Approximations for stochastic graph rewriting[⋆]

Vincent Danos[1], Tobias Heindel[1], Ricardo Honorato-Zimmer[1], and Sandro Stucki[2]

[1] School of Informatics, University of Edinburgh, Edinburgh, United Kingdom
[2] Programming Methods Laboratory, EPFL, Lausanne, Switzerland

Introduction. We present a method to compute approximate descriptions of a specific class of stochastic systems. For the method to apply, the system must be presented as a Markov chain on a state space consisting in graphs, and jumps must be described by transformations which follow a finite set of local rules.

The method is a form of static analysis and uses concepts reminiscent of critical pairs in term rewriting systems. Its output is a system of coupled ordinary differential equations (ODE) which tracks the evolution of the mean number of (typically small) subgraphs. In favourable cases, the set of ODEs is an exact and finite description of these mean numbers. But even when it is not, it often describes an approximation which can reveal interesting properties of the original system.

The method was first conceived in relation to a special type of graphs, namely the site graphs which form the basis of the Kappa language [3]. Recently, the authors have taken again this method with the goal to extend it to a broader class of "graph-like" objects and suitable transformations based on adhesive categories [5]. In this note, the goal is rather the opposite. We narrow down the construction to consider only simple graphs and invertible rules, to not be distracted by technicalities, and obtain a simple account.

An example - the voter model. Let us start with an example of the type of systems we are interested in. One has a graph G whose nodes can be in either of two states 0 or 1. There are two possible rules to transform G which we call flips and swaps. To apply either type of rule, we first need to locate in G a pair of neighboring nodes u, v with different states. Then we can either: 1) flip the internal state of u or v to match its neighbor's state; or 2) swap the edge connecting u and v with an edge connecting u or v to another node w.

In each case we obtain a new graph (on the same set of nodes). The evolution of the system consists then in repeatedly applying flips and swaps. Fig. 1 and Fig. 2 illustrate the basic transformation steps. Fig. 3 shows an example of a graph which can be transformed by both types of rules.

If we say that colors represent opinions, then we can interpret the rules as the nodes trying to not have neighbors of a different opinion. A node with a neighbor of the opposite persuasion can change his (by a flip), or turn to another neighbor

[⋆] This paper is an improved version of an invited paper presented at ICFEM'14. The research was sponsored by the European Research Council (ERC) under the grants 587327 "DOPPLER" and 320823 "RULE".

(by a swap). Several variants of this "voter" models are studied [1]. For instance w, the target node of the swap, can be chosen of the same color as u the node doing the swapping, and/or can be picked within a prescribed distance of u.

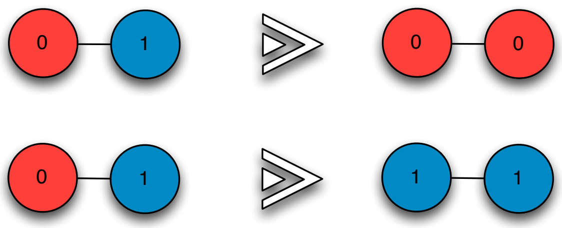

Fig. 1. Flip rules - we use colors to represent the internal states of nodes.

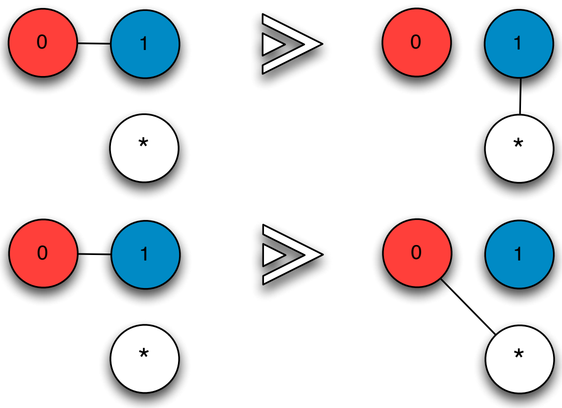

Fig. 2. Swap rules - we use $*$ for the unknown state.

At any given point, several transformation rules might be applicable to a graph, and each rule may be applied in several ways depending on where the left hand side of the rule is matched in the current graph. A way in which a rule can be applied is called an instance of that rule. If a graph is such that no rules can be applied to it, we call it a normal form (or a frozen state). In the example, normal forms are "fragmented" graphs with no edge connecting two nodes of opposing colors. The graph of Fig. 3 can be frozen in just one step. In the opinion interpretation, one is interested, among other things, in understanding how likely it is that an opinion wins over the entire graph; that is to say, how likely it is that one reaches a normal form which is monochrome. For instance, in Fig. 3, it is still possible, by a long series of steps, to propagate the red color to the entire graph, but perhaps it is not very likely. To address this type of question, one needs to define the likelihood of a given instance to apply, and equip transformations with a probabilistic structure.

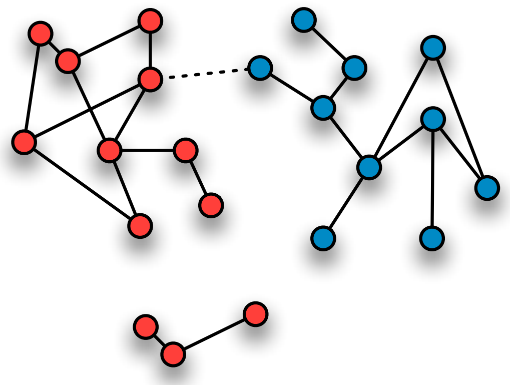

Fig. 3. An almost frozen state: it is enough to swap the dotted edge to reach a normal form where red and blue nodes no longer have any connexions; on the other hand, it also possible to reach a frozen state which is entirely red.

Notations and Definitions. Before we turn to probabilities, we fix a few notations. This example, as all systems considered here, has rules which preserve the underlying set of nodes and can explore only finitely many colors (those mentioned in the rules). Therefore the set of graphs reachable from a given initial graph is finite. If we write N for the set of nodes of an initial graph G_0, and $\mathscr{G}_N$ for the (finite) set of all graphs on N with reachable colors, then all graphs reachable from G_0 will be in $\mathscr{G}_N$. In our voter example, the number of edges is also preserved and this provides a further restriction on the set of reachable graphs.

Let us assume from now on that we are given a finite set of rules $\mathscr{R}$ and an initial graph G_0 with nodes in N. The objects which we transform are simple graphs where nodes have colors (represented by integers). In other words, we consider triples N, E, σ where N is a finite set of nodes, E a finite set of undirected edges over N, and σ maps a subset of N to integers. Partially colored graphs are only used in rules (eg the swap rules in Fig. 2).

We will use the following typographic conventions: we will write A, B, etc, for (typically small) graphs (e.g. those which appear on the left hand sides of rules in $\mathscr{R}$) which may have nodes without colors, and x, y, etc, for arbitrary graphs in $\mathscr{G}_N$ to which rules are applied and which are fully colored.

Matches and Observables. A match $f : A \to x$ is a graph morphism from A to x which 1) preserves internal states, and 2) is injective on nodes. We write $c(f)$ for the codomain x of f. The codomain of f is not to be confused with f's image, written $f(A)$, which typically is a strict subgraph of its codomain x.

Let us write $[A; x]$ for the set of matches between A and x and, $[A]$ for the map from $\mathscr{G}_N$ to $\mathbb{N}$ defined as $[A](x) = |[A; x]|$. This map $[A]$ is counting the number of instances of A in x. We call such maps graph observables. We write $\mathscr{C}$ for the set of connected graph observables, $\mathscr{B}$ for set of all graph observables, and $\mathscr{A}$ for the linear subspace spanned by $\mathscr{B}$ in the vector space $\ell(\mathscr{G}_N)$ of real-valued functions over $\mathscr{G}_N$.

Clearly:

$$\mathscr{C} \subseteq \mathscr{B} \subseteq \mathscr{A} \subseteq \ell(\mathscr{G}_N)$$

We call $\mathscr{A}$ the algebra of graph observables, because it also has a commutative algebra structure, with multiplicative unit $[\varnothing]$, as we will see shortly.

Rules. A rule is α is a pair of graphs α_L, α_R which are defined on the same set of nodes. To apply such a rule to a graph x, we choose a match $f \in [\alpha_L; x]$ (if any), and replace the edges and states in the image subgraph $f(\alpha_L)$ of α_L as we find them in α_R.

An example of rule and rule application is given in Fig. 4. The corresponding match is shown in Fig. 5.

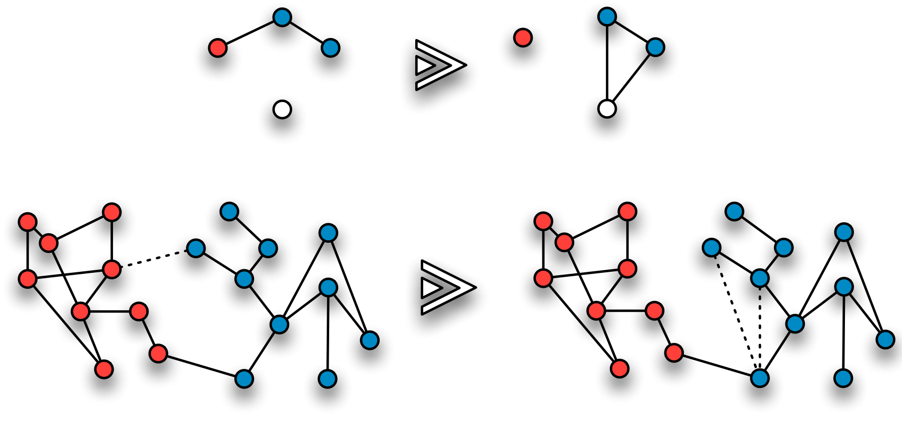

Fig. 4. Example of a rule and rule application. Note that the rule left hand side does not need to be connected. The white node stands for a node of unspecified color.

Thus, the difference between α_L and α_R, the left and right hand sides of the rule, represent the modifications subsequent to the application of the rule. We write $\alpha(f)$ for the post-match, that is the residue of the match after applying

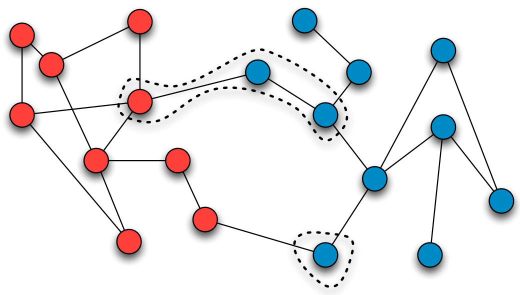

Fig. 5. Example of a match of the rule above.

the rule. Note that $\alpha(f)$ is the same map as f on nodes as our rules do not modify the underlying node set.

We write $\alpha^\dagger$ for the rule inverse to α.

With the non-deterministic structure in place, we move on to the quantitative aspects. The idea is that by assigning rates to rules in $\mathscr{R}$ we can generate a continuous-time Markov chain (CTMC) on $\mathscr{G}_N$.

CTMC preliminaries (See Ref. [6]). In general, to define a CTMC on a finite space all one needs to do is to provide a rate matrix Q where $q_{xy} \geq 0$ is the rate at which the chain jumps from x to a y different from x, and, conventionally, q_{xx} is defined as $-\sum_{x \neq y} q_{xy}$.

Suppose now we write $p(x)$ for the time-dependent probability to be at a certain state x in $\mathscr{G}_N$. We can consider p as an element of $\ell(\mathscr{G}_N)$. The rate matrix Q governs the evolution of p via the master equation:

$$\frac{d}{dt} p^T = p^T Q \tag{1}$$

where p^T is the transpose of p. (The transpose comes from the convention that q_{xy} is the rate at which the chain jumps from x to y.)

For ϕ a function in $\ell(\mathscr{G}_N)$, the (time-dependent) mean (or expected value or average) of ϕ according to p is $E_p(\phi) := p^T \phi$, and it follows directly from the master equation that:

$$\frac{d}{dt} E_p(\phi) = E_p(Q(\phi)) \tag{2}$$

It easy to see that if we take as function $\phi = \delta_x$ the function which is 1 at x and zero else, $E_p(\delta_x)$ is the same as $p(x)$ and the equation we have just written is the master equation (1) for $p(x)$. That is to say equation (2) becomes the projection of the master equation on the x-coordinate. And, as such, it contains as much information.

Thus, suppose given a rate map $k : \mathscr{R} \to \mathbb{R}^+$ which associates to each rule a positive real number.

We define the transition rates of the associated CTMC on $\mathscr{G}_N$ as follows. For α in $\mathscr{R}$, x, y in $\mathscr{G}_N$, we define a rate matrix Q_α with coefficients:

$$\begin{aligned} q^\alpha_{xy} &= |\{f \in [\alpha_L; x] \mid c(\alpha(f)) = y \neq x\}| \\ q^\alpha_{xx} &= -\textstyle\sum_{x \neq y} q^\alpha_{xy} \end{aligned}$$

The coefficient q^α_{xy} counts the number of instances of the rule α which transform state x into state y. (In the example, flips have non trivial multiplicities: specifically, if u is the node being flipped, q^α_{xy} is the number of neighbors of u of the opposite color.) This simply means that Q_α samples transitions uniformly at random. The competition between rules (eg, between flips and swaps in our example) is taken care of by the rate map k.

The rate matrix Q of our system is then defined by combining the Q_α:

$$Q = \textstyle\sum_{\alpha \in \mathscr{R}} k(\alpha) \cdot Q_\alpha$$

The rate matrix (also known as the infinitesimal generator of the CTMC) defines a linear operator on the vector space $\ell(\mathscr{G}_N)$. Specifically, if we write q_{xy} for Q's coefficients, and pick ϕ a function in $\ell(\mathscr{G}_N)$, Q's action on ϕ is given by:

$$\begin{aligned} Q(\phi)(x) &= \textstyle\sum_y q_{xy}(\phi(y) - \phi(x)) \\ &= \textstyle\sum_{\alpha \in \mathscr{R}} k(\alpha) \sum_y q^\alpha_{xy}(\phi(y) - \phi(x)) \\ &= \textstyle\sum_{\alpha \in \mathscr{R}} k(\alpha) \sum_{f \in [\alpha_L; x]} q^\alpha_{xc(\alpha(f))}(\phi(c(\alpha(f)) - \phi(x)) \end{aligned}$$

In words, $Q(\phi)(x)$ is the mean rate of change of ϕ at x.

Return to the example. Suppose we pick as our ϕ the graph observable [0] which counts the number of nodes in state 0. Clearly $Q_{swap_0}([0]) = Q_{swap_1}([0]) = 0$ as swaps do not change colors.

For the flips from 0 to 1, we compute:

$$Q_{flip_0}([0])(x) = -\textstyle\sum_{y \neq x} q_{xy} = -[01]$$

where 01 is short for the pattern $0^a, 1^a$, and [01] is the observable which counts the number of edges between neighbors of opposite colors.

The symmetric flip is computed in the same way and by summing all contributions we get the following instance of (2):

$$\frac{d}{dt} E_p([0]) = (k_{10} - k_{01}) E_p([01]) \tag{3}$$

with k_{01} and k_{10} the respective rates associated to *flip*$_0$ (flip from 0 to 1), and *flip*$_1$ (the symmetric flip).

We can already notice a few things.

First, a formal remark: the equation obtained for the evolution of [0], which is in our algebra $\mathscr{A}$, introduces another function [01] also in $\mathscr{A}$. In other words, $Q([0])$ is a (linear) function of [01]. This is a general fact. For all αs, $\mathscr{A}$ is closed under the linear map Q_α. Therefore, the same holds of Q which is a linear combination of Q_αs. In fact, this is our main result! We will derive below a concrete expression for $Q_\alpha([F])$ for any graph observable F, and any rule α. This will establish the closure of $\mathscr{A}$ under Q, and give an effective way to write (2) for all observables in $\mathscr{A}$.

Second, a concrete remark: if the flip rules are symmetric (corresponding to opinions which are equally persuasive), that is to say if $k_{01} = k_{10}$, then $\frac{d}{dt}E_p([0]) = 0$. This does not mean that the final number of 0s will be the same in all trajectories to normal form, just that, on average, this number will be exactly what it was at the start. Thus, interesting information about the dynamics can be found from ODEs such as the one we have derived above. So seeking a general method to generate them, as we do here, is a worthy pursuit.

Last, another general remark: the new observable [01] is larger than [0] in the sense that the underpinning graph is larger. This is also general. As we will see, the new observables needed to express $Q([F])$ can be larger than F. The idea is that one has to write an instance of equation (2) for them as well. Hence, the process of deriving the ODE system for a graph observable of interest can be seen as an expansion. Even if in our case the expansion is finite, as $[F] = 0$ as soon as F has more than N nodes, in practice, one needs to truncate the expansion.

Gluings. To derive an effective version of (2) in the general case, we need a additional ingredient, namely minimal gluings. A gluing μ of two graphs A, B is a pair of matches $f : A \to x$, $g : B \to x$. We write $\pi_0(\mu) = f$, $\pi_1(\mu) = g$, and $c(\mu) = x$ for the common codomain of f and g. Given μ, one can always obtain a new gluing $f_1 : A \to C$, $g_1 : B \to C$ with C the union of the images of f and g, and $f = j \circ f_1$, $g = j \circ g_1$, where j is the inclusion of C in x. We call the pair f_1, g_1 a minimal gluing of A and B.

There are finitely many minimal gluings of A and B up to isomorphism. We write $A * B$ for this (finite) set of minimal gluings. In the worst case, there can be exponentially many non-isomorphic minimal gluings, as each corresponds to determining a shared subgraph of A and B in the gluing, namely the intersection of the images of f and g. There is a largest minimal gluing, corresponding to no sharing at all, which is the disjoint sum of A and B, written $A + B$.

A gluing decomposes through exactly one minimal gluing. Hence:

$$[A][B] = \textstyle\sum_{\mu \in m(A,B)} [\mu]$$

It follows that $\mathscr{A}$ is closed under product, hence is a sub-algebra. Besides, we can rewrite the above as:

$$[A + B] = [A][B] - \textstyle\sum_{\mu \in m(A,B) \setminus \{A+B\}} [\mu]$$

and one sees that non-connected observables can be expressed as polynomials of connected ones. In other words, $\mathscr{A}$ is the polynomial closure of $\mathscr{C}$ the set of connected observables. (The degree of the polynomial decomposition of $[F]$ in $\mathscr{C}$ is the number of connected components of F.)

Proving that $\mathscr{A}$ is closed. We can now prove that $\mathscr{A}$ is closed under the action of Q. As Q is a linear combination of Q_αs, it is enough to prove closure under Q_α, and as $\mathscr{B}$ spans $\mathscr{A}$, it is enough to examine the action of Q_α on a graph observable. So, let $[F]$ be that observable, and x a graph in $\mathscr{G}_N$. By definition of Q_α we get:

$$\begin{aligned}(Q_\alpha[F])(x) &= \textstyle\sum_y q^\alpha_{xy}([F](y) - [F](x)) \\ &= \textstyle\sum_{f\in[\alpha_L;x]} |[F;c(\alpha(f))]| - \sum_{f\in[\alpha_L;x]} |[F;x]|\end{aligned}$$

where $\alpha(f)$ is the post-match corresponding to f after firing α at f, and $c(\alpha(f))$ its codomain, that is the graph resulting from firing α.

We see that the action of Q_α at x decomposes naturally in two terms, $Q_\alpha = Q^+_\alpha - Q^-_\alpha$, one which produces new instances of F and one which consumes existing ones. The consumption part is easy to evaluate:

$$Q^-_\alpha([F]) = \textstyle\sum_{\mu\in F*\alpha_L}[c(\mu)]$$

Indeed the right hand side is equal to $[F][\alpha_L]$ by definition of minimal gluings. For the production term, we get a similar expression:

$$Q^+_\alpha([F]) = \textstyle\sum_{\mu\in F*\alpha_R}[c(\alpha^\dagger(\pi_0(\mu))))]$$

Recall that $\pi_0(\mu)$ is the first match in the gluing μ. We apply the inverse rule $\alpha^\dagger$ to this post-match to obtain $c(\alpha^\dagger(\pi_0(\mu)))$. This counting is correct because there is a bijection between post-matches from $c(\pi_0(\mu))$ to $c(\alpha(f))$, and pre-matches from $c(\alpha^\dagger(\pi_0(\mu)))$ to x.

Thus we obtain that $\mathscr{B}$, and therefore evidently $\mathscr{A}$ its linear span, is closed under the action of Q_α, and therefore any linear combination of such.

Remarks. Again there are a few remarks worth making.

First, even if the graph observable F which we start from is connected, the observables on the right hand side of $Q^\pm_\alpha([F])$ might not be. That is to say, the linear span of $\mathscr{C}$ is not necessarily closed under Q. However, it is not difficult to see that this will be the case if all rules contributing to the generator Q have a connected left hand side (so-called linear rules). Such rules sets form an interesting subclass of "solid-state" transformations. [1]

The second remark is a caveat. For rules more general than the ones considered here, where one can create nodes, the bijection argument which we have relied on to justify the production term fails. A more detailed analysis is needed. But the ideas are essentially the same and the formula obtained only slightly more complex.

Last, in the two terms which we have introduced above, $Q^\pm_\alpha([F])$, the summation extends to all minimal gluings of F on both sides of the rule; in practice,

[1] In other words: while the linear span of $\mathscr{B}$ is closed under general rules, the linear span of $\mathscr{C}$ is closed under *linear* ones. From this second fact follows the well-known fact that, for linear Petri Net systems (chemical reaction networks with unary reactions), the rate equation describes the evolution of the mean of the state.

we can restrict these sums to gluings where F undergoes an actual modification due to the firing of α or $\alpha^\dagger$. We call these gluings relevant. The contributions of the irrelevant ones cancel out. In examples, we never consider those.

The general rate equation for graphs. From the above, using the linearity of expectations, we derive the explicit form of (2) which we seek. Specifically, for a graph observable F in $\mathscr{B}$, we get:

$$\begin{aligned}\tfrac{d}{dt}E_p([F]) = &\textstyle\sum_{\alpha\in\mathscr{R}} k(\alpha)\sum_{\mu\in F*\alpha_R} E_p([c(\alpha^\dagger(\pi_0(\mu)))]) \\ &-\textstyle\sum_{\alpha\in\mathscr{R}} k(\alpha)\sum_{\mu\in F*\alpha_L} E_p([c(\mu)])\end{aligned}$$

So far there is no approximation involved. The equation is exact. But as we have seen in the example, it requires the knowledge of additional observables which leads to writing more similar equations, possibly of increasing complexity.

Example continued. To see concretely how more complex terms follow from the expansion, we can return to the example and compute the equations associated to [01] the number of opposing neighbors or the distance to normal form. As we have seen earlier, the equation for $E_p[0]$ generates [01] as a new observable (in the non symmetric case at least). So it is the natural next step.

Below we neglect irrelevant gluings. To denote chain motifs, we use abbreviation similar to the ones used before, eg we write 101 for the graph with three nodes and two edges.

$$\begin{aligned}Q^-_{flip_0}([01]) &= -[01]-[101]\\ Q^+_{flip_0}([01]) &= [001]\\ Q^-_{flip_1}([01]) &= -[01]-[010]\\ Q^+_{flip_1}([01]) &= [011]\\ Q^-_{swap_0}([01]) &= -[01]\\ Q^+_{swap_0}([01]) &= [01+1]\\ Q^-_{swap_1}([01]) &= -[01]\\ Q^+_{swap_1}([01]) &= [01+0]\end{aligned}$$

Hence if we write k_0, k_1 for the swap rates we can collect all the contributions above and we obtain the following ODE:

$$\begin{aligned}\tfrac{d}{dt}E_p([01]) = \;& k_{01}(E_p[001]-E_p[01]-E_p[101])\\ &+k_{10}(E_p[011]-E_p[01]-E_p[010])\\ &+k_0(E_p[01+1]-E_p[01])\\ &+k_1(E_p[01+0]-E_p[01])\end{aligned}$$

We can simplify this general expression by supposing that flips and swaps are symmetric. If we set the following notations: $k=k_{01}=k_{10}$, $k'=k_0=k_1$, and arrange the terms below by size, we get:

$$\begin{aligned}\tfrac{d}{dt}E_p([01]) = &-2(k+k')E_p[01]\\ &+k(E_p[001]+E_p[011]-E_p[101]-E_p[010])\\ &+k'(E_p[01+1]+E_p[01+0])\end{aligned}$$

Non-connected observables $[01+1]$ and $[01+0]$ appear as anticipated, as well as larger observables such as $[001]$. In Ref. [1], where this is example is developed, the authors derive a similar ODE by hand. (There is a slight difference due to the fact that their swap rules do not take into account the multiplicity of the $*$ node in the definition of an instance; but that is of no consequence for our exposition.)

At this stage, we are facing the problem of either writing an ODE for all the new terms which have appeared (which poses no conceptual problem but would be extremely tedious to do by hand), or to truncate and express the new larger observables as functions of simpler ones. Even if we were to go for the former option, we would have to find a way of truncating the expansion at some point! So let us follow the second option and see how this can be done.

To get rid of the non-connected observables, we can exploit the polynomial decomposition above. This gives us $[01+1] = [01][1]-[01]$ and hence $E_p[01+1] = E_p([01][1]) - E_p[01]$. Now, using an approximation, we can simplify the first term as:

$$E_p([01][1]) \sim E_p([01])E_p([1])$$

This type of approximation can be performed in general and consists in assuming independence of observables. To get rid of the connected terms of the form $[001]$, we need another approximation principle. We can either set them brutally to zero, or else, more subtly, apply what is known as a pair approximation which in this case takes the form:

$$E_p([001])E_p([0]) \sim E_p([00])E_p([01])$$

This second type of approximation is an assumption of conditional independence. Neither comes with a general bound on the error they introduce. But in practice, they often give interesting results.

Example concluded. Using the same machinery, one can compute higher order moments of (the distributions of) observables. Say we want to estimate the variance of $[0]$ the mean of which we have seen is a constant in the symmetric case $k = k_{01} = k_{10}$. In the extreme case where there are no swaps allowed ($k' = 0$), and assuming the initial graph G_0 is connected, normal forms can only be monochrome. This means that one opinion disappears (and it is easy to see that the probability for an opinion to win this all-or-nothing competition is equal to its initial fraction). So, intuitively, in this case $[0]$ will have a high variability, and in general, the lower the variance the more likely it is that the graph will split in two separate colors with none of the colors completely winning.

To compute this variance, we write:

$$\begin{aligned}\tfrac{d}{dt}V_p([0]) &= \tfrac{d}{dt}E_p([0]^2) - \tfrac{d}{dt}E_p([0])^2\\ &= \tfrac{d}{dt}E_p([0+0]) + \tfrac{d}{dt}E_p([0]) - 2E_p([0])\tfrac{d}{dt}E_p([0])\end{aligned}$$

Using the connected decomposition and the general equation (2), we get:

$$\begin{aligned}\tfrac{d}{dt}E_p([0+0]) &= -2k_{01}E_p([0+01]) + 2k_{10}E_p([0+01]) + 2k_{10}E_p([01])\\ &= 2(k_{10}-k_{01})E_p([0+01]) + 2k_{10}E_p([01])\end{aligned}$$

Using (3), we get:

$$\frac{d}{dt}V([0]) = (k_{10} + k_{01})E([01]) + 2(k_{10} - k_{01})Cov([01][0]) \tag{4}$$

This expression, differently to that for the mean $E_p([0])$ is not degenerate even in the symmetric case. In the latter case, the second term disappears and we see that the variance will be monotonically increasing (starting at 0) with a slope proportional to the current mean number of 01-edges. Interestingly, the constant of proportionality does not depend directly on the swap rate.

Conclusion. There are many examples other than the one we have used here where the type of deterministic approximations considered in this note have been found useful. Examples abound in particular in the literature of the so-called adaptive networks [2]. The ability to define and generate them in a systematic way, as we have presented, is important on several counts. For one thing, the derivation involves combinatorics and there is a limit to the size of an expansion one can do by hand. With an implementation, one could go higher in the order of expansion before introducing approximations, and thus obtaining potentially more accurate approximations. For the same reason, the derivation of these approximations is quite error-prone and a mechanical derivation can be beneficial. A case in point is equation (4) which we had formally derived by hand wrongly, and has been now derived using a prototype implementation (see below). Our careful and explicit construction carries over to several graph-like structures with little modifications. One can play with the type of objects (eg directed graphs, hypergraphs, simplicial sets), or the type of matches (eg induced subgraphs) or even the type of rules (eg considering rules which create and/or merge nodes). The general axiomatic approach leads to a more unified picture. Finally, our method to generate the differential system associated to an observable, and its subsequent expansion, could lead to interesting formalizations of the approximation principles used to cut the expansion beyond the simple pair approximation.

Tool. We have developed a prototype tool in Scala which generates approximations as described above [4], which is extremely useful when expansions become too large.

References

1. R. Durrett, J. P. Gleeson, A. L. Lloyd, P. J. Mucha, F. Shi, D. Sivakoff, J. E. Socolar, and C. Varghese. Graph fission in an evolving voter model. *Proceedings of the National Academy of Sciences*, 109(10):3682–3687, 2012.
2. J. P. Gleeson. High-accuracy approximation of binary-state dynamics on networks. *Physical Review Letters*, 107(6):068701, 2011.
3. R. Harmer, V. Danos, J. Feret, J. Krivine, and W. Fontana. Intrinsic information carriers in combinatorial dynamical systems. *Chaos*, 20(3), 2010.

4. R. Honorato-Zimmer. Graph rewriting library for Scala. `https://github.com/rhz/graph-rewriting`, 2014.
5. S. Lack and P. Sobocinski. Adhesive categories. In I. Walukiewicz, editor, *FoSSaCS*, volume 2987 of *LNCS*, pages 273–288. Springer, 2004.
6. J. R. Norris. *Markov chains*. Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998.
7. T. Sternagel and H. Zankl. Kbcv a knuth-bendix completion visualizer. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 530–536. Springer Berlin Heidelberg, 2012.