Ordering Networks

Lars Hellström[1]

1 **Division of Applied Mathematics, The School of Education, Culture and Communication, Mälardalen University, Box 883, 721 23 Västerås, Sweden;** `lars.hellstrom@residenset.net`

Abstract

This extended abstract discusses the problem of defining quasi-orders that are suitable for use with network rewriting. The author's primary interest is in using network rewriting as a tool for equational reasoning in algebraic theories with both operations and co-operations.

Keywords and phrases rewriting, network, PROP, PROP order

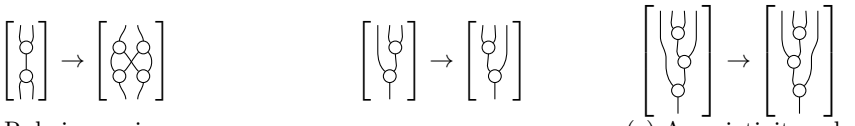
1 Introduction

Network rewriting [1] is a kind of graph rewriting; networks are directed acyclic graph with some extra structure—roughly the same extra structure as makes terms out of trees, but completely symmetric with respect to input and output. This allows networks to be viewed as expressions, so that on one hand one can use networks as an alternative notation where ordinary terms do not quite suffice, and on the other one can take a network and *evaluate* it with rather arbitrary interpretations of the symbols. This latter approach turns out to be convenient for defining orders on networks.

Formally, a *network* is a directed acyclic graph (DAG) with the following extra data. (i) There are two distinguished vertices 0 and 1 that represent the output and input respectively sides of the network; edges from 1 are input legs of the network, and edges to 0 are output legs of the network. (ii) Each inner vertex (those other than 0 or 1) is decorated with a symbol from a doubly ranked alphabet. If the symbol $D(v)$ of vertex v has rank (m, n), then the in-degree of v must be n (the *arity*) and the out-degree of v must be m (the *coarity*). (iii) There is at each vertex a total ordering of the incoming edges, and a separate total ordering of the outgoing edges. The arity of the network as a whole is the degree (all outgoing) of the input vertex 1, and the coarity of the network as a whole is the degree (all incoming) of the output vertex 0. By convention here, networks are drawn with all edges oriented downwards, so no arrowheads need to be drawn in them.

The use of networks as expressions when ordinary terms do not suffice may be observed in several specialities—physicists working with tensor fields (e.g. in General Relativity) may use the Penrose [6] graphical notation to visualise the structure of a complex expression, algebraicists studying Hopf algebras may use 'diagram shorthand' (see e.g. [4]) to do their calculations, and quantum computer programming is very much a matter of building 'arrays of quantum gates'—all of which may be formalised as networks or minor variations thereof. The common factor in these applications are operations that produce multiple results (in the sense of a subroutine having several out-parameters, not in the sense of a multivalued function). Much of what specialists in these fields do with their diagrams can be described as informal network rewriting.

The abstract setting within which one may evaluate a network is that of an algebraic structure known as a PROP [3, Ch. V]. This consists of a set of doubly ranked elements (sometimes formalised as the set of all morphisms in a category whose objects are the nonnegative integers; the domain of a morphism is then its arity and the codomain is its



(a) Rule increasing the number of vertices

(b) Associativity rule

(c) Associativity rule in a context

Figure 1 Network rewrite rules (that can be troublesome to order)

coarity) together with two composition operations $\circ$ and $\otimes$, and a mapping ϕ of permutations to PROP elements. One PROP that elegantly illustrates the syntactic constraints of the PROP concept is that which takes as underlying set the set $\mathcal{R}^{\bullet\times\bullet}$ of all matrices over some (semi)ring $\mathcal{R}$: the arity is then the number of columns, the coarity is the number of rows, the $\circ$ operation is matrix multiplication (not defined unless the arity of the left factor equal the coarity of the right factor, just like composition of morphisms in a category), the image of a permutation is the corresponding permutation matrix, and the $\otimes$ operation constructs a larger matrix with the two operands as blocks, like $A \otimes B = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$. PROPs also need to satisfy a number of axioms, but it would take too long time to state those here; suffices it to say that they are equivalent to the claim that networks can serve as expressions for PROPs [1, Th. 5.17].

This last point may also be stated as the claim that the set of all networks (or rather isomorphism classes of networks) on a given alphabet constitutes the free PROP with respect to that alphabet. The $\circ$ operation then amounts to joining the outputs of the right operand to the inputs of the left operand, whereas $\otimes$ simply places the operands side-by-side, exposing each input and output of either operand as an input or output of the combined network. The network corresponding to a permutation σ consists only of edges from 1 to 0, the j'th outgoing edge at 1 also being the $\sigma(j)$'th incoming edge at 0. Formalised this way, the reason that an arity- and coarity-preserving function f from a doubly ranked set X to a PROP $\mathcal{P}$ gives rise to an evaluation map eval_f from the set of all networks on X to $\mathcal{P}$ is that eval_f is the unique morphism from the free PROP to $\mathcal{P}$ whose existence is guaranteed by the universal property.

2 The biaffine PROP

One slightly nontrivial PROP is the *biaffine PROP* $\mathrm{Baff}(\mathcal{R})$, which can be defined over any associative (semi)ring $\mathcal{R}$ with unit. The name can be understood as hinting at the fact that the matrix PROP $\mathcal{R}^{\bullet\times\bullet}$ defined above can be described as a PROP of linear transformations. If each PROP element in addition to the matrix part also gets a translation part, then one could make a PROP of *affine* transformations (an element of arity n and coarity m maps an n-dimensional space into an m-dimensional space). The biaffine PROP does that too, but goes further to preserve the symmetry of input and output. A rank (m, n) element of the biaffine PROP $\mathrm{Baff}(\mathcal{R})$ consists of four parts: an $m \times n$ matrix A, an $m \times 1$ vector $\mathbf{b}$, a $1 \times n$ vector $\mathbf{c}$, and a scalar d, wherein all elements are from the (semi)ring $\mathcal{R}$. It is often convenient to place these parts as blocks into an $(m+2) \times (n+2)$ matrix as follows

$$\begin{bmatrix} 1 & d & \mathbf{c} \\ 0 & 1 & 0 \\ 0 & \mathbf{b} & A \end{bmatrix} \tag{1}$$

since the composition $\circ$ is then ordinary matrix multiplication. The image under ϕ of a permutation has the matrix part A equal to the permutation matrix but the other three parts zero. The $\otimes$ operation is given by

$$\begin{bmatrix} 1 & d_1 & \mathbf{c}_1 \\ 0 & 1 & 0 \\ 0 & \mathbf{b}_1 & A_1 \end{bmatrix} \otimes \begin{bmatrix} 1 & d_2 & \mathbf{c}_2 \\ 0 & 1 & 0 \\ 0 & \mathbf{b}_2 & A_2 \end{bmatrix} := \begin{bmatrix} 1 & d_1 + d_2 & \mathbf{c}_1 & \mathbf{c}_2 \\ 0 & 1 & 0 & 0 \\ 0 & \mathbf{b}_1 & A_1 & 0 \\ 0 & \mathbf{b}_2 & 0 & A_2 \end{bmatrix}.$$

It is always technically possible to decompose a network into simpler networks using $\circ$ and $\otimes$, and through such a decomposition calculate its value in a particular PROP, but it is often inconvenient to do so. In the biaffine PROP, it is fairly straightforward to evaluate a network without that detour over $\circ$ and $\otimes$. To do this, each inner vertex v of the network should first have been assigned a corresponding biaffine PROP element (usually the value of the symbol at the vertex) with parts $A(v)$, $\mathbf{b}(v)$, $\mathbf{c}(v)$, and $d(v)$. Proceeding from input side to output side (the converse is equally possible), one calculates for each edge (i) a row of an intermediate A matrix, and (ii) an element of an intermediate $\mathbf{b}$ vector. Denote by $A^-(v)$ the matrix obtained by stacking the rows assigned to the incoming edges at vertex v, in the order of those edges at that vertex, and similarly denote by $A^+(v)$ the matrix obtained by stacking the rows assigned to outgoing edges at that vertex. Then at the input vertex 1 initialise $A^+(1) = I$ and at each inner vertex v let $A^+(v) = A(v)A^-(v)$; the matrix part of the value of the network as a whole is then the $A^-(0)$ matrix of the output vertex 0. If one similarly denotes by $\mathbf{b}^-(v)$ and $\mathbf{b}^+(v)$ respectively the vectors obtained by combining the vector elements assigned to the incoming and outgoing edges at vertex v, then $\mathbf{b}^+(1) = 0$ and $\mathbf{b}^+(v) = \mathbf{b}(v) + A(v)\mathbf{b}^-(v)$ at each inner vertex v, with $\mathbf{b}^-(0)$ being the $\mathbf{b}$ part of the value of the network. The $\mathbf{c}$ and d parts of the value of the network as a whole are then

$$\mathbf{c} = \sum_{\text{inner vertex } v} \mathbf{c}(v)A^-(v), \qquad d = \sum_{\text{inner vertex } v} \big(d(v) + \mathbf{c}(v)\mathbf{b}^-(v)\big).$$

Yet another way to understand at least the biaffine PROP $\mathrm{Baff}(\mathbb{N})$ is as a generalised path-counting device. Consider the element of $\mathrm{Baff}(\mathbb{N})$ to which a particular network evaluates. In the matrix part A, element $A_{i,j}$ then keeps track of the number of paths from input leg j to output leg i. In the vector parts $\mathbf{b}$ and $\mathbf{c}$, element b_i keeps track of the number of paths which begin somewhere inside the network and leave through output leg i whereas element c_j keeps track of the number of paths which enter through input leg j and end somewhere inside the network, and the scalar part d keeps track of the number of paths which both begin an end inside the network. It is easily checked that that the definitions of $\circ$, $\otimes$, and permutations in $\mathrm{Baff}(\mathbb{N})$ are consistent with this interpretation. What makes it a *generalised* path-counting device is that the PROP elements assigned to the individual vertices need not reflect the number of paths in the actual DAG underlying the network.

3 Network rewriting and PROP orders

When formalising, and in particular *automating*,[1] network rewriting, it becomes necessary to somehow order the networks so that no rewrite cycles arise. Defining orders that take the graph-theoretic structure of a network into account has however turned out to be surprisingly difficult, so the point of this text is to summarise the solutions that the author has found, and to point out some of the difficulties that one encounters.

[1] See `http://www.mdh.se/ukk/personal/maa/lhm03/sw/rewriting` for one utility that does this.

What is easy to do is to count vertices carrying a particular symbol, and order by that. This corresponds quite directly to ordering by (weighted) degree of polynomial, but that rarely gets one all the way, and there are even cases in which the intuitive rewrite direction may cause the number of vertices to increase (Figure 1a).

Similarly counting edges is not at all straightforward, as illustrated in Figure 1b: one may think that the purpose of this rewrite rule is to eliminate instances of a  vertex as the right child of another, and in a way it is, but one cannot state this goal simply as decreasing the number of edges from the output of a vertex to the right input of another vertex. Applying the rule of Figure 1b clearly consumes such an edge, but the catch is that it can also create another such an edge, as shown in Figure 1c; the rule is being applied to the bottom two vertices. The problem with 'count two-vertex subgraphs of the  form' is that this quantity does not change deterministically when a rule is placed in a context. It is possible to get somewhere with this ordering idea, but it requires keeping track of more than just the number of edges where the rule might apply, and in the end it turns out that the construction can be expressed more succintly in terms of the biaffine PROP Baff($\mathbb{N}$).

A *PROP quasi-order* is a transitive and reflexive relation $\leqslant$ on a PROP $\mathcal{P}$ such that $a_1 \leqslant a_2$ and $b_1 \leqslant b_2$ implies $a_1 \circ b_1 \leqslant a_2 \circ b_2$ (whenever those compositions are defined) and $a_1 \otimes b_1 \leqslant a_2 \otimes b_2$. The order is said to be *strict* if $\circ$ and $\otimes$ preserve strictness of inequalities. Given any PROP with a strict PROP quasi-order, one can pull that order back to the free PROP of networks via an evaluation map eval_f, and thereby define a strict PROP quasi-order on the networks. Because the direction of inequalities with respect to such an order is preserved when using $\circ$ and $\otimes$ to embed a network as a subexpression of a larger network, a network rewrite rule $l \to r$ where $r < l$ with respect to such an order will remain consistently oriented no matter what context C it gets placed into, as it will follow that $C(r) < C(l)$. (Technically, in the case of the rewriting machinery of [1], it is also necessary that the order has the *strict uncut property* [1, pp. 152–157], but that has so far never emerged as an obstacle.)

What makes the biaffine PROP Baff($\mathbb{N}$) useful here is that the element-wise partial order on it (standard matrix order, if one considers the block matrix form (1) for an element) is a well-founded PROP quasi-order, and if one restricts to matrices with at least one positive element in each row and each column (again as with respect to the block matrix form) then this quasi-order will be strict. An assignment f that is useful for the rule in Figure 1b is

$$f\left(\text{  }\right) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

(network has coarity 1 and arity 2, so the element of Baff($\mathbb{N}$) it is mapped to must have that as well, and with the padding of the block matrix form that comes out as 3×4)

making

$$\mathrm{eval}_f\left(\left[\text{}\right]\right) = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} > \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} = \mathrm{eval}_f\left(\left[\text{}\right]\right).$$

An assignment that is useful for the rule in Figure 1a is

$$g\left(\text{}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}, \qquad g\left(\text{}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}^{\mathrm{T}}$$

as that will have the d part of eval_g of the left hand side of Figure 1a come out as 1, but the d part of eval_g of the right hand side come out as 0, which suffices for a strict inequality; all other parts come out equal.

(a) Given rule (orientation of the equation itself) (b) First derived rule

Figure 2 Rewrite system based on the Yang–Baxter equation

The nice thing about using the biaffine PROP for ordering networks is that it turns out to be very versatile. What is perhaps a bit worrying is that there seems to be few known examples of going beyond the biaffine PROP. Lafont [2, p. 300] sets out with a seemingly more general construction of an order, but upon closer examination it turns out that the functions used must satisfy some additional condition in order for everything to fit together, and if that condition is to be polynomials of degree at most one then we are back to a special case of the biaffine PROP. The only example of a PROP order genuinely distinct from what the biaffine PROP can produce that is known to the author is instead the connectivity PROP of [1, Ex. 3.3], which embellishes the cyclomatic number of the underlying graph.

So far, neither of these have been of much use when completing the rewrite system consisting of the rule in Figure 2a (this *braid identity* constitutes an abstract form of the Yang–Baxter equation, and is mentioned as an example in e.g. [5]). In the case of the biaffine PROP, choosing an order of the networks amounts to picking a value for the  vertices, i.e., to assign values to the elements of the corresponding (1) matrix; there are nine elements whose values are not fixed, and these may be taken as variables parametrising the space of binaffine PROP-based orderings of networks with only  vertices. The claim that a particular rule is oriented in a particular direction gives rise to a system of polynomial inequalities in those nine variables. Considering only the rule of Figure 2a, that system has a solution with strict inequality. Completion will however immediately proceed to derive the rule in Figure 2b (by vertical symmetry of the first rule, either orientation of the derived rule is possible), and if adding also the inequalities resulting from that comparison, there is no longer a solution with strict inequality; the biaffine PROP fails to distinguish one side of a rule as strictly larger than the other. Switching to $\mathrm{Baff}(\mathcal{R})$ for a more general semiring $\mathcal{R}$ has been tried, but so far without much luck.

References

1. Lars Hellström. *Network Rewriting I: The Foundation*, 2012. arXiv:1204.2421v1 [math.RA].
2. Yves Lafont. Towards an algebraic theory of Boolean circuits. *J. Pure Appl. Algebra* **184** (2003), 257–310.
3. Saunders MacLane. Categorical Algebra. *Bull. Amer. Math. Soc.* **71** (1965), 40–106.
4. Shahn Majid. Cross Products by Braided Groups and Bosonization. *J. Algebra* **163** (1994), 165–190.
5. Samuel Mimram. *Computing Critical Pairs in 2-Dimensional Rewriting Systems.* arXiv:1004.3135v1 [cs.FL].
6. Roger Penrose. Applications of negative dimensional tensors. In Dominic J.A. Welsh, editor, *Combinatorial Mathematics and its Applications*, pages 211–244. Academic Press, 1971.