Commutative monoid formalism for weighted coupled cell networks and invariant synchrony patterns*

Pedro Sequeira†, A. Pedro Aguiar†, and João P. Hespanha‡

Abstract. This paper presents a framework based on matrices of monoids for the study of coupled cell networks. We formally prove within the proposed framework, that the set of results about invariant synchrony patterns for unweighted networks also holds for the weighted case. Moreover, the approach described allows us to reason about any multi edge and multi edge-type network as if it was single edge and single edge-type. Several examples illustrate the concepts described. Additionally, an improvement of the coarsest invariant refinement (CIR) algorithm to find balanced partitions is presented that exhibits a worst-case complexity of $\mathbf{O}(|\mathcal{C}|^3)$, where $\mathcal{C}$ denotes the set of cells.

Key words. Coupled cell networks, Synchrony, Balanced partitions

AMS subject classifications. 34A34, 34C45

1. Introduction.

Networks are used to describe systems with multiple components called **nodes** or **cells**. These cells can be pairwise connected by **edges**, describing the effect that one cell has on the other. These edges can be either directed or undirected and can also have weights in order to parameterize their interaction.

Such networks are ubiquitous, be it in the natural world or in engineering. From an electronic circuit or the electric grid, to the neural networks in our brain, food webs or the spread of a virus in a pandemic, this is a fundamental structure to study.

A big step in understanding these structures was the realization that real-world networks show properties that are pervasive across very different domains of application, such as being 'small-world' [16] and having a 'scale-free' degree distribution [3], and the existence of 'motifs' [8]. All these properties are related to the structure of the network and would not appear in a random one. Reviews on these types of statistical properties and their use in real-world applications are presented in [10], [4].

There are networks in which synchrony between the different cells is of the utmost importance [15]. Some examples are the cardiac pacemaker cells responsible for our heartbeat, the flashing of a swarm of fireflies, the consensus problem in control theory and the different gaits in animal locomotion generated by 'central pattern generators' (**CPG**). There are, however, situations in which too much synchronism is actually undesirable, such as in epileptic seizures in the brain.

*Submitted to the editors at 21 of December of 2020.

Funding: This work was supported in part by the FCT Project IMPROVE (POCI-01-0145-FEDER-031823), funded by the FEDER Funds through the COMPETE2020 - POCI, in part by the National Funds (PIDDAC) and in part by the National Science Foundation under Grant No. ECCS-2029985. The work of P. Sequeira was supported by a Ph.D. Scholarship, grant SFRH/BD/119835/2016 from Fundação para a Ciência e a Tecnologia (FCT), Portugal (POCH program).

†Faculdade de Engenharia, Universidade do Porto, Portugal (pedro.sequeira@fe.up.pt, pedro.aguiar@fe.up.pt).

‡Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA (hespanha@ece.ucsb.edu).

The model most commonly used to describe synchronism is the Kuramoto model, which consists on a large set ($N \to \infty$) of simple oscillators that are weakly coupled in an all-to-all fashion. Some reviews on the Kuramoto and its variants can be found in [2], [5], [12]. This simplistic network structure is, however, in direct opposition to our interest in understanding how the structure of network constrains the function executed by a networked system. The approach in this paper allows us to analyze patterns of synchrony that result exclusively from the topology of the network, regardless of the details of the specific dynamics. We consider both continuous and discrete-time dynamical systems.

The theory of coupled cell networks (**CCN**) was first mathematically formalized in [14], [7] and [6]. In that work, the concept of **admissibility** was defined by the minimal properties that a function must obey to model a network. This formalism is based on groupoids of bijections between in-neighborhoods of cells. This line of work also introduced the notion of **quotient network**, which is a smaller network that describes the behavior of the original network when the state of a system is in an **invariant synchrony pattern**. This means that some cells are sharing the same state and will continue doing so. Some issues arose from the fact that this formalism assumed only single edges between each ordered pair of cells. However, a quotient network might not satisfy this assumption even if the original network of interest does. This issue was solved by the 'multiarrow formalism' developed in [7], which allows the existence of multiple edges between the same pair of cells and self-loops. This formalism has been used with simple integer weights, in which the weight is used to represent a number of identical 'unitary' edges in parallel. This particular case is the simplest weighted case scenario and the previous formalism happens to be able to cover it. In this paper we properly extend the theory to be able to deal with the general weighted case. This theory is also more general in the sense that it does not require the so called 'consistency condition', which is enforced by changing the original network into another one that contains the exact same synchrony patterns. However, this change is not invertible and one loses dynamical information when doing so. We find that such an artificial condition is not necessary.

In this paper, a formalism based on matrices of commutative monoids is introduced in section 2. This formalism allows us to extend the previous known results about CCN's to networks with weighted connections, with arbitrary amount of edges and edge types. We develop the concept of **oracle functions**, which allows us to evaluate the dynamics of different networks that are composed of the same cell types in a very systematic and self-consistent manner.

In section 3 we use the new and more general definitions of admissibility to extend the previous known results about balanced partitions and invariant synchrony spaces.

In section 4 we focus on the particular case where the output set of the admissible functions is a vector space. Furthermore, we provide results in terms of local robustness that apply for this type of spaces.

Section 5 verifies that the connection between quotient networks and synchrony spaces given by balanced partitions work as expected in the general framework.

In section 6 we prove that the lattice structure of balanced partitions is the same as usual. Here, the join operation ($\vee$) is proved in a novel, algebraic way, instead of the usual duality argument between balanced partitions and invariant subspaces.

In section 7 we propose a novel CIR algorithm for arbitrary weights which has a worst-case time complexity of $\mathbf{O}(|\mathcal{C}|^3)$ instead of $\mathbf{O}((|\mathcal{E}| + |\mathcal{C}|)^4)$ as in [1], where $\mathcal{C}$ and $\mathcal{E}$ denote the sets

of cells and edges, respectively.

2. Weighted multi-edge CCN's. In this section we describe a formalism based on a matrix of monoids to represent networks and show that the previous known results about CCN's can be extended to networks with multi edge, multi edge-type, weighted connections.

2.1. CCN formalism. We start by introducing the definition of a cell coupled network according to the general weight formalism of this paper.

Definition 2.1. *A network $\mathcal{G}$ consists on a set of cells $\mathcal{C}_\mathcal{G}$, where each cell has a type, given by an index set $T = \{1, \dots, |T|\}$ according to $\mathcal{T}_\mathcal{G}\colon\ \mathcal{C}_\mathcal{G} \to T$ and has an $|\mathcal{C}_\mathcal{G}| \times |\mathcal{C}_\mathcal{G}|$ in-adjacency matrix $M_\mathcal{G}$. The entries of $M_\mathcal{G}$ are elements of a family of commutative monoids $\{\mathcal{M}_{ij}\}_{i,j\in T}$ such that $[M_\mathcal{G}]_{cd} = m_{cd} \in \mathcal{M}_{ij}$, for any cells $c, d \in \mathcal{C}_\mathcal{G}$ with types $i = \mathcal{T}_\mathcal{G}(c)$, $j = \mathcal{T}_\mathcal{G}(d)$.* □

We will write the monoid operation and the identity element of $\mathcal{M}_{ij}$ as $\|_{ij}$ and 0_{ij} respectively. The entries of m_{cd} are able to encode the complete connectivity (multi edge, multi edge-type) of the directed edges from d to c. This is thanks to the algebraic structure of the commutative monoid which we illustrate in the following section.

Remark 2.2. The subscripts $_\mathcal{G}$ are omitted when the network of interest is clear from context. □

2.2. Commutative monoids. The commutative monoid is the simplest algebraic structure that can be used to describe arbitrary finite parallels of edges. In this paper, we denote the monoid 'sum' operation by $\|$, due to the context in which it is used, with the meaning of 'adding in parallel'. Nevertheless, it is convenient to think of this as a sum. Likewise, the notation $\sum$ is used to describe parallels of multiple edges. In this context, the element $0_\mathcal{M} \in \mathcal{M}$ should be interpreted as 'no edge'.
The commutative monoid is associative and commutative. This reflects the fact that, for any given set of edges in parallel, it is irrelevant the order in which we enumerate the individual edges. Those are exactly the properties that provide invariance to this symmetry.
Note that we do not impose the existence of inverse elements since it is not guaranteed that we can cancel the effect of a set of edges by adding more edges to it in parallel.
We now show how a commutative monoid can be explicitly constructed using what is called a **presentation**.
The first step is to create a **free commutative monoid**. Given a set $\mathbb{W}$, that describes elemental edges, the free commutative monoid on $\mathbb{W}$ is $\mathcal{W} = (\mathbb{W}^*, \|_f)$, where $\mathbb{W}^*$ is the set of all finite multisets of the elements of $\mathbb{W}$, which represents all possible finite parallels of edges. Here, $\|_f$ encodes the multiset sum (free sum) and the element $0_\mathcal{W}$ is the empty multiset. Note that the set $\mathbb{W}$ itself does not need to be finite, or even countable.
At this point, the structure is certainly a commutative monoid. However, it is not yet capable of describing an arbitrary one. In particular, it is blind to the possibility of different sets of edges in parallel being equivalent (with regard to the application at hand). For instance, if we are working with the parallel of resistors, we would like to be able to encode into the structure the fact that $30\|15 = 20\|20$, from basic circuit theory.
In order to generalize this, the second step of the procedure is to quotient the free commutative monoid $\mathcal{W}$ over a **congruence relation** $\mathcal{R}$. A congruence relation on an algebraic structure

is an equivalence relation that is compatible with that structure. In our case, this means that we require $\mathcal{R}$ to be such, that the quotient $\mathcal{M} = \mathcal{W}/\mathcal{R}$ is a commutative monoid. Here, we think of the equivalence relation $\mathcal{R}$ as a function in $\mathbb{W}^* \to \mathbb{M}$ such that its level sets are the corresponding equivalence classes.
In order to satisfy the compatibility condition, we require that if $\mathcal{R}(a_1) = \mathcal{R}(a_2) = A$ and $\mathcal{R}(b_1) = \mathcal{R}(b_2) = B$ then $\mathcal{R}(a_1 \|_f b_1) = \mathcal{R}(a_2 \|_f b_2) = A \| B$, for any such $a_1, a_2, b_1, b_2 \in \mathbb{W}^*$. That is, for any equivalence classes, we can choose any of its elements as a representative, and when operating them ($\|_f$) the result should be exactly the same, which defines a consistent operation $\|$ on the equivalence classes.
Note that any commutative monoid has a presentation. Given a commutative monoid $\mathcal{M} = (\mathbb{M}, \|)$, we can create the free monoid $\mathcal{W} = (\mathbb{M}^*, \|_f)$. To this end, define the congruence relation $\mathcal{R}\colon \mathbb{M}^* \to \mathbb{M}$ such that for any element $w = w_1 \|_f \dots \|_f w_k$, with $w \in \mathbb{M}^*$ and $w_i \in \mathbb{M}$, $i \in \{1, \dots, k\}$, we have $\mathcal{R}(w) = w_1 \| \dots \| w_k$. Then, we have that $\mathcal{M} = \mathcal{W}/\mathcal{R}$.
We can also construct our commutative monoid of interest $\mathcal{M}$ using the set that describes the elemental edges $\mathbb{W}$ and defining the congruence relation $\mathcal{R}$ implicitly using a set of equations E. This can be written as $\mathcal{M} = \langle \mathbb{W} | E \rangle$. In the particular case of a free monoid, we write $\mathcal{M} = \langle \mathbb{W} | \rangle$. We illustrate these concepts with the following examples.

Example 2.3. Consider the commutative monoid generated by finite parallels of resistors. In this case, one has $\mathcal{M} = \langle \mathbb{W} | E \rangle$, with

$$\mathbb{W} = \mathbb{R}_0^+ \cup \{\infty\}$$

and

$$E = \begin{cases} w_1 \| w_2 = w_1 w_2 / (w_1 + w_2) & \forall w_1, w_2 \in \mathbb{W}\colon w_1 + w_2 \neq 0 \\ w_1 \| \infty = w_1 & \forall w_1 \in \mathbb{W} \\ 0 \| 0 = 0 \end{cases}$$

This allows us to verify that indeed $30 \| 15 = 20 \| 20$. In particular, those parallels are equivalent to an elemental edge of value 10. For the case of resistors, any set of parallel edges can be simplified into a single edge in $\mathbb{W}$. This is not true in general for an arbitrary commutative monoid.
The identity of this monoid is $0_{\mathcal{M}} = \infty$. Note that there is no element in $\mathcal{M}$, except for the identity $0_{\mathcal{M}}$ that has an inverse. That is, if there is a finite resistor w between two nodes, there is no resistor w^{-1} that we can add in parallel that will cancel it, that is $w \| w^{-1} = 0_{\mathcal{M}} = \infty$. □

Remark 2.4. Note that this formalism is extremely general. It allows us to parameterize individual edges with anything we might want, such as complex numbers, vectors, matrices, functions or any data structure as abstract as necessary. □

In Example 2.3 it can be seen that the zero-valued resistor, which is **not** the 'zero' of the monoid ($0_{\mathcal{M}}$), is an **annihilator**. That is, an element $a \in \mathcal{M}$ such that $w \| a = a$ for all $w \in \mathcal{M}$. Not every monoid has an annihilator, but if it exists, it is unique.

Example 2.5. Consider the commutative monoid $\mathcal{M} = (\mathbb{N}, \cdot)$, that is, the integers under the usual product, which has $0_{\mathcal{M}} = 1$. Define now the **free** monoid $\mathcal{N} = ((\{1\} \cup \mathbb{P})^*, \|_f)$,

where $\mathbb{P}$ is the set of prime numbers and $0_{\mathcal{N}} = 1$. Then, the fundamental theorem of arithmetic says that these monoids are two different ways of describing the exact same object. They are called **isomorphic**. This means that there is a bijective mapping $f\colon (\{1\} \cup \mathbb{P})^* \to \mathbb{N}$ that preserves the monoid structure (isomorphism). In particular, $f(p_1 \|_f p_2) = f(p_1) \cdot f(p_2)$ for all $p_1, p_2 \in (\{1\} \cup \mathbb{P})$ and $f(0_{\mathcal{N}}) = 0_{\mathcal{M}}$. We can find such an f by defining $f\left(\sum_{i=1}^{k} p_i\right) = \prod_{i=1}^{k} p_i$, in which $\sum$ is with regard to the multiset sum $\|_f$. This satisfies $f(1) = 1$ and the bijectivity comes from the uniqueness of prime factorization. □

Remark 2.6. Note that for the monoid $\mathcal{N}$ in Example 2.5, in opposition to the resistor case (Example 2.3), two elemental edges in parallel are almost never equivalent to another elemental edge. In fact, the only exception is the parallel with identity elements, for which this is inevitable. □

Example 2.7. The structure $\mathcal{M} = (\mathbb{R} \to \mathbb{R}, *)$, that is, the set of (generalized) functions together with the convolution operation forms a commutative monoid. Its identity is $0_{\mathcal{M}} = \delta(\cdot)$, the dirac delta distribution. □

Example 2.8. Consider a network with two types of elemental edges, each with its own commutative monoid structure. For instance, $\mathcal{M}_1 = (\mathbb{R}, +)$ and $\mathcal{M}_2 = (\mathbb{R} \to [-1, 1], \cdot)$.
We can merge them into a single commutative monoid by doing a direct product $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2$.
An element $m \in \mathcal{M}$ is an ordered pair $[m_1, m_2]$ such that $m_1 \in \mathbb{R}$ and $m_2 \in \mathbb{R} \to [-1, 1]$.
The operation $\|$ of the new monoid is then given by

$$w \| v = [w_1, w_2] \,\|\, [v_1, v_2] = [w_1 + v_1, w_2 \cdot v_2]$$

That is, the concatenation of applying the respective monoid operations to each component. The identity element of the new monoid is $0_{\mathcal{M}} = [0_{\mathcal{M}_1}, 0_{\mathcal{M}_2}] = [0, 1]$. □

This approach of constructing a commutative monoid $\mathcal{M}$ by merging smaller monoids that represent different edge-types, allows us to use a single monoid structure to fully describe the possible multi edge, multi edge-type connectivity between two cells.
Note that for each particular pair of cell types $i, j \in T$, we could have different monoid structures, which we denote as $\mathcal{M}_{ij}$, with respect to directed edges from cells of type j into cells of type i.
The network connectivity of the network can then be described by a single matrix whose entries are elements of the appropriate monoid.

2.3. Partition representations. In this paper we often refer to each class of a given partition on the set of cells $\mathcal{C}$ by the term **color**.
Consider a given partition $\mathcal{A}$ that divides a set of cells $\mathcal{C}$ into r colors. We can associate with each color an index from $\{1, \ldots, r\}$. Then, if a cell $c \in \mathcal{C}$ has the color associated with index k we say that $\mathcal{A}(c) = k$. This association allows us to represent the partition by saying that two cells $c, d \in \mathcal{C}$ have the same color if and only if $\mathcal{A}(c) = \mathcal{A}(d)$. We can think of this representation as a column vector or a function.
Another very useful representation is to define a partition matrix P of size $|\mathcal{C}| \times r$ such that $[P]_{ck} = 1$ if $\mathcal{A}(c) = k$ and $[P]_{ck} = 0$ otherwise.

Note that the same partition can be indexed by $\{1, \dots, r\}$ in different ways which will correspond to multiple partition matrices that are related to each other by a reordering of their columns.
The number of colors in a partition is called its **rank**, which in fact, corresponds to the rank of any of its matrix representations. That is, $r = rank(\mathcal{A}) = rank(P)$.
Given two partitions $\mathcal{A}$, $\mathcal{B}$ on a set of cells $\mathcal{C}$, we say that $\mathcal{A}$ is **finer** than $\mathcal{B}$ if for all $c, d \in \mathcal{C}$

$$\mathcal{A}(c) = \mathcal{A}(d) \implies \mathcal{B}(c) = \mathcal{B}(d) \tag{2.1}$$

which is denoted as $\mathcal{A} \leq \mathcal{B}$. Conversely, $\mathcal{B}$ is said to be **coarser** than $\mathcal{A}$. Roughly speaking, (2.1) means that if any pair of cells in partition $\mathcal{A}$ have the same color, then these two cells also have the same color in $\mathcal{B}$. In other words, if we merge some of the colors of $\mathcal{A}$ together, we can obtain $\mathcal{B}$. Conversely, we can obtain $\mathcal{A}$ by starting with $\mathcal{B}$ and splitting some of its colors into smaller ones.
The **trivial** partition, in which each individual cell has its own color is the **finest**, its rank is $|\mathcal{C}|$ and can be represented by any $|\mathcal{C}| \times |\mathcal{C}|$ permutation matrix, one of which is the identity.
We will often use the partition and its matrix interchangeably, that is, $P_{\mathcal{A}} \leq \mathcal{B}$ or $P_{\mathcal{A}} \leq P_{\mathcal{B}}$ to mean $\mathcal{A} \leq \mathcal{B}$.
If $\mathcal{A} \leq \mathcal{B}$, then there is a partition $\mathcal{B}/\mathcal{A}$ on the set of colors of $\mathcal{A}$ that describes how to merge them in order to achieve partition $\mathcal{B}$. That is, $(\mathcal{B}/\mathcal{A} \circ \mathcal{A})(c) = \mathcal{B}(c)$. Equivalently, for partition matrices such that $P_{\mathcal{A}} \leq P_{\mathcal{B}}$, there exists a partition matrix $P_{\mathcal{AB}}$, representing $\mathcal{B}/\mathcal{A}$ such that $P_{\mathcal{B}} = P_{\mathcal{A}} P_{\mathcal{AB}}$. The next example illustrates these concepts.

Example 2.9. Consider a set of cells $\mathcal{C} = \{a, b, c, d\}$ and partitions $\mathcal{A} = \{\{a, b\}, \{c\}, \{d\}\}$, $\mathcal{B} = \{\{a, b, c\}, \{d\}\}$. We have that $\mathcal{A} \leq \mathcal{B}$. Moreover, if one defines the characteristic matrices $P_{\mathcal{A}}, P_{\mathcal{B}}$ as

$$P_{\mathcal{A}} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_{\mathcal{B}} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

where the column number of each matrix corresponds to the index of the color that are arbitrarily assigned, then, there is a matrix $P_{\mathcal{AB}}$

$$P_{\mathcal{AB}} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

such that $P_{\mathcal{B}} = P_{\mathcal{A}} P_{\mathcal{AB}}$. Note that in $P_{\mathcal{AB}}$, the rows corresponds to the colors in $\mathcal{A}$, and the columns to the colors in $\mathcal{B}$, with the 1's describing the inclusion relationship between the different colorings.
These matrices correspond to a particular indexing such that the partitions can also be rep-

resented as the column vectors/functions

$$\mathcal{A} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \end{bmatrix} \quad \mathcal{B}/\mathcal{A} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \quad \mathcal{B} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

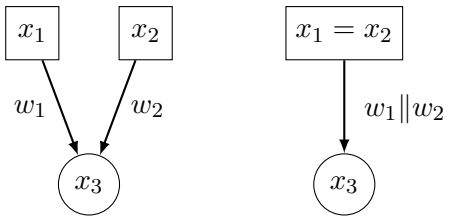
Note for example that $\mathcal{A}(4) = 3$ and $\mathcal{B}/\mathcal{A}(3) = 1$, that is, $(\mathcal{B}/\mathcal{A} \circ \mathcal{A})(4) = 1$ which is equal to $\mathcal{B}(4)$. □

Note that due to the particular structure of the characteristic matrices it is possible to multiply them together with matrices whose elements are not necessarily in the usual real/complex fields. For a characteristic matrix P and a given matrix M of appropriate dimensions, the product PM is always well-defined as an 'expansion' of M where its rows get replicated. If the product is MP, then it consists of sums of columns of M which requires a 'sum' operation to be defined on its elements (e.g., operations $\|_{ij}$ if M represents a network).

2.4. Admissibility. In this section, we describe the properties that a function $f\colon \mathbb{X} \to \mathbb{Y}$ has to respect in order to describe some **first-order property** of a network. Such a property, when evaluated at a particular cell $c \in \mathcal{C}$, is given by some $f_c\colon \mathbb{X} \to \mathbb{Y}_i$, with $i = \mathcal{T}(c)$ that is dependent only on the states of cells in the set $\{c \cup \mathcal{N}^-(c)\}$, where $\mathcal{N}^-(c)$ is the **in-neighborhood** of c, that is, $\mathcal{N}^-(c) = \{d \in \mathcal{C}\colon m_{cd} \neq 0_{ij}, i = \mathcal{T}(c), j = \mathcal{T}(d)\}$.
For this purpose, we impose on such functions two minimal assumptions (see Definition 2.10) that makes them behave as would be expected.
Note that these functions can be used to define measurements on a network, e.g., $\mathbf{y} = f(\mathbf{x})$, as well as to describe the evolution of dynamical systems, e.g., $\dot{\mathbf{x}} = f(\mathbf{x})$ or $\mathbf{x}^+ = f(\mathbf{x})$.
This does not mean that everything on a network has to (or can) be defined by such a function. For instance, the second derivative or the two-step evolution of the mentioned dynamical systems will not be of this form. Those functions will be 'second-order' in the sense that they are dependent on they first and second in-neighborhoods (neighbor of neighbor). They are, however, fully defined from the original first-order functions.
Consider the simple network of Figure 2.1a, (which could be part of a larger network) consisting on cell 3 and its in-neighborhood. We have cell types $T = \{1, 2\}$ which represent 'circle' and 'square' cells, respectively. We use $\|$ to mean $\|_{12}$ since it is the only monoid operation in this example. Since cells 1 and 2 are of the same cell type (square) ($\mathcal{T}(1) = \mathcal{T}(2) = 2$) and also, are currently in the same state ($x_1 = x_2$), the total input received by cell 3 at that instant, is the same as if both edges originated from a single 'square' cell with that state. Furthermore, if the weights cancel ($w_1 \| w_2 = 0_{12}$), then cell 3 should act as if cells 1, 2 are not there whenever $x_1 = x_2$. Moreover, the input received by a cell is independent of how we draw the network, that is, we do not expect f_3 to be different if cell 2 was at the left of cell 1. That is, f_3 should obey

$$f_3\left(x_3; \begin{bmatrix} w_1 & w_2 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = f_3\left(x_3; \begin{bmatrix} w_2 & w_1 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}\right)$$

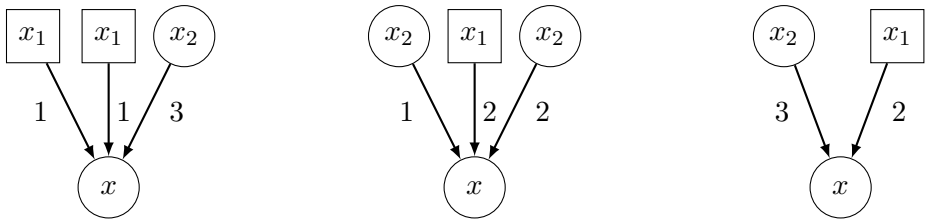
Extending these principles, if two cells c, d are in the same internal state $x_c = x_d = x$ and if the edge-compression argument can transform their in-neighborhoods into the same network,



(a) Original (b) Merged

Figure 2.1: Edge merging

then $f_c = f_d$. This is illustrated in Figure 2.2, with the monoid operation $\|$ being the usual addition. If we write the state and weight vectors from left to right, we get



(a) First input set (b) Second input set (c) Common edge compression to both input sets

Figure 2.2: Input equivalent networks

$$\mathbf{x}_{\mathcal{A}} = \begin{bmatrix} x_1 & x_1 & x_2 \end{bmatrix}^\top \qquad \mathbf{w}_{\mathcal{A}} = \begin{bmatrix} 1 & 1 & 3 \end{bmatrix}$$
$$\mathbf{x}_{\mathcal{B}} = \begin{bmatrix} x_2 & x_1 & x_2 \end{bmatrix}^\top \qquad \mathbf{w}_{\mathcal{B}} = \begin{bmatrix} 1 & 2 & 2 \end{bmatrix}$$
$$\overline{\mathbf{x}} = \begin{bmatrix} x_2 & x_1 \end{bmatrix}^\top \qquad \overline{\mathbf{w}} = \begin{bmatrix} 3 & 2 \end{bmatrix}$$

for each of the Figures 2.2a to 2.2c, respectively. The process of edge-merging the in-neighborhoods from Figures 2.2a and 2.2b into Figure 2.2c can be described through the partition matrices $P_{\mathcal{A}}, P_{\mathcal{B}}$

$$P_{\mathcal{A}} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad P_{\mathcal{B}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

such that

$$\mathbf{x}_{\mathcal{A}} = P_{\mathcal{A}} \overline{\mathbf{x}}, \quad \mathbf{x}_{\mathcal{B}} = P_{\mathcal{B}} \overline{\mathbf{x}}$$
$$\mathbf{w}_{\mathcal{A}} P_{\mathcal{A}} = \overline{\mathbf{w}} = \mathbf{w}_{\mathcal{B}} P_{\mathcal{B}}$$

where we considered that the edge merging can only be done with edges that come from cells of the same type in the same state. That is, we have also assumed implicitly that

$$\mathcal{T}_{\mathcal{A}} = P_{\mathcal{A}}\overline{\mathcal{T}}, \quad \mathcal{T}_{\mathcal{B}} = P_{\mathcal{B}}\overline{\mathcal{T}}$$

where $\mathcal{T}_{\mathcal{A}}$, $\mathcal{T}_{\mathcal{B}}$ and $\overline{\mathcal{T}}$ describe the typings of the corresponding cells, that is,

$$\mathcal{T}_{\mathcal{A}} = \begin{bmatrix}2 & 2 & 1\end{bmatrix}^{\top} \qquad \mathcal{T}_{\mathcal{B}} = \begin{bmatrix}1 & 2 & 1\end{bmatrix}^{\top} \qquad \overline{\mathcal{T}} = \begin{bmatrix}1 & 2\end{bmatrix}^{\top}$$

This type checking can be omitted if declare that, by definition, the states of two cells can only be compared in the first place if they are of the same type. That is, if P satisfies this assumption in $\mathbf{x} = P\overline{\mathbf{x}}$, there will be no danger of trying to 'sum' elements of different monoids in $\mathbf{w}P = \overline{\mathbf{w}}$.
This edge-merging example motivates the following definitions.

Definition 2.10. *Consider the set of cell types T, and the related sets $\{\mathbb{X}_j, \mathbb{Y}_j\}_{j\in T}$ together with a family of commutative monoids $\{\mathcal{M}_{ij}\}_{j\in T}$, for a given fixed $i \in T$. Take a function $\hat{f}_i$ defined on*

$$\hat{f}_i \colon \mathbb{X}_i \times \overset{\circ}{\bigcup_{\mathbf{k}\in\mathbb{N}_0^{|T|}}} \left(\mathcal{M}_i^{\mathbf{k}} \times \mathbb{X}^{\mathbf{k}}\right) \to \mathbb{Y}_i \tag{2.2}$$

where $\overset{\circ}{\bigcup}$ denotes the disjoint union and $\mathbf{k} = \left[k_1, \ldots, k_{|T|}\right]$ is a multi-index with $k_i \in \mathbb{N}_0$ for all $i \in T$ such that $\mathbb{X}^{\mathbf{k}} := \mathbb{X}_1^{k_1} \times \ldots \times \mathbb{X}_{|T|}^{k_{|T|}}$ and $\mathcal{M}_i^{\mathbf{k}} := \mathcal{M}_{i1}^{k_1} \times \ldots \times \mathcal{M}_{i|T|}^{k_{|T|}}$.
*The function $\hat{f}_i$ is called an **oracle component of type i**, if it has the property that for every $x \in \mathbb{X}_i$, $\mathbf{x} \in \mathbb{X}^{\mathbf{k}}$, $\mathbf{w} \in \mathcal{M}_i^{\mathbf{k}}$, $\overline{\mathbf{x}} \in \mathbb{X}^{\overline{\mathbf{k}}}$, $\overline{\mathbf{w}} \in \mathcal{M}_i^{\overline{\mathbf{k}}}$ and partition matrix P, that satisfy*

$$\begin{cases} \mathbf{x} = P\overline{\mathbf{x}} \\ \mathbf{w}P = \overline{\mathbf{w}} \end{cases} \tag{2.3}$$

then

$$\hat{f}_i(x; \mathbf{w}, \mathbf{x}) = \hat{f}_i(x; \overline{\mathbf{w}}, \overline{\mathbf{x}}) \tag{2.4}$$

Furthermore, if $\mathbf{w}$ has its k^{th} element (corresponding to cell c_k) equal to 0_{ij}, with $j = \mathcal{T}(c_k)$, then

$$\hat{f}_i(x; \mathbf{w}, \mathbf{x}) = \hat{f}_i(x; \mathbf{w}_{-k}, \mathbf{x}_{-k}) \tag{2.5}$$

where $\mathbf{w}_{-k}$, $\mathbf{x}_{-k}$ denotes the result of removing the k^{th} element of the original vectors $\mathbf{w}$, $\mathbf{x}$. In (2.3), equality between states assumes compatible cell types. □

The disjoint union allows us to distinguish neighborhoods of different types, that is, the set $\mathbb{X}_1 \times \mathbb{X}_1$ is always taken as a different set from $\mathbb{X}_1 \times \mathbb{X}_2$ even in the particular case of $\mathbb{X}_1 = \mathbb{X}_2$.

Remark 2.11. Note that the way the domain of $\hat{f}_i$ was defined allows us to deal with variable input set configurations. This is an equivalent, but cleaner way of defining a family of functions, each on a different domain based on its specific input set, such that they are all connected by the self-consistency rules that we expect from them. This way, we can use a single function to describe what really matters to us, that is, describing how a cell is affected by its in-neighbors. □

Remark 2.12. It is easy to verify that for a **permutation** matrix P that preserves cell typing ($\mathbb{X}_{\mathbf{k}} = P\mathbb{X}_{\mathbf{k}}$) we have

$$\hat{f}_i(x; \mathbf{w}, \mathbf{x}) = \hat{f}_i(x; \mathbf{w}P^\top, P\mathbf{x})$$

which is in accordance to the idea that a cell does not care about the order in which its input set is drawn. □

The **oracle set** is the set of all $|T|$-tuples of oracle components, such that each element of the tuple represents one of the types in T. It is denoted as

$$\hat{\mathcal{F}}_T = \prod_{i=1}^{|T|} \hat{\mathcal{F}}_i$$

where $\hat{\mathcal{F}}_i$ is the set of all oracle components of type i . We are always implicitly assuming sets $\{\mathbb{X}_i, \mathbb{Y}_i\}_{i\in T}$ and commutative monoids $\{\mathcal{M}_{ij}\}_{i,j\in T}$. Note that modeling some aspect of a network that follows our assumptions is effectively choosing one of the elements of $\hat{\mathcal{F}}_T$, which we call **oracle functions**.

Definition 2.13. *Consider a network $\mathcal{G}$ defined on a cell set $\mathcal{C}$ with cell types in T according to the cell type partition $\mathcal{T}$, and an in-adjacency matrix M. Assume without loss of generality that the cells are ordered according to the cell types such that we can associate with the network a state $\mathbb{X} := \mathbb{X}^{\mathbf{k}}$ and output $\mathbb{Y} := \mathbb{Y}^{\mathbf{k}}$ sets, with $|\mathcal{C}| = |\mathbf{k}|$*

$$|\mathbf{k}| = \sum_{i=1}^{|T|} k_i$$

and k_i is the number of cells in $\mathcal{C}$ of type $i \in T$.
A function $f\colon \mathbb{X} \to \mathbb{Y}$, given as

$$f = (f_c)_{c\in\mathcal{C}}, \quad \text{with } f_c\colon \mathbb{X} \to \mathbb{Y}_i, \quad i = \mathcal{T}(c)$$

*is said to be **$\mathcal{G}$-admissible** if there is some oracle function $\hat{f} \in \hat{\mathcal{F}}_T$, $\hat{f} = (\hat{f}_i)_{i\in T}$ such that*

$$f_c(\mathbf{x}) = \hat{f}_i\left(x_c; \mathbf{m}_c, \mathbf{x}\right) \tag{2.6}$$

for $\mathbf{x} \in \mathbb{X}$, where x_c is the c^{th} coordinate of $\mathbf{x}$ and $\mathbf{m}_c$ is the c^{th} row of matrix M. In this case we write $f = \hat{f}|_{\mathcal{G}}$. □

The set of all $\mathcal{G}$-admissible functions is denoted as $\mathcal{F}_\mathcal{G}$. It can be thought of as the result of evaluating $\hat{\mathcal{F}}_T$ at $\mathcal{G}$, which can be written as $\hat{\mathcal{F}}_T|_\mathcal{G}$. Note that process of evaluating oracle functions at a network is not necessarily injective. There might be oracle functions $\hat{f}, \hat{g} \in \hat{\mathcal{F}}_T$ with $\hat{f} \neq \hat{g}$ such that $\hat{f}|_\mathcal{G} = \hat{g}|_\mathcal{G}$.
The next example makes explicit the relation between the connectivity graph of a network and how that constraints any possible admissible function that acts on it.

Example 2.14. Figure 2.3 shows an example of a **CCN** of three cells of the same type. This **CCN** can be described by the in-adjacency matrix M

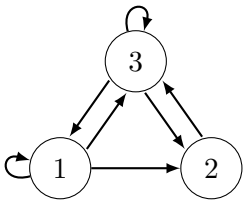


Figure 2.3: Simple network with admissible functions that have the structure given by (2.8)–(2.10)

$$M = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.7}$$

together with the cell type partition $\mathcal{T} = \{\{1, 2, 3\}\}$. This means that a suitable $f \in \mathcal{F}_\mathcal{G}$ should have the following structure

$$f_1(\mathbf{x}) = \hat{f}(x_1; \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \mathbf{x}) \tag{2.8}$$

$$f_2(\mathbf{x}) = \hat{f}(x_2; \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, \mathbf{x}) \tag{2.9}$$

$$f_3(\mathbf{x}) = \hat{f}(x_3; \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \mathbf{x}) \tag{2.10}$$

for some $\hat{f} \in \hat{\mathcal{F}}_T$. Note that here T only has one type, that is $\hat{\mathcal{F}}_\circ = \hat{\mathcal{F}}_T$. Therefore, we use $\hat{f}$ interchangeably as both oracle function and oracle component. It is exactly the same treatment as not differentiating between an one-dimensional vector and the element it contains. □

To make more explicit the importance of a rigorous definition for admissibility, the following example presents a case that might look reasonable at a first glance but ends up not being admissible.

Example 2.15. Consider the simple network in Figure 2.1 that was used to illustrate the edge merging concept. We will propose a function on the original network Figure 2.1a and verify if it satisfies our assumptions.
We consider that the cells have associated state and output sets given by $\mathbb{X}_1 = \mathbb{X}_2 = \mathbb{Y}_1 = \mathbb{R}$, such that $T = \{1, 2\}$ identify the cell types 'circle' and 'square' respectively.

The directed edges from 'square' into 'circle' are in $\mathcal{M}_{12}$ and we denote $\|_{12}$ by $\|$ for simplicity. Given functions $g\colon \mathbb{R} \to \mathbb{R}$ and $p\colon \mathcal{M}_{12} \to \mathbb{R}$, with $p(0_{12}) = 0$, it is tempting to think that the function f_3, could be modeled by

$$f_3(\mathbf{x}) = g(x_3) + p(w_1)x_1 + p(w_2)x_2 + p(w_1)p(w_2)x_1x_2 \tag{2.11}$$

After all, if we simultaneously switch $w_1 \leftrightarrow w_2$ and $x_1 \leftrightarrow x_2$, f_3 would still look the same. Consider, $w_1 = w$, $x_1 = x$ and $w_2 = 0_{12}$. Then, if cell 3 only had one neighbor (of type square), f_3 would be given by

$$f_3(\mathbf{x}) = g(x_3) + p(w)x$$

If we have $x_1 = x_2 = x_{12}$, from the edge-merging principle, we should be in the situation of Figure 2.1b. We would have

$$f_3(\mathbf{x}) = g(x_3) + p(w_1\|w_2)x_{12}$$

However, from direct substitution on (2.11) we obtain

$$f_3(\mathbf{x}) = g(x_3) + (p(w_1) + p(w_2) + p(w_1)p(w_2)x_{12})x_{12}$$

which means that this is not admissible since

$$p(w_1\|w_2) = p(w_1) + p(w_2) + p(w_1)p(w_2)x_{12}$$

goes against the assumption that p depends only on the edge weights.
Consider that instead f_3 was modeled as

$$f_3(\mathbf{x}) = g(x_3) + p(w_1)x_1 + p(w_2)x_2 + p(w_1)p(w_2)\frac{x_1 + x_2}{2}$$

Following the exact same approach this requires

$$p(w_1\|w_2) = p(w_1) + p(w_2) + p(w_1)p(w_2)$$

which is a valid constraint. It only depends on its inputs and is compatible with a commutative monoid structure, that is,

$$\begin{aligned} p(w\|0_{12}) &= p(w) \\ p(w_1\|w_2) &= p(w_2\|w_1) \\ p((w_1\|w_2)\|w_3) &= p(w_1\|(w_2\|w_3)) \end{aligned}$$

Note that for each of the three equalities, the inputs for both members are the same element of $\mathcal{M}_{12}$. The same input of a function has to output the same value. □

Remark 2.16. Assume that in the previous example there was an annihilator element in $\mathcal{M}_{12}$, that is, an element $a_{12} \in \mathcal{M}_{12}$ such that $w\|a_{12} = a_{12}$ for all $w \in \mathcal{M}_{12}$. Then, either $p(a_{12}) = -1$ or we are in the non interesting case where $p(w) = 0$ for all $w \in \mathcal{M}_{12}$.
An example of an annihilator is the short-circuit ($R = 0$) with regard to the parallel of resistors, as in Example 2.3. □

Remark 2.17. Note that we solved the problem of the 2-coupling component being quadratic on x_{12} when $x_1 = x_2 = x_{12}$ by modeling that component additively with $(x_1 + x_2)/2$. This is not the only approach. Consider now that we have some function $q\colon \mathcal{M}_{12} \times \mathbb{R} \to \mathbb{R}$ with $q(0_{12}, x) = 0$ such that we model f_3 by

$$f_3(\mathbf{x}) = g(x_3) + q(w_1, x_1) + q(w_2, x_2) + q(w_1, x_1)q(w_2, x_2)$$

This is also valid when

$$q(w_1 \| w_2, x) = q(w_1, x) + q(w_2, x) + q(w_1, x)q(w_2, x)$$

Similarly, this also compatible with a commutative monoid structure. □

3. Invariant synchrony. In subsection 2.4 we introduced the concept of oracle components $\hat{f}_i \in \hat{\mathcal{F}}_i$ which describe the way a cell of type $i \in T$ reacts to its input set in a way that is self-consistent. That is, having the same state and equivalent inputs generates the same output and the output is only functionally dependent on the in-neighbors.
The functional modeling of a network $\mathcal{G}$ should then be though of as choosing an oracle function $\hat{f} \in \hat{\mathcal{F}}_T$ and constraining it to $\mathcal{G}$ as in Definition 2.13. This gives us an $\mathcal{G}$-admissible function $f \in \mathcal{F}_\mathcal{G}$ such that $f = \hat{f}|_\mathcal{G}$.
The oracle components, which are the core concept of the modeling of a network were defined in terms of equality relationships. This motivates the study of whether a set of equality relations in the state set $\mathbb{X}$ are preserved by f regardless of the specific admissible f that acts on the network.

Definition 3.1. *Given a partition $\mathcal{A} \leq \mathcal{T}$ we call the subset of $\mathbb{X}$*

$$\Delta_\mathcal{A}^\mathbb{X} = \{\mathbf{x} \in \mathbb{X}\colon \mathcal{A}(c) = \mathcal{A}(d) \implies x_c = x_d\} \tag{3.1}$$

*the **polydiagonal** of $\mathcal{A}$ in $\mathbb{X}$.* □

This means that any $\mathbf{x} \in \Delta_\mathcal{A}^\mathbb{X}$ can be given by $\mathbf{x} = P\overline{\mathbf{x}}$ for some $\overline{\mathbf{x}}$, where P is a characteristic matrix of $\mathcal{A}$. Note that

$$\mathcal{A} \leq \mathcal{B} \iff \Delta_\mathcal{A}^\mathbb{X} \supseteq \Delta_\mathcal{B}^\mathbb{X} \tag{3.2}$$

Definition 3.2. *If for a function $f\colon \mathbb{X} \to \mathbb{Y}$ and a partition $\mathcal{A} \leq \mathcal{T}$ we have*

$$f\left(\Delta_\mathcal{A}^\mathbb{X}\right) \subseteq \Delta_\mathcal{A}^\mathbb{Y} \tag{3.3}$$

*then f is $\mathcal{A}$-**invariant**.* □

Note that if f is $\mathcal{A}$-invariant, then for every $\mathbf{x} \in \mathbb{X}$ such that $\mathbf{x} = P\overline{\mathbf{x}}$, with P representing $\mathcal{A}$, there is $\overline{\mathbf{y}}$ such that $f(P\overline{\mathbf{x}}) = P\overline{\mathbf{y}}$. This means that there is a function $\overline{f}\colon \overline{\mathbb{X}} \to \overline{\mathbb{Y}}$ with $\mathbb{X} = P\overline{\mathbb{X}}$, $\mathbb{Y} = P\overline{\mathbb{Y}}$ such that

$$f(P\overline{\mathbf{x}}) = P\overline{f}(\overline{\mathbf{x}}) \tag{3.4}$$

Consider a discrete-time system $\mathbf{x}^+ = f(\mathbf{x})$ that evolves according to an $\mathcal{G}$-admissible map $f\colon \mathbb{X} \to \mathbb{X}$. If f is $\mathcal{A}$-invariant, then

$$\mathbf{x}_{n_0} \in \Delta_{\mathcal{A}}^{\mathbb{X}} \implies \mathbf{x}_n \in \Delta_{\mathcal{A}}^{\mathbb{X}} \quad \forall n \in \mathbb{N}\colon n \geq n_0 \tag{3.5}$$

Similarly, for a continuous-time system $\dot{\mathbf{x}} = f(\mathbf{x})$ that evolves according to an $\mathcal{G}$-admissible vector field $f(\mathbf{x})\colon \mathbb{X} \to T_{\mathbf{x}}\mathbb{X}$ where f is Lipschitz, $\mathbb{X}$ is a smooth manifold and $T_{\mathbf{x}}\mathbb{X}$ its tangent space at $\mathbf{x}$. If f is $\mathcal{A}$-invariant, then

$$\mathbf{x}(t_0) \in \Delta_{\mathcal{A}}^{\mathbb{X}} \implies \mathbf{x}(t) \in \Delta_{\mathcal{A}}^{\mathbb{X}} \quad \forall t \in \mathbb{R} \tag{3.6}$$

Note that in both cases the polydiagonals $\Delta_{\mathcal{A}}^{\mathbb{X}}$ are invariant with respect to the dynamics. Moreover, the evolution of $\mathbf{x}$ is fully determined by $\overline{\mathbf{x}}$, which in turn evolves according to

$$\overline{\mathbf{x}}^+ \backslash \dot{\overline{\mathbf{x}}} = \overline{f}(\overline{\mathbf{x}}) \tag{3.7}$$

The following concept relates the structure of a network with its capability to preserve a given synchrony pattern. This works by verifying whether, for a given coloring of the network, if cells that have the same color also have colored input sets that are equivalent.

Definition 3.3. *Consider a network $\mathcal{G}$ defined on a cell set $\mathcal{C}$ with a cell type partition $\mathcal{T}$ and an in-adjacency matrix M. A partition $\mathcal{A} \leq \mathcal{T}$ with characteristic matrix P is said to be* ***balanced*** *on $\mathcal{G}$ if for all $c, d \in \mathcal{C}$*

$$\mathcal{A}(c) = \mathcal{A}(d) \implies \mathbf{m}_c P = \mathbf{m}_d P \tag{3.8}$$

where $\mathbf{m}_c, \mathbf{m}_d$ are the c^{th} and d^{th} rows of matrix M. □

A balanced partition is usually indicated with the symbol $\bowtie$. Note that a partition is balanced if and only if there is a matrix Q of elements in the appropriate monoids $\{\mathcal{M}_{ij}\}_{i,j\in T}$ such that

$$MP = PQ \tag{3.9}$$

Clearly, the trivial partition is always balanced. That is, for any M, the condition (3.9) is satisfied with $P = I$ and $Q = M$. We now show the following result.

Theorem 3.4. *Consider $\mathcal{F}_{\mathcal{G}}$, defined on a network $\mathcal{G}$ with sets $\{\mathbb{X}_i, \mathbb{Y}_i\}_{i\in T}$ and commutative monoids $\{\mathcal{M}_{ij}\}_{i,j\in T}$.*
If a partition $\bowtie$ on $\mathcal{G}$ is balanced, then every $f \in \mathcal{F}_{\mathcal{G}}$ is $\bowtie$-invariant. □

Proof. Consider $\mathbf{x} \in \Delta_{\bowtie}^{\mathbb{X}}$, that is, $\mathbf{x} = P\overline{\mathbf{x}}$ for some $\overline{\mathbf{x}}$. From the balanced definition (3.8) we have $\bowtie(c) = \bowtie(d) \implies \mathbf{m}_c P = \mathbf{m}_d P$, which satisfies conditions (2.3) and therefore from the definition of admissibility $f_c(P\overline{\mathbf{x}}) = f_d(P\overline{\mathbf{x}})$. This means that there is a $\overline{f}$ such that $f(P\overline{\mathbf{x}}) = P\overline{f}(\overline{\mathbf{x}})$ and every $\mathcal{G}$-admissible function $f \in \mathcal{F}_{\mathcal{G}}$ is $\bowtie$-invariant. ■

The unweighted version of this theorem is presented in [14], [7] and [6] as an equivalence. Note however, that although the forward direction is a strong result ($\bowtie$ balanced $\implies$ f is $\bowtie$-invariant $\forall f \in \mathcal{F}_{\mathcal{G}}$, for any $\mathcal{F}_{\mathcal{G}}$), the backwards one is extremely weak by comparison, requiring every single $\mathcal{G}$-admissible function to be $\bowtie$-invariant. For this reason, we present the backwards direction separately.

Theorem 3.5. *Consider $\mathcal{F}_\mathcal{G}$, defined on a network $\mathcal{G}$ with sets $\{\mathbb{X}_i, \mathbb{Y}_i\}_{i\in T}$ and commutative monoids $\{\mathcal{M}_{ij}\}_{i,j\in T}$.*
For a partition $\bowtie$ on $\mathcal{G}$, if every $f \in \mathcal{F}_\mathcal{G}$ is $\bowtie$-invariant, then $\bowtie$ is balanced. □

Proof. Assume $\bowtie$ is not balanced. Then, there are $\bowtie(c) = \bowtie(d)$ such that $\mathbf{m}_c P \neq \mathbf{m}_d P$. Consider k one of the colors in which they differ. That is, $[\mathbf{m}_c P]_k \neq [\mathbf{m}_d P]_k$.
Moreover, choose some state $\mathbf{x} \in \Delta_\bowtie^\mathbb{X}$, that is, $\mathbf{x} = P\overline{\mathbf{x}}$, such that $\overline{x}_k$ is different from all other entries of $\overline{\mathbf{x}}$.
Then, there is an $f \in \mathcal{F}_\mathcal{G}$, that is, $f = \hat{f}|_\mathcal{G}$ such that $\hat{f}_i(x; \mathbf{w}, \mathbf{x}) = y_1$ if $\mathbf{w}$, summed over the entries such that $\mathbf{x}$ is $\overline{x}_k$, results into $[\mathbf{m}_c P]_k$, and $\hat{f}_i(x; \mathbf{w}, \mathbf{x}) = y_2$ otherwise, with $y_1 \neq y_2$, $y_1, y_2 \in \mathbb{Y}_i$ and $i = \mathcal{T}(c)$.
Then, we have an $f \in \mathcal{F}_\mathcal{G}$ and $\overline{\mathbf{x}}$ such that $f(P\overline{\mathbf{x}}) \notin \Delta_\bowtie^\mathbb{Y}$. Therefore, the only way to not be able to find such an f, is for $\bowtie$ to be balanced. ∎

For the unweighted case, the authors of [14], [7] and [6] proved the backwards direction by actually proving a stronger result that they hid away in the proof.
Their result is stronger in the sense that they show that only a particular subset of $\mathcal{F}_\mathcal{G}$ is necessary to be $\bowtie$-invariant to enforce $\bowtie$ to be balanced. This works in the weaker framework of the unweighted case, which corresponds to the simple monoid structure $\mathcal{M} = (\mathbb{N}_0, +)$.
Nevertheless, a stronger result deserves to be shown in its own right. Therefore, in the next section, we present similar results for special cases of monoids that extends their previous result.

4. Output vector spaces. The following results apply when the output sets are vector spaces.

Lemma 4.1. *Consider $\hat{\mathcal{F}}_i$, defined on an output set $\mathbb{Y}_i$ that is a vector space. Then, $\hat{\mathcal{F}}_i$ is itself a vector space.* □

Proof. Consider components $\hat{f}_i, \hat{g}_i \in \hat{\mathcal{F}}_i$. Defining, $\hat{h}_i = \hat{f}_i + \hat{g}_i$, means that if conditions (2.3) are satisfied for $\hat{f}_i$ and $\hat{g}_i$, then

$$\begin{aligned}\hat{h}_i(x; \mathbf{w}, \mathbf{x}) &= \hat{f}_i(x; \mathbf{w}, \mathbf{x}) + \hat{g}_i(x; \mathbf{w}, \mathbf{x}) \\ &= \hat{f}_i(x; \overline{\mathbf{w}}, \overline{\mathbf{x}}) + \hat{g}_i(x; \overline{\mathbf{w}}, \overline{\mathbf{x}}) \\ &= \hat{h}_i(x; \overline{\mathbf{w}}, \overline{\mathbf{x}})\end{aligned}$$

which means that $\hat{h}_i$ satisfies condition (2.4). Condition (2.5) is verified in exactly the same way. Therefore, $\hat{h}_i \in \hat{\mathcal{F}}_i$.
If $\hat{f}_i$ satisfies (2.4) and (2.5) then $\alpha\hat{f}_i$ also satisfies it for any scalar α. Therefore $\alpha\hat{f}_i \in \hat{\mathcal{F}}_i$ and $\hat{\mathcal{F}}_i$ is a vector space. ∎

Corollary 4.2. *Consider $\hat{\mathcal{F}}_T$, defined on the output sets $\{\mathbb{Y}_i\}_{i\in T}$ that are vector spaces. Then, $\hat{\mathcal{F}}_T$ is itself a vector space.* □

Lemma 4.3. *Assume $\hat{\mathcal{F}}_T$ is a vector space. Evaluation on a network $(|_\mathcal{G})$ is a linear operator.* □

Proof. Consider oracle functions $\hat{f}, \hat{g} \in \hat{\mathcal{F}}_T$. Since $\hat{\mathcal{F}}_T$ is a vector space, there is a $\hat{h} \in \hat{\mathcal{F}}_T$

such that $\hat{h} = \hat{f} + \hat{g}$. Define $f = \hat{f}|_{\mathcal{G}}$ and $g = \hat{g}|_{\mathcal{G}}$. Then $h = \hat{h}|_{\mathcal{G}}$ is such that

$$\begin{aligned} h_c(\mathbf{x}) &= \hat{h}_i\left(x_c; \mathbf{m}_c, \mathbf{x}\right) \\ &= \hat{f}_i\left(x_c; \mathbf{m}_c, \mathbf{x}\right) + \hat{g}_i\left(x_c; \mathbf{m}_c, \mathbf{x}\right) \\ &= f_c(\mathbf{x}) + g_c(\mathbf{x}) \end{aligned}$$

for all $c \in \mathcal{C}$ with $i = \mathcal{T}(c)$. That is, $(\hat{f} + \hat{g})|_{\mathcal{G}} = \hat{f}|_{\mathcal{G}} + \hat{g}|_{\mathcal{G}}$.
Similarly, for any $\hat{f} \in \hat{\mathcal{F}}_T$ and any scalar α, there is a $\hat{h} \in \hat{\mathcal{F}}_T$ such that $\hat{h} = \alpha\hat{f}$. Define $f = \hat{f}|_{\mathcal{G}}$. Then $h = \hat{h}|_{\mathcal{G}}$ is such that

$$\begin{aligned} h_c(\mathbf{x}) &= \hat{h}_i\left(x_c; \mathbf{m}_c, \mathbf{x}\right) \\ &= \alpha\hat{f}_i\left(x_c; \mathbf{m}_c, \mathbf{x}\right) \\ &= \alpha f_c(\mathbf{x}) \end{aligned}$$

for all $c \in \mathcal{C}$ with $i = \mathcal{T}(c)$. That is, $(\alpha\hat{f})|_{\mathcal{G}} = \alpha \cdot \hat{f}|_{\mathcal{G}}$.
Therefore, evaluation on a network $(|_{\mathcal{G}})$ is a linear operator on $\hat{\mathcal{F}}_T$. ■

Corollary 4.4. *Assume $\hat{\mathcal{F}}_T$ is a vector space. Then, $\mathcal{F}_{\mathcal{G}} = \hat{\mathcal{F}}_T|_{\mathcal{G}}$ is also a vector space.* □

Corollary 4.5. *Assume $\hat{\mathcal{F}}_T$ is a vector space. Then, evaluating at a network $\mathcal{G}$ (operator $|_{\mathcal{G}}$) partitions the space of functions $\hat{\mathcal{F}}_T$ into affine planes parallel to the kernel (or nullspace)* $\ker(|_{\mathcal{G}})$ *such that each plane represents the set of oracle functions that behave the same in that network, that is,*

$$\hat{f}|_{\mathcal{G}} = \hat{g}|_{\mathcal{G}} \iff \hat{f} - \hat{g} \in \ker(|_{\mathcal{G}})$$

for every $\hat{f}, \hat{g} \in \hat{\mathcal{F}}_T$. □

We now present synchrony properties for output vector spaces.

Lemma 4.6. *Assume $\mathcal{F}_{\mathcal{G}}$ is a vector space. Then, for any partition $\mathcal{A} \leq \mathcal{T}_{\mathcal{G}}$, the subset of $\mathcal{F}_{\mathcal{G}}$ that is $\mathcal{A}$-invariant is also a vector space.* □

Proof. Consider functions $f, g \in \mathcal{F}_{\mathcal{G}}$ that are $\mathcal{A}$-invariant. Defining, $h = f + g$, if $\mathbf{x} = P\overline{\mathbf{x}}$, where P is a partition matrix that represents $\mathcal{A}$, we have

$$\begin{aligned} h(P\overline{\mathbf{x}}) &= f(P\overline{\mathbf{x}}) + g(P\overline{\mathbf{x}}) \\ &= P\overline{f}(\overline{\mathbf{x}}) + P\overline{g}(\overline{\mathbf{x}}) \\ &= P\left(\overline{f}(\overline{\mathbf{x}}) + \overline{g}(\overline{\mathbf{x}})\right) \\ &= P\left(\overline{h}(\overline{\mathbf{x}})\right) \end{aligned}$$

so h is also $\mathcal{A}$-invariant. Moreover,

$$\alpha f(P\overline{\mathbf{x}}) = P\left(\alpha\overline{f}(\overline{\mathbf{x}})\right)$$

for any scalar α. ■

Corollary 4.7. *Assume $\mathcal{F}_\mathcal{G} = \hat{\mathcal{F}}_T|_\mathcal{G}$ is a vector space. Then, for any partition $\mathcal{A} \leq \mathcal{T}_\mathcal{G}$, the subset of $\hat{\mathcal{F}}_T$ that is $\mathcal{A}$-invariant is also a vector space.* □

In a practical application, the nominal admissible function f^* that we desire in theory might not be the one that is actually realized. This motivates the interest in having some sort of local robustness so functions f that are sufficiently close to $\hat{f}$ show similar properties.

Corollary 4.8. *Assume $\mathcal{F}_\mathcal{G}$ is a **normed** vector space. Given a $f^* \in \mathcal{F}_\mathcal{G}$, if we require that for some $\varepsilon > 0$, all the functions $f \in \mathcal{F}_\mathcal{G}$ in the ball $\|f - f^*\| < \varepsilon$ are $\mathcal{A}$-invariant, then the whole $\mathcal{F}_\mathcal{G}$ has to be $\mathcal{A}$-invariant.* □

This means that asking for local robustness in terms of $\mathcal{A}$-invariance is the same as requiring the whole $\mathcal{F}_\mathcal{G}$ to be $\mathcal{A}$-invariant, that is, global $\mathcal{A}$-invariance.
From Theorem 3.5, we know this can only be achieved when $\mathcal{A}$ is a balanced partition in $\mathcal{G}$. We now present special cases for particular monoids in which we only require certain subsets of $\mathcal{F}_\mathcal{G}$ to be $\mathcal{A}$-invariant in order to enforce $\mathcal{A}$ to be balanced.
The usual proof for Theorem 3.5 in the unweighted and scalar-weighted cases, uses functions that are linear in the weights. This approach, however, does not scale well to general weight sets. Note that the analogous in this framework is to consider functions that are **additive in the weights**, that is,

$$p(w_1 \| w_2) = p(w_1) + p(w_2)$$

If there is an annihilator in $\mathcal{M}$, then $p(w) = 0_\mathcal{M}$ for all $w \in \mathcal{M}$. That is, only the trivial case for such functions exists. We now present an extension of the linear in the weights argument to a particular type of weight monoids for which it works.

Remark 4.9. We always assume non-trivial output vector spaces. That is, $\mathbb{Y}_i \neq \{0\}$. Otherwise, synchrony would always be trivially guaranteed but it would be a non-interesting particular case. □

Remark 4.10. If the state sets $\mathbb{X}_i$ only have one element, then it is irrelevant to talk about synchronicity in the first place. For this reason, we are assuming that the state sets are non-singleton and we can choose $x_a \neq x_b$ with $x_a, x_b \in \mathbb{X}_i$. □

Theorem 4.11. *Consider non-trivial output vector spaces $\{\mathbb{Y}_i\}_{i\in T}$ and assume that the edges are in the monoid $\mathcal{M} = \langle \mathbb{M} | E \rangle$, with $\mathbb{M} = \mathbb{R} \times \mathbb{W}$ and*

$$E = \{\lambda_1 w \| \lambda_2 w = (\lambda_1 + \lambda_2) w, \quad \forall \lambda_1, \lambda_2 \in \mathbb{R}, w \in \mathbb{W}\}$$

where $\mathbb{W}$ is not necessarily countable.
Consider the set of oracle components $\hat{f}_i$, $i \in T$, that are only dependent on neighbors that are in a specific state $\overline{x}_k$, of the form

$$\hat{f}_i \left(x; \sum \lambda_w w, \overline{x}_k \right) = \lambda_e \mathbf{v}, \quad \mathbf{v} \in \mathbb{Y}_i, \mathbf{v} \neq 0_{\mathbb{Y}_i}$$

for some $e \in \mathbb{W}$.
If the subset of oracle functions in $\mathcal{F}_\mathcal{G}$ that are built with $\hat{f}_i$'s of the type above is $\bowtie$-invariant, then $\bowtie$ is balanced in $\mathcal{G}$. □

Proof. Assume $\bowtie$ is not balanced. Then, there are $\bowtie(c) = \bowtie(d)$ such that $\mathbf{m}_c P \neq \mathbf{m}_d P$. Consider k one of the colors in which they differ. That is, $[\mathbf{m}_c P]_k \neq [\mathbf{m}_d P]_k$.
Moreover, choose some state $\mathbf{x} \in \Delta_{\bowtie}^{\mathbb{X}}$, that is, $\mathbf{x} = P\overline{\mathbf{x}}$, such that $\overline{x}_k$ is different from all other entries of $\overline{\mathbf{x}}$.
An element of the monoid $\mathcal{M}$ can be written as linear combination over a finite subset of elements in $\mathbb{W}$, that is, $\sum \lambda_w w$. If $[\mathbf{m}_c P]_k \neq [\mathbf{m}_d P]_k$, then they differ on the associated coefficient of at least one element $e \in \mathbb{W}$. Then, there is an $\hat{f}_i$ as defined above, sensitive to that element e, so that $\hat{f}_i\left(x; [\mathbf{m}_c P]_k, \overline{x}_k\right) = \lambda_e^c \mathbf{v} \neq \lambda_e^d \mathbf{v} = \hat{f}_i\left(x; [\mathbf{m}_d P]_k, \overline{x}_k\right)$.
Then, we have an $f \in \mathcal{F}_{\mathcal{G}}$ and $\overline{\mathbf{x}}$ such that $f\left(P\overline{\mathbf{x}}\right) \notin \Delta_{\bowtie}^{\mathbb{Y}}$. Therefore, the only way to not be able to find such an f, is for $\bowtie$ to be balanced. ∎

The next result is valid for systems in which the weight set allows for the existence of an annihilator. However, the monoid is almost free, in the sense that its congruence relation does not define further equivalence classes.

Theorem 4.12. *Consider non-trivial output vector spaces $\{\mathbb{Y}_i\}_{i\in T}$ and assume that the edges are either on a free monoid $\mathcal{M} = \langle \mathbb{W}|\rangle$ or the result of adding an annihilator to a free monoid. That is, $\mathcal{M} = \langle \{a\} \cup \mathbb{W}|E\rangle$, with*

$$E = \{w \| a = a, \quad \forall w \in \mathcal{M}\}$$

where $\mathbb{W}$ is not necessarily countable.
Consider the set of oracle components $\hat{f}_i$, $i \in T$, that are only dependent on neighbors that are in a specific state $\overline{x}_k$, of the form

$$\hat{f}_i\left(x; \sum w, \overline{x}_k\right) = \mathbf{v} \prod p(w), \quad \mathbf{v} \in \mathbb{Y}_i, \mathbf{v} \neq 0_{\mathbb{Y}_i}$$

If the subset of oracle functions in $\mathcal{F}_{\mathcal{G}}$ that are built with $\hat{f}_i$'s of the type above is $\bowtie$-invariant, then $\bowtie$ is balanced in $\mathcal{G}$. □

Proof. Assume $\bowtie$ is not balanced. Then, there are $\bowtie(c) = \bowtie(d)$ such that $\mathbf{m}_c P \neq \mathbf{m}_d P$. Consider k one of the colors in which they differ. That is, $[\mathbf{m}_c P]_k \neq [\mathbf{m}_d P]_k$.
Moreover, choose some state $\mathbf{x} \in \Delta_{\bowtie}^{\mathbb{X}}$, that is, $\mathbf{x} = P\overline{\mathbf{x}}$, such that $\overline{x}_k$ is different from all other entries of $\overline{\mathbf{x}}$.
An element of the monoid $\mathcal{M}$ can be written as a finite sum over elements in $\mathbb{W}$. Call the support, that is, the elements that appear at least once in $[\mathbf{m}_c P]_k$ as $w_1, \ldots, w_n$ and the support of $[\mathbf{m}_d P]_k$ as $v_1, \ldots, v_m$. We can, with some function p, assign to each distinct element of the union of both sets, a distinct prime number, with the exception of the zero element $0_{\mathbb{M}}$ and a possible annihilator a, in which we have instead that $p(0_{\mathcal{M}}) = 1$ and $p(a) = 0$.
If $[\mathbf{m}_c P]_k \neq [\mathbf{m}_d P]_k$, then, there is an $\hat{f}_i$ as defined above, so that $\hat{f}_i\left(x; [\mathbf{m}_c P]_k, \overline{x}_k\right) \neq \hat{f}_i\left(x; [\mathbf{m}_d P]_k, \overline{x}_k\right)$.
Then, we have an $f \in \mathcal{F}_{\mathcal{G}}$ and $\overline{\mathbf{x}}$ such that $f\left(P\overline{\mathbf{x}}\right) \notin \Delta_{\bowtie}^{\mathbb{Y}}$. Therefore, the only way to not be able to find such an f, is for $\bowtie$ to be balanced. ∎

Remark 4.13. Note that additional conditions on E, that defines congruence relation of the monoid, might invalidate this approach, e.g., $w_1 \| w_2 = w_3 \| w_4$, where all weights are different. □

5. Quotients. In this section we describe how the behavior of a network $\mathcal{G}$ when evaluated at some polydiagonal $\Delta_{\bowtie}$ for some balanced partition $\bowtie$ can be described by a smaller network $\mathcal{Q}$.

Definition 5.1. *Consider a network $\mathcal{G}$ defined on a cell set $\mathcal{C}_\mathcal{G}$ with a cell type partition $\mathcal{T}_\mathcal{G}$ and an in-adjacency matrix M. Take a partition $\bowtie$ balanced on $\mathcal{G}$.*
*The **quotient network** $\mathcal{Q}$ of $\mathcal{G}$ over $\bowtie$, denoted $\mathcal{Q} = \mathcal{G}/\bowtie$ is defined on a cell set $\mathcal{C}_\mathcal{Q} = \mathcal{C}_\mathcal{G}/\bowtie$ with a cell type partition $\mathcal{T}_\mathcal{Q} = \mathcal{T}_\mathcal{G}/\bowtie$ and an in-adjacency matrix Q given by $MP = PQ$, where P represents $\bowtie$.* □

Definition 5.2. *Consider networks $\mathcal{G}$ and $\mathcal{Q}$ such that $\mathcal{Q} = \mathcal{G}/\bowtie$ and some $\mathcal{G}$-admissible function $f = \hat{f}|_\mathcal{G}$, with $\hat{f} \in \hat{\mathcal{F}}_T$.*
*The **quotient function** g of f over the balanced partition $\bowtie$, denoted $g = f/\bowtie$ is given by $g = \hat{f}|_\mathcal{Q}$.* □

Lemma 5.3. *The quotient function $g = f/\bowtie$ is well-defined since is does not depend on the particular choice of oracle function. That is, for any $\hat{f}^1, \hat{f}^2 \in \hat{\mathcal{F}}_T$*

$$\hat{f}^1|_\mathcal{G} = \hat{f}^2|_\mathcal{G} \implies \hat{f}^1|_\mathcal{Q} = \hat{f}^2|_\mathcal{Q}$$

that is, a particular $f \in \mathcal{F}_\mathcal{G}$ implies a unique $g \in \mathcal{F}_\mathcal{Q}$. □

Proof. By assumption of $\hat{f}^1|_\mathcal{G} = \hat{f}^2|_\mathcal{G}$ we have,

$$\hat{f}^1_i\left(\overline{x}_k; \mathbf{m}_c, P\overline{\mathbf{x}}\right) = \hat{f}^2_i\left(\overline{x}_k; \mathbf{m}_c, P\overline{\mathbf{x}}\right)$$

for all $c \in \mathcal{C}_\mathcal{G}$ with $k = \bowtie(c)$.
Since $\mathbf{q}_k = \mathbf{m}_c P$, where the weight vector $\mathbf{q}_k$ is the k^{th} row of Q, an in-adjacency matrix of $\mathcal{Q}$. This implies

$$\hat{f}^1_i(\overline{x}_k; \mathbf{q}_k, \overline{\mathbf{x}}) = \hat{f}^2_i(\overline{x}_k; \mathbf{q}_k, \overline{\mathbf{x}})$$

for any $k \in \mathcal{C}_\mathcal{Q}$, with $i = \mathcal{T}_\mathcal{Q}(k)$. That is $\hat{f}^1|_\mathcal{Q} = \hat{f}^2|_\mathcal{Q}$. ■

In section 3 it was shown that any admissible $f \in \mathcal{F}_\mathcal{G}$ when evaluated on $\Delta_{\bowtie}$ can be determined by a simpler function, related to it by (3.4). The following results show that this is what connects the quotient network and quotient function.

Theorem 5.4. *Consider a network $\mathcal{G}$ and a partition $\bowtie$ balanced on it. Let $f \in \mathcal{F}_\mathcal{G}$. The function obtained by constraining f to the polydiagonal $\Delta_{\bowtie}$, is the quotient function $g = f/\bowtie$, that is,*

$$f(P\overline{\mathbf{x}}) = Pg(\overline{\mathbf{x}})$$

□

Proof. Consider some oracle function $\hat{f} \in \hat{\mathcal{F}}_T$ such that $f = \hat{f}|_\mathcal{G}$ and $g = \hat{f}|_\mathcal{Q}$. Then, when $\mathbf{x} \in \Delta_{\bowtie}$, that is $\mathbf{x} = P\overline{\mathbf{x}}$, we have that

$$\begin{aligned} f_c(P\overline{\mathbf{x}}) &= \hat{f}_i\left(\overline{x}_k; \mathbf{m}_c, P\overline{\mathbf{x}}\right) \\ &= \hat{f}_i(\overline{x}_k; \mathbf{q}_k, \overline{\mathbf{x}}) \\ &= g_k(\overline{\mathbf{x}}) \end{aligned}$$

since $\mathbf{q}_k = \mathbf{m}_c P$, for all $c \in \mathcal{C}_\mathcal{G}$ with $i = \mathcal{T}_\mathcal{G}(c)$ and $k = \bowtie(c)$. Therefore, the function $g = (g_k)_{k \in \mathcal{C}_\mathcal{Q}}$ is related to f by equation (3.4). ■

Example 5.5. Consider the given partition $\mathcal{A} = \{\{1, 2\}, \{3\}\}$ on the **CCN** of Example 2.14 (Figure 2.3). One partition matrix of $\mathcal{A}$ is

$$P = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.1}$$

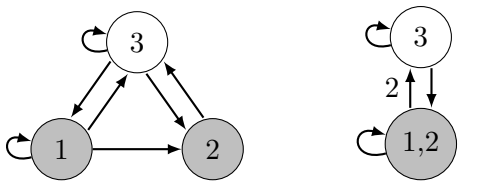
in which each column identifies one of the colors of the partition. From this we obtain the product

$$MP = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix} \tag{5.2}$$

Note that rows 1 and 2 are the same. That means that for any admissible f we have $f_1(\mathbf{x}) = f_2(\mathbf{x})$ when $x_1 = x_2$.
Observe that this is in agreement with the functional form we wrote in (2.8) and (2.9).
Since the rows of MP respect an equality relationship according to $\mathcal{A}$, then $\mathcal{A}$ is balanced and there is a quotient matrix Q that obeys the balanced condition (3.9). In fact, the quotient matrix Q is

$$Q = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix} \tag{5.3}$$

which is directly obtained from MP by compressing its rows according to $\mathcal{A}$.
The behavior of this **CCN** when $x_1 = x_2$ is then described by the smaller **CCN** given by the quotient matrix Q which is represented in Figure 5.1b. The coloring is a way of representing



(a) Original (b) Quotient

Figure 5.1: Color-coded network of Figure 2.3 and its quotient over the balanced partition $\{\{1, 2\}, \{3\}\}$

the partition $\mathcal{A} = \{\{1, 2\}, \{3\}\}$ over which the quotient is done. Note that in both Figures 5.1a and 5.1b each gray cell receives one connection from a gray cell and one connection from a white cell. On the other hand, each white cell receives a connection from a white cell and two

connections from a gray cell. The function $g = f/\bowtie$ has the following structure

$$g_{12}(\mathbf{x}) = \hat{f}(x_{12}; \begin{bmatrix}1 & 1\end{bmatrix}, \mathbf{x}) \tag{5.4}$$

$$g_3(\mathbf{x}) = \hat{f}(x_3; \begin{bmatrix}2 & 1\end{bmatrix}, \mathbf{x}) \tag{5.5}$$

where $\hat{f} \in \hat{\mathcal{F}}_T$ is any oracle function such that $f = \hat{f}|_{\mathcal{G}}$. □

Remark 5.6. Note that finding a balanced partition from its graph representation or its matrix M is not obvious. See Example 5.7. □

Example 5.7. Consider the following network illustrated in Figure 5.2. Since cells 2 and

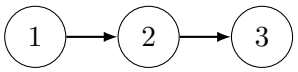


Figure 5.2: Chain **CCN**

3 have the same type of input it might be tempting to think that $\mathcal{A} = \{\{1\}, \{2, 3\}\}$ should be balanced. Note, however, that the rows of the corresponding matrix MP (5.6) do not respect the row equalities according to $\mathcal{A}$, which means that it is not balanced.

$$MP = \begin{bmatrix}0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix} \tag{5.6}$$

Another way to see this is to color the cells according to the partition (Figure 5.3) and see that cells with the same color do not have equivalent colored input sets. Note that cells 2 and

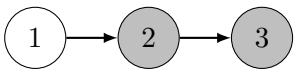


Figure 5.3: Unbalanced coloring

3 are both gray but one of them receives one edge from a white cell and the other receives one edge from a gray cell. Therefore, this coloring (partition) is not balanced. In fact, it can be easily seen that the only balanced partition of this network is the trivial one. □

The following result shows that the quotient operation is transitive.

Lemma 5.8. *Consider networks $\mathcal{G}_0$, $\mathcal{G}_1$, $\mathcal{G}_2$ such that there are balanced partitions $\bowtie_{01}$, $\bowtie_{12}$ such that $\mathcal{G}_1 = \mathcal{G}_0/\bowtie_{01}$ and $\mathcal{G}_2 = \mathcal{G}_1/\bowtie_{12}$. Then, there is a partition $\bowtie_{02}$ with $\bowtie_{01} \leq \bowtie_{02}$ such that $\mathcal{G}_2 = \mathcal{G}_0/\bowtie_{02}$. Furthermore, $P_{02} = P_{01}P_{12}$ where the matrices represent the corresponding indexed partitions.* □

Proof. Name the cell type partitions of the networks $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{T}_2$ accordingly. Then,

$$\mathcal{T}_0 = P_{01}\mathcal{T}_1, \quad \mathcal{T}_1 = P_{12}\mathcal{T}_2$$

implies

$$\mathcal{T}_0 = (P_{01}P_{12})\mathcal{T}_2$$

Let M_0, M_1, M_2 represent the in-adjacency matrices of the networks. From being balanced we know that

$$M_0P_{01} = P_{01}M_1, \quad M_1P_{12} = P_{12}M_2$$

by multiplying the first equality by P_{12} on the right

$$\begin{aligned} M_0(P_{01}P_{12}) &= P_{01}(M_1P_{12}) \\ &= (P_{01}P_{12})M_2 \end{aligned}$$

which means that

$$\mathcal{G}_2 = (\mathcal{G}_0/\bowtie_{01})/\bowtie_{12} = \mathcal{G}_0/\bowtie_{02}$$

and that $P_{02} = P_{01}P_{12}$. ■

The next results shows how two different quotients of the same network can be related by one of them being a quotient of the other

Lemma 5.9. *Consider networks $\mathcal{G}_0$, $\mathcal{G}_1$, $\mathcal{G}_2$ such that there are balanced partitions $\bowtie_{01}$, $\bowtie_{02}$ such that $\mathcal{G}_1 = \mathcal{G}_0/\bowtie_{01}$ and $\mathcal{G}_2 = \mathcal{G}_0/\bowtie_{02}$. If $\bowtie_{01} \leq \bowtie_{02}$ there is a balanced partition $\bowtie_{12}$ such that $\mathcal{G}_2 = \mathcal{G}_1/\bowtie_{12}$. Furthermore, $P_{02} = P_{01}P_{12}$ where the matrices P represent the correspondingly indexed partitions.* □

Proof. Name the cell type partitions of the networks $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{T}_2$ accordingly. Then,

$$\mathcal{T}_0 = P_{01}\mathcal{T}_1, \quad \mathcal{T}_0 = P_{02}\mathcal{T}_2$$

from $\bowtie_{01} \leq \bowtie_{02}$ we know that there is a partition matrix P_{12} such that

$$P_{02} = P_{01}P_{12}$$

replacing P_{02} in the second equality

$$\mathcal{T}_0 = P_{01}P_{12}\mathcal{T}_2$$

using the first equality

$$P_{01}\mathcal{T}_1 = P_{01}(P_{12}\mathcal{T}_2)$$

Since P_{01} has full column rank, it can be canceled on the left

$$\mathcal{T}_1 = P_{12}\mathcal{T}_2$$

Let M_0, M_1, M_2 represent the in-adjacency matrix of the networks. From being balanced we know that

$$M_0P_{01} = P_{01}M_1, \quad M_0P_{02} = P_{02}M_2$$

replacing P_{02} in the second equality

$$(M_0P_{01})P_{12} = P_{01}P_{12}M_2$$

which by the first balanced equality

$$(P_{01}M_1)P_{12} = P_{01}P_{12}M_2$$

now, canceling P_{01} on the left is allowed

$$M_1P_{12} = P_{12}M_2$$

which means that $\bowtie_{12}$, represented by P_{12} is balanced on $\mathcal{G}_1$ and

$$\mathcal{G}_2 = \mathcal{G}_1/\bowtie_{12}$$

■

6. Lattice of balanced partitions. This sections presents the properties of $\Lambda_{\mathcal{G}}$, which denotes the set of all balanced partitions of a given network $\mathcal{G}$.
In [13] it is shown that $\Lambda_{\mathcal{G}}$ forms a lattice under the sub-partition operation $\leq$ as described in (2.1). That is, it forms a partially ordered set such that for any $\bowtie_1, \bowtie_2 \in \Lambda_{\mathcal{G}}$, there also exist in $\Lambda_{\mathcal{G}}$ partitions $\bowtie_1 \vee \bowtie_2$ and $\bowtie_1 \wedge \bowtie_2$ which are the **least upper bound** or **join** and the **greatest lower bound** or **meet**. This means that there is a **maximal** ($\top$) and a **minimal** ($\bot$) balanced partitions, the last of which we already know to be the trivial partition.
The coarsest invariant refinement (**CIR**) algorithm, is a polynomial-time algorithm that was first developed in [1] with the goal of finding the maximal balanced partition. Recently, in [9] it was noted that this algorithm does more that just finding the maximal balanced partition. In fact, given any input partition, it outputs the greatest balanced partition that is finer ($\leq$) than the input one. Therefore, the maximal partition is given by $\top = cir(\mathcal{T})$. We now show that the meet ($\wedge$) and join ($\vee$) are also defined.

Lemma 6.1. *For every pair $\bowtie_1, \bowtie_2 \in \Lambda_{\mathcal{G}}$ there is a least upper bound or join $\bowtie_3 \in \Lambda_{\mathcal{G}}$ denoted by $\bowtie_3 = \bowtie_1 \vee \bowtie_2$.* □

Proof. Any partition $\bowtie$ that is simultaneous coarser than $\bowtie_1$ and $\bowtie_2$ has to obey

$$\begin{cases} \bowtie_1(c) & = \bowtie_1(d) \\ & \text{or} \\ \bowtie_2(c) & = \bowtie_2(d) \end{cases} \implies \bowtie(c) = \bowtie(d)$$

For such partition, any chain of cells $c = c_1, \ldots, c_k = d$ such that either $\bowtie_1(c_i) = \bowtie_1(c_{i+1})$ or $\bowtie_2(c_i) = \bowtie_2(c_{i+1})$ implies that $\bowtie(c) = \bowtie(d)$. The finest such partition $\bowtie_3$ is the one such that $\bowtie_3(c) = \bowtie_3(d)$ if and only if there is such a chain.
We now show that $\bowtie_3$ is balanced. Choose any two colors $A, B \in \bowtie_3$. Since $\bowtie_1, \bowtie_2$ are both sub-partitions of $\bowtie_3$ there are colors $b^1_1, \ldots, b^1_{k_1} \in \bowtie_1$ and $b^2_1, \ldots, b^2_{k_2} \in \bowtie_2$ such that $B = \bigcup_{i=1}^{k_1} b^1_i = \bigcup_{i=1}^{k_2} b^2_i$. For any $c, d \in A$ such that $\bowtie_i(c) = \bowtie_i(d)$ for some $i \in \{1, 2\}$ we have that

$$\sum_{e \in b^i_j} w_{ce} = \sum_{e \in b^i_j} w_{de} \quad \forall j \in \{1, \ldots, k_i\}$$

which implies

$$\sum_{e\in B} w_{ce} = \sum_{e\in B} w_{de}$$

since for each link of the chain this value is preserved, it is also preserved across the whole chain and therefore the whole color A. This applies to every pair of colors $A, B \in \bowtie$ which means that $\bowtie$ is balanced. ■

Lemma 6.2. *For every pair $\bowtie_1, \bowtie_2 \in \Lambda_{\mathcal{G}}$ there is a greatest lower bound or meet $\bowtie_3 \in \Lambda_{\mathcal{G}}$ denoted by $\bowtie_3 = \bowtie_1 \wedge \bowtie_2$.* □

Proof. Any partition $\bowtie$ that is simultaneous finer than $\bowtie_1$ and $\bowtie_2$ has to obey

$$\bowtie(c) = \bowtie(d) \implies \begin{cases} \bowtie_1(c) = \bowtie_1(d) \\ \bowtie_2(c) = \bowtie_2(d) \end{cases}$$

call $\bowtie_{12}$ the coarsest such partition, created by making the implication into an equivalence. Since such a partition is unique, $\bowtie_3$ is given by $\bowtie_3 = cir(\bowtie_{12})$. Note that there was no reason to believe that $\bowtie_{12}$ was itself balanced and can be verified empirically not to be. ■

The following result relates the lattice of a network with the lattice of one of its quotients.

Lemma 6.3. *Consider networks $\mathcal{G}$ and $\mathcal{Q}$ related by $\mathcal{Q} = \mathcal{G}/\bowtie$. Then, there is a one-to-one correspondence between the elements of $\Lambda_{\mathcal{Q}}$ and the elements of $\Lambda_{\mathcal{G}}$ that are coarser than $\bowtie$. This relation is given as $P_{\mathcal{G}} = P_{\bowtie} P_{\mathcal{Q}}$ where $P_{\mathcal{G}}$, $P_{\mathcal{Q}}$ represent partitions in $\Lambda_{\mathcal{G}}$, $\Lambda_{\mathcal{Q}}$ respectively. Therefore, we say that $\Lambda_{\mathcal{Q}} = \Lambda_{\mathcal{G}}/\bowtie$.* □

Proof. Lemma 5.8 shows that for any partition $\bowtie_{12} \in \Lambda_{\mathcal{Q}}$ there is a partition $\bowtie_{02} \in \Lambda_{\mathcal{G}}$, coarser than $\bowtie$, such that $P_{02} = P_{\bowtie} P_{12}$. Conversely, Lemma 5.9 shows that for any $\bowtie_{02} \in \Lambda_{\mathcal{G}}$ that is coarser than $\bowtie$ there is a partition $\bowtie_{12} \in \Lambda_{\mathcal{Q}}$ such that they are related to each other in the same way as before. ■

The problem of trying to get an exhaustive list of the elements of a lattice Λ can be potentially intractable. Note that given a partition, it is easy and efficient to verify whenever it is balanced. However, even for relatively small networks, the set of all possible partitions is simply too large to do an exhaustive search on it. The number of partitions on a set is given by the Bell numbers B_n (also called exponential numbers), referenced in the online database OEIS (The On-Line Encyclopedia of Integer Sequences by the code A000110 [11]. A method to reduce the search space is described in [9]. The algorithm, called '**SPLIT** and **CIR**', uses the idea that instead of testing all $(\prod_{i=1}^{r}(B_{k_i} - 1))$ sub-partitions of $\top$, one can apply the **CIR** method to its $\left(\sum_{i=1}^{r} 2^{k_i - 1}\right)$ immediate descendants and then repeat, finding all the balanced partitions layer by layer.

The specific gains of this approach are difficult to analyze and can be highly dependent of the particular network of study (e.g., the number of layers in the lattice $\Lambda_{\mathcal{G}}$). A worst-case scenario (all B_n partitions balanced) evaluation could be too pessimistic and a bad metric to decide if it would be an approach of interest for application in a real-world network.

This worst-case is scenario is a shortcoming that is common to all approaches that try to find all the balanced partitions in an explicit exhaustive way. Consider for example an all-to-all

uniform-connection network of size n, with a single cell type. In this case, even for a relatively small n, the lattice would be too large to enumerate its (B_n) elements or draw any schematic, yet it can be described in one simple sentence (i.e., "every partition is balanced"). Ironically, the simplest network, whose lattice is the easiest to determine corresponds exactly to the worst-case scenario of such methods.

7. CIR algorithm improvement. In this section we describe our improvement of the **CIR** algorithm that works with general weight sets and has a worst-case complexity of $\mathbf{O}(|\mathcal{C}|^3)$ in the case of a dense graph and $\mathbf{O}(|\mathcal{C}|^2)$ in the sparse case.
Consider a network represented by a matrix M together with an initial partition $\mathcal{A}_0 \leq \mathcal{T}$ represented by matrix P_0, of which we want to find the coarsest refinement (e.g., make P_0 the characteristic matrix of $\mathcal{T}$ if the goal is to find the maximal balanced partition $\top$).

7.1. Method. The idea of this algorithm is to start with the initial partition $\mathcal{A}_0$ and progressively refine it in a conservative manner. That is, given a partition $\mathcal{A}_i$, we construct a partition $\mathcal{A}_{i+1} \leq \mathcal{A}_i$ such that any balanced partition finer than $\mathcal{A}_i$ is also finer than $\mathcal{A}_{i+1}$. We create $\mathcal{A}_{i+1}$ by taking each color of $\mathcal{A}_i$ and splitting its cells according to whenever their corresponding rows in MP_i match or not. If $\mathcal{A}_{i+1} = \mathcal{A}_i$ the algorithm has converged and we found $\mathcal{A}_i = cir(\mathcal{A}_0)$, otherwise we continue iterating.

Lemma 7.1. *According to the described iterative method, any balanced partition finer than $\mathcal{A}_i$ is also finer than $\mathcal{A}_{i+1}$.* □

Proof. Assume that there are cells c, d such that $\mathcal{A}_i(c) = \mathcal{A}_i(d)$ but rows c and d of MP_i do not match perfectly (assume on k^{th} column). Note that the k^{th} color of $\mathcal{A}_i$ will correspond to either a color, or a union of colors of any balanced partition finer than $\mathcal{A}_i$. This means that no matter what refinement happens, the cells c and d will have no chance of having the same color in a balanced refinement, since if the sum of the parts is different, it will not be possible for the parts themselves to match. Therefore, any balanced partition finer than $\mathcal{A}_i$ is also finer than $\mathcal{A}_{i+1}$. ■

Remark 7.2. Note that if at a certain iteration no more refinement happens, that means that the balanced condition (3.8) has been achieved and we found $cir(\mathcal{A}_0)$. □

Lemma 7.3. *The iterative procedure always converges in at most $|\mathcal{C}| - rank(\mathcal{A}_0)$ iterations.* □

Proof. Note that in each iteration, either the rank of the partition increases or the algorithm stops because a balanced partition was achieved. In the worst case scenario, the rank increases by one until the trivial partition is reached. Therefore, the algorithm always converges in at most $|\mathcal{C}| - rank(\mathcal{A}_0)$ iterations. ■

Since this algorithm always converges, this shows by construction that $cir(\mathcal{A}_0)$ exists. That is, for any partition $\mathcal{A}_0$, there is a unique balanced partition $\mathcal{A}_i = cir(\mathcal{A}_0)$ such $\mathcal{A}_i \leq \mathcal{A}_0$ and $\bowtie \leq \mathcal{A}_i$ for any balanced partition $\bowtie$ such that $\bowtie \leq \mathcal{A}_0$.

7.2. Efficient implementation and cost analysis. Note that a partition matrix P on a set of cells $\mathcal{C}$ can be efficiently represented by a vector of size $|\mathcal{C}|$ as seen in Example 2.9. Calculating the product MP_i consists on summing ($\|$) certain elements of M according to the

pattern described in P_i. To compare rows of MP_i previous works considered a quadratic cost which was the bottleneck of the algorithm. If the appropriate data structure (hash table) is used, such operation is of the order $\mathbf{O}(rank(\mathcal{A}_i))$. A pseudo-code description of the algorithm implementation is presented in Algorithms 7.1 and 7.2.

Algorithm 7.1 CIR algorithm

$M \leftarrow$ CCN matrix
$p_0 \leftarrow$ initial partition vector
$r_0 \leftarrow$ rank of p_0
$p_{new} \leftarrow p_0$
$r_{new} \leftarrow r_0$
repeat
  $p_{old} \leftarrow p_{new}$
  $r_{old} \leftarrow r_{new}$
  $(p_{new}, r_{new}) \leftarrow cir\_iteration(M, p_{old}, r_{old})$
until $r_{new} == r_{old}$

Algorithm 7.2 CIR iteration

$M \leftarrow$ CCN matrix
$p_{old} \leftarrow$ previous partition vector
$r_{old} \leftarrow$ rank of p_{old}
$p_{new} \leftarrow$ new partition vector
$r_{new} \leftarrow 0$
for $r = 1 : |\mathcal{C}|$ **do**
  $v \leftarrow$ zero vector of size r_{old}
  for $c : (r, c) \in \mathcal{E}$ **do**
    $v[p_{old}(c)] \leftarrow v[p_{old}(c)] + M(r, c)$
  end for
  $s \leftarrow vec2string([p_{old}(r), v])$
  $value \leftarrow hash\_table.find(s)$
  if value NOT_FOUND **then**
    $r_{new} \leftarrow r_{new} + 1$
    $p_{new}[r] \leftarrow r_{new}$
    $hash\_table.insert(s, r_{new})$
  else
    $p_{new}[r] \leftarrow value$
  end if
end for

Lemma 7.4. *This implementation of the **CIR** algorithm leads to a worst-case complexity of* $\mathbf{O}(|\mathcal{C}|^3))$. □

Proof. In each iteration we are summing ($\|$) a total of $|\mathcal{E}|$ entries of M. The lookup and insertion in an hash table are fast operations with complexity $\mathbf{O}(1)$ which are each executed

$|\mathcal{C}|$ times. The $|\mathcal{C}|$ strings that are used as key in the hash table have size proportional to $rank(\mathcal{A}_i)$.
The complexity of the i^{th} iteration is then $\mathbf{O}(|\mathcal{E}| + |\mathcal{C}| + |\mathcal{C}|rank(\mathcal{A}_i))$. In the worst-case scenario the rank increases by one and the number of iterations is $\mathbf{O}(|\mathcal{C}|)$. This implies total worst-case complexity of $\mathbf{O}(|\mathcal{C}|^3)$. ■

Remark 7.5. In practice, the number of iterations seems to be much lower than $|\mathcal{C}|$ which means that this is a very pessimistic upper bound for the complexity. □

We illustrate this algorithm with the following example.

Example 7.6. Consider the network illustrated in Figure 7.1 with cell type partition $\mathcal{T} = \{\{1, 2, 5, 6\}, \{3, 4\}\}$. The edge weight monoid is the same as in the parallel of resistors (Example 2.3). We assume that the arrows all represent values of 30. Note that the zero of the monoid is $0_{\mathcal{M}} = \infty$. This is represented by the matrix in (7.1).

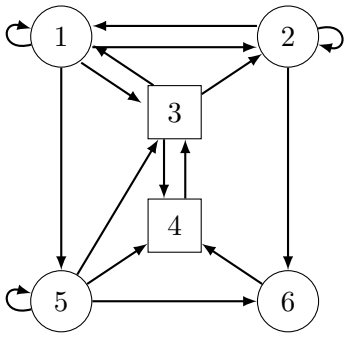


Figure 7.1: Network of Example 7.6 illustrating the CIR algorithm

$$M = \begin{bmatrix} 30 & 30 & 30 & \infty & \infty & \infty \\ 30 & 30 & 30 & \infty & \infty & \infty \\ 30 & \infty & \infty & 30 & 30 & \infty \\ \infty & \infty & 30 & \infty & 30 & 30 \\ 30 & \infty & \infty & \infty & 30 & \infty \\ \infty & 30 & \infty & \infty & 30 & \infty \end{bmatrix} \tag{7.1}$$

If we are interested in finding the top partition $\top$, we initialize $\mathcal{A}_0 = \mathcal{T}$. This partition can be represented by the matrix P_0

$$\mathcal{A}_0 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix} \quad P_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Applying the algorithm we get

$$\left[\begin{array}{c|cc} \mathcal{A}_0 & MP_0 \end{array}\right] = \left[\begin{array}{c|cc} 1 & 15 & 30 \\ 1 & 15 & 30 \\ 2 & 15 & 30 \\ 2 & 15 & 30 \\ 1 & 15 & \infty \\ 1 & 15 & \infty \end{array}\right]$$

whose row comparison determines the next iteration $\mathcal{A}_1$ and P_1

$$\mathcal{A}_1 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{bmatrix} \qquad P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Applying the same procedure

$$\left[\begin{array}{c|ccc} \mathcal{A}_1 & MP_1 \end{array}\right] = \left[\begin{array}{c|ccc} 1 & 15 & 30 & \infty \\ 1 & 15 & 30 & \infty \\ 2 & 30 & 30 & 30 \\ 2 & \infty & 30 & 15 \\ 3 & 30 & \infty & 30 \\ 3 & 30 & \infty & 30 \end{array}\right]$$

and we get the second iteration defined by

$$\mathcal{A}_2 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 4 \\ 4 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[\begin{array}{c|ccc} \mathcal{A}_2 & MP_2 \end{array}\right] = \left[\begin{array}{c|ccc} 1 & 15 & 30 & \infty & \infty \\ 1 & 15 & 30 & \infty & \infty \\ 2 & 30 & \infty & 30 & 30 \\ 3 & \infty & 30 & \infty & 15 \\ 4 & 30 & \infty & \infty & 30 \\ 4 & 30 & \infty & \infty & 30 \end{array}\right]$$

We can now see that $\mathcal{A}_2 = \mathcal{A}_3$. This means that we have converged and $\mathcal{A}_2 = cir(\mathcal{A}_0) = cir(\mathcal{T}) = \top$.

This is not the only non-trivial balanced partition on this network. For example, with an initial

partition $\mathcal{B}_0 = \{\{1,2,5\},\{3,4\},\{6\}\}$ we find the other balanced partition $\mathcal{B}_1 = cir(\mathcal{B}_0) = \{\{1,2\},\{3\},\{4\},\{5\},\{6\}\}$.
Note that we already knew that any other balanced partitions would have to be finer that $\top$. Therefore we could have instead just verified if any of the partitions $\{\{1\},\{2\},\{3\},\{4\},\{5,6\}\}$ or $\{\{1,2\},\{3\},\{4\},\{5\},\{6\}\}$ were balanced. □

8. Conclusion. This paper generalizes the theory of coupled cell networks to multi edge and multi edge-type networks with arbitrarily complex edge weights. The formalism that was introduced here is simpler than the usual one based on groupoids of bijections of input sets. Moreover, we do not require the networks to obey such an artificial condition such as the 'consistency condition'. We extend previous results about balanced partitions and invariant synchrony patterns to this more general setting. An implementation of the CIR algorithm is presented which has a worst-case time complexity of $\mathbf{O}(|\mathcal{C}|^3)$ in opposition to the previous $\mathbf{O}((|\mathcal{E}|+|\mathcal{C}|)^4)$ cost.

REFERENCES

[1] J. W. Aldis, *A polynomial time algorithm to determine maximal balanced equivalence relations*, International Journal of Bifurcation and Chaos, 18 (2008), pp. 407–427.
[2] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, *Synchronization in complex networks*, Physics reports, 469 (2008), pp. 93–153.
[3] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, science, 286 (1999), pp. 509–512.
[4] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, *Complex networks: Structure and dynamics*, Physics reports, 424 (2006), pp. 175–308.
[5] F. Dörfler and F. Bullo, *Synchronization in complex networks of phase oscillators: A survey*, Automatica, 50 (2014), pp. 1539–1564.
[6] M. Golubitsky and I. Stewart, *Nonlinear dynamics of networks: the groupoid formalism*, Bulletin of the american mathematical society, 43 (2006), pp. 305–364.
[7] M. Golubitsky, I. Stewart, and A. Török, *Patterns of synchrony in coupled cell networks with multiple arrows*, SIAM Journal on Applied Dynamical Systems, 4 (2005), pp. 78–100.
[8] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, *Network motifs: simple building blocks of complex networks*, Science, 298 (2002), pp. 824–827.
[9] J. M. Neuberger, N. Sieben, and J. W. Swift, *Invariant synchrony subspaces of sets of matrices*, arXiv preprint arXiv:1908.05797, (2019).
[10] M. E. Newman, *The structure and function of complex networks*, SIAM review, 45 (2003), pp. 167–256.
[11] OEIS, *The on-line encyclopedia of integer sequences*. https://oeis.org/A000110, 2019. Accessed: 2019-010-30.
[12] F. A. Rodrigues, T. K. D. Peron, P. Ji, and J. Kurths, *The kuramoto model in complex networks*, Physics Reports, 610 (2016), pp. 1–98.
[13] I. Stewart, *The lattice of balanced equivalence relations of a coupled cell network*, in Mathematical Proceedings of the Cambridge Philosophical Society, vol. 143, Cambridge University Press, 2007, pp. 165–183.
[14] I. Stewart, M. Golubitsky, and M. Pivato, *Symmetry groupoids and patterns of synchrony in coupled cell networks*, SIAM Journal on Applied Dynamical Systems, 2 (2003), pp. 609–646.
[15] S. H. Strogatz and I. Stewart, *Coupled oscillators and biological synchronization*, Scientific American, 269 (1993), pp. 102–109.
[16] D. J. Watts and S. H. Strogatz, *Collective dynamics of 'small-world'networks*, nature, 393 (1998), pp. 440–442.