical Systems. American Mathematical Society (2012)
14. Vasic, M., Chalk, C., Khurshid, S., Soloveichik, D.: Deep molecular programming: a natural implementation of binary-weight ReLU neural networks. In: International Conference on Machine Learning, pp. 9701–9711. PMLR (2020)
15. Vasić, M., Soloveichik, D., Khurshid, S.: CRN++: molecular programming language. Nat. Comput. **19**(2), 391–407 (2020)