Graph Theoretic Foundations of Cyclic and Acyclic Linear Dynamic Networks

Charles A. Johnson * **Nathan Woodbury** * **Sean Warnick** *

* *Information and Decision Algorithms Laboratories, Computer Science Department, Brigham Young University, Provo, UT 84602 USA (e-mail: charles.addisonj@byu.edu, nathanswoodbury@gmail.com, sean@cs.byu.edu)*

Abstract: Dynamic Networks are signal flow graphs explicitly partitioning *structural* information from *dynamic* or *behavioral* information in a dynamic system. This paper develops the mathematical foundations underlying this class of models, revealing structural roots for system concepts such as system behavior, well-posedness, causality, controllability, observability, minimality, abstraction, and realization. This theory of abstractions uses graph theory to systematically and rigorously relate LTI state space theory, developed by Kalman and emphasizing differential equations and linear algebra, to the operator theory of Weiner, emphasizing complex analysis, and Willem's behavioral theory. New systems concepts, such as *net effect*, *complete abstraction*, and *extraneous realization*, are introduced, and we reveal conditions when *acyclic abstractions* exist for a given network, opening questions about their use in network reconstruction and other applications.

Keywords: Dynamic Networks, Network Reconstruction, Model Reduction, Graph Theory

1. INTRODUCTION

Signal flow graphs have a rich history in the control literature. First introduced by Shannon, they were developed and popularized by Mason resulting in Mason's celebrated Gain Formula, see Shannon (1993) and Mason (1952). This technique enabled control engineers to compute the transfer function from a particular input to a particular output using easy-to-understand rules that exploited the topological structure of the network of components comprising the system.

Later work eschewed this signal flow perspective, driven especially by situations where there didn't seem to be natural definitions of inputs and outputs. This work, dubbed *Behavioral Control theory*, focused on the idea of systems as constraints and the resulting behavior of admissible values manifest variables could adopt, see, for example, Willems and Polderman (2013).

Nevertheless, the signal flow paradigm again appeared with the emergence of dynamic networks, especially in the context of network reconstruction: that is, identifying the topology and dynamics of modules in the network from data (Gonçalves and Warnick (2008)). Incorporating various influences from information theory (Etesami and Kiyavash (2014), Quinn et al. (2015), and Subramanian et al. (2017)), computer science (Pearl (2014)), Bayesian statistics (Koller and Friedman (2009)), and system identification (Dankers et al. (2014)), these methods revisited the problem of characterizing signal flow in complex systems.

This paper takes a unique perspective to review some of the most important results from the theory of dynamic networks. Using graph theory as a vehicle to systematically decouple structure and dynamics in the representation of dynamic networks, this paper systematically builds the theory of linear dynamic networks from basic definitions, some of which are nonstandard and new to graph theory, some of which are nonstandard and new to signal flow graphs, and some of which are new to both. The basic idea is that dynamics in the system appear through the choice of algebraic field from which labels are chosen on nodes and edges–all the graph results discussed here with real-valued labels (for ease of exposition) become relevant for dynamic networks when the field is changed to rational functions of a complex variable.

In particular, contributions of this work include detailing the structural roots behind key systems concepts such as system behavior, well-posedness, causality, abstraction, and realization. New results about the existence and construction of acyclic abstractions are included, and new concepts of the completeness of abstractions and extraneousness of realizations that lay the foundation for understanding the structural roots of controlabillity, observability, and minimality are introduced. We end the paper by asking how complete acyclic abstractions may facilitate the reconstruction of complex cyclic networks.

2. FOUNDATIONS

Definition 1. A *directed graph* is a pair $(\mathcal{V}, \mathcal{E})$ where

- $\mathcal{V}$, called the node set, is a finite, totally ordered set and
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is called the *directed edge set.*

Let n, called the *order* of the node set, be the cardinality of $\mathcal{V}$. □

Definition	Summary
Node Labels and Edge Weights	Elements of an algebraic field, $\mathbb{F}$. When $\mathbb{F} = \mathbb{R}$, the model represents a standard weighted digraph. When $\mathbb{F}$ is the field of rational functions over $\mathbb{C}$ then the model represents LTI dynamic networks.
Well-Labeled Nodes	Set of node weights that satisfies linear constraints imposed by the edge weights.
Behavior of a Digraph	The set of all well-labeled node labelings.
Well-Labeled Graph	A graph whose behavior is constrained to be a unique value.
Ill-Labeled Graph	A graph which is not well-labeled.
Net Effect	The cumulative influence of node labels on each other. For LTI dynamic networks this is the transfer function.
Bipartite Digraph	A weighted digraph whose nodes are partitioned into two subsets. All edges are directed from the "input" subset.
Open Digraph	A well-labeled graphical union of some digraph with a bipartite digraph. Allows for an interpretation of edges as computational dependence. DNFs, DSFs and LDGs are all special examples of open digraphs.
Abstraction	A representation of an open digraph with fewer internal edges that shares behavior and nodes with the digraph it represents.
Realization	The open digraph we represent with an abstraction.
Complete Abstraction	An abstraction that preserves a subset of nodes; for example, all sink and source nodes.
Extraneous Realization	A realization of a non-complete abstraction; in the above example, this would be a realization with at least one sink or source node missing in its abstraction.

Table 1. A table summarizing the novel/non-standard definitions in this paper.

Definition 2. An *undirected graph* is a simple directed graph, $(\mathcal{V}, \mathcal{E})$, with an additional *undirected edge symmetry constraint*, that if $(v_i, v_j) \in \mathcal{E}$ then $(v_j, v_i) \in \mathcal{E}$. □

Definition 3. The set of *labels* or *weights*, $\mathbb{F}$, is any field with field operations $+$ and $\cdot$, where we use standard multiplication notation for the $\cdot$ operation, however it may be defined, and other standard notation, e.g. for the additive and multiplicative identities, 0 and 1, respectively.

Definition 4. A *weighted (un)directed graph* is the pair, $(\mathcal{V}, \mathcal{E})$, with labeling functions x and w, where:

- $x : \mathcal{V} \to \mathbb{F}$ that assigns exactly one value in $\mathbb{F}$ to every element of $\mathcal{V}$. Notice that x is an element of a vector space $x \in \mathcal{X} \triangleq \mathbb{F}^n$, where vector addition and scalar multiplication are defined pointwise.
- $w : \mathcal{E} \to \mathbb{F}$ that assigns exactly least one value in $\mathbb{F}$ to every element of $\mathcal{E}$. □

There is a body of work concerned with constructing labeling functions x and w to meet various properties, cf. Gallian (2009), such as graph coloring problems, etc. In some respects this work also focuses on labelings that satisfy particular conditions related to the role of graphs in modeling distributed computation.

2.1 Well-Labeled Digraphs, Net Effect, and Graph Behavior

Recall from Definition 1 that $\mathcal{V}$ is a totally ordered set. This means that we may index nodes according to this ordering. Let $v_i \in \mathcal{V}$ be the i^{th} node.

Definition 5. Given a weighted directed graph, $(\mathcal{V}, \mathcal{E}, x, w)$, the matrix $W \in \mathbb{F}^{n \times n}$, called the *weighted adjacency matrix*, where

$$W_{ij} \triangleq \begin{cases} w(v_j, v_i) & \text{if } (v_j, v_i) \in \mathcal{E} \\ 0 & \text{otherwise,} \end{cases}$$ □

Usually it will be convenient to refer to a weighted (un)directed graph as $(\mathcal{V}, \mathcal{E}, x, W)$ instead of $(\mathcal{V}, \mathcal{E}, x, w)$. Also, note that there are two types of zero entries in W. First, a zero can arise because a potential edge is not part of the graph, indicating "missing" edges. We call these zeros *structural zeros*. On the other hand, a zero can arise because an existing edge in the graph is labeled with the value of zero; these are called *non-structural* zeros. Although there is no way to distinguish these different types of zero from W alone, $\mathcal{E}$ makes the difference clear.

Weighted (un)directed graphs can be effective models of distributed computation when the structure of the graph induces constraints on the admissible values of node and edge labels. We accomplish this by choosing values of node and edge labels from the same field, and interpreting the interaction of nodes and adjacent edges in terms of the field operations. In particular, admissible labelings are those where the value of every node label equals a quantity computed from the values of edges that target the node and the values of nodes in their source sets.

Definition 6. The nodes of a graph $\mathcal{D} = (\mathcal{V}, \mathcal{E}, x, W)$ are *well-labeled* if they satisfy:

$$x = Wx. \tag{1}$$ □

Definition 7. The *behavior*, $\mathfrak{B}$, of an n^{th} order digraph $\mathcal{D} = (\mathcal{V}, \mathcal{E}, y, W)$ is a subset of $\mathbb{F}^n$ given by:

$$\mathfrak{B}(\mathcal{D}) = \{y \in \mathbb{F}^n | y = Wy\}.$$ □

Example 1. Multidigraphs and Behavioral Equivalence. In some applications one may consider *Multidigraphs*, that is, digraphs that admit multiple parallel edges between the same pair of nodes, such as that in Fig. 1. In this case we see that the node labels are subject to additional constraints defined by the parallel edges, but the algebraic properties of the edge labels enable the construction of a *behaviorally equivalent* simple digraph with edge label

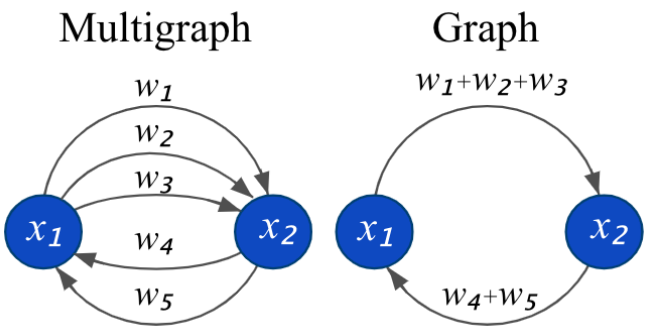

Fig. 1. Multidigraphs (left) can be associated with nominal digraphs with corresponding edge weights (right) such that the graphs are *behaviorally equivalent*. This is possible because of the algebraic properties of the label set $\mathbb{F}$.

w_{ji} equal to the sum of all the parallel edge labels from node v_i to node v_j in the multidigraph. In this way the simple digraph can be seen to have the same behavior as the original multidigraph. □

Some well-labeled digraphs will not interest us because they will not constrain the node labels to take on *unique* values.

In particular, note that rearranging the equation $y = Wy$ yields

$$(I - W)y = 0,$$

suggesting that any combination of labels y that happen to form a vector in the null space of $(I - W)$ will not only be in the behavior of $\mathcal{D}$, but so will any scaling of y with an arbitrary element of $\mathbb{F}$. As a result, labelings such that $(I-W)$ is not full rank admit multiple solutions y, leading to the following definition.

Definition 8. A graph $\mathcal{D} = (\mathcal{V}, \mathcal{E}, y, W)$ is said to be *well-labeled* if $(I - W)$ is nonsingular and *ill-labeled* otherwise. Also, we associate a matrix $G \triangleq (I - W)^{-1}$, called the *net effect matrix*, with every well-labeled graph $\mathcal{D}$. □

Example 2. Net Effect vs. Transitive Closure. If $\mathcal{V}$ is a set with n members, then a *binary relation*, $\mathcal{R}$, on $\mathcal{V}$ is a subset of $V \times V$, and thus is associated with a corresponding digraph, $\mathcal{D}(\mathcal{R})$. The *transitive closure* of $\mathcal{R}$, denoted $\mathcal{R}^*$, is the smallest extension of $\mathcal{R}$ that is *transitive*. Graphically this means that if there is a path from any node v_i to any node v_j in $\mathcal{D}(\mathcal{R})$, then there is a direct edge (v_i, v_j) in $\mathcal{D}(\mathcal{R}^*)$. The graph characterized by the net effect matrix G of $\mathcal{D}(\mathcal{R})$ usually is $\mathcal{D}(\mathcal{R}^*)$, but sometimes the weights in G can cancel just right leaving a net effect of zero even though paths between the relevant nodes exist.

Consider, for example, a weighted digraph characterized by weighted adjacency matrix

$$W = \begin{bmatrix} 0 & 0 & 0 \\ w_{21} & 0 & 0 \\ w_{31} & w_{32} & 0 \end{bmatrix} \tag{2}$$

with net-effect matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ w_{21} & 1 & 0 \\ w_{21}w_{32} + w_{31} & w_{32} & 1 \end{bmatrix}. \tag{3}$$

Thus we see that when $w_{31} = -w_{21}w_{32}$ then $g_{31} = 0$, that is, the net effect matrix has an exact cancellation even though there are paths from v_1 to v_3, indicating that the transitive closure has an edge from v_1 to v_3. □

Graphs with a well-defined net effect matrix are precisely those that lay the foundation of our analyses. Although the only admissible node labeling of such graphs *in isolation* is $y = 0$, the fact that this is the *unique* solution satisfied by the graph operating on the node labels becomes foundational to subsequent results.

2.2 Bipartite and Open Digraphs

A special kind of digraph that will play an important role in this analysis is one that partitions its nodes into two sets with edges emanating only from one to the other. By treating the labels of source nodes as independent variables and those of target nodes as dependent variables, these structures introduce a particular meaning to the directionality of edges in our models, as described below.

Definition 9. A digraph of order $p + m$,

$$\mathcal{D}_B = (\mathcal{V}, \mathcal{E}, \begin{bmatrix} y \\ u \end{bmatrix}, W),$$

is said to be *bipartite* if its node set, $\mathcal{V}$, can be partitioned into two subsets, $\mathcal{V} = \{\mathcal{Y}, \mathcal{U}\}$, with corresponding node labels, $y \in \mathbb{F}^p$ and $u \in \mathbb{F}^m$ such that the corresponding weighted adjacency matrix W has the structure:

$$W = \begin{bmatrix} 0 & V \\ 0 & 0 \end{bmatrix}, \text{ suggesting } \begin{bmatrix} y \\ u \end{bmatrix} = \begin{bmatrix} 0 & V \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ u \end{bmatrix}.$$

- Note the overloading of notation, where y generally represents all node labels but is also used to represent one part of the partitioned labels in bipartite graphs.
- It is easy to see that every bipartite digraph is well-labeled, since $I - W$ will always be full rank.
- The second set of behavioral equations above, enforcing that $u = 0$, is inconsistent with the interpretation of u as independent variables. As a result, we only enforce the first set, redefining the *behavior* of a bipartite digraph to be the set

  $$\mathfrak{B}(\mathcal{D}_B) \triangleq \{[y'\ u']' \in \mathbb{F}^{p+m} | y = Vu\}$$

  for any value u, thus making admissible values for y a well-defined linear function of the (arbitrary) values of u. Similarly, the *net effect* matrix of a bipartite digraph is simply V, characterizing how u effects y. The matrix V is the upper right hand entry of $(I - W)^{-1}$.
- This interpretation of y as *dependent* variables and u as *independent* variables implies:
  - The behavioral equation $y = Vu$ inherits the interpretation of *assignment*, not merely equality,
  - Directionality of edges in the graph thus correspond to *computational dependence*,
  - It is reasonable to interpret u as *inputs* and y as *outputs* in the graph,
  - While the only behavior of a well-labeled digraph is $y = 0$, a bipartite digraph's behavior is an r-dimensional subspace in $\mathbb{F}^{p+m}$, where r is the rank of V.

- We denote bipartite digraphs as

$$\mathcal{D}_B = (\mathcal{Y} \cup \mathcal{U}, \mathcal{E}, \begin{bmatrix} y \\ u \end{bmatrix}, V)$$

and identify properties of V with $\mathcal{D}_B$, specifically:
  - When V is 1-1 we also call $\mathcal{D}_B$ 1-1,
  - When V is onto we also call $\mathcal{D}_B$ onto. □

Bipartite graphs allow the modeling of *open* systems by relaxing constraints on u in the definition of their behavior and treating them as independent variables driving the behavior of y. This idea of an open system, driven by independent variables u, can be generalized by considering the graphical union of a (nominal, or *closed*) digraph with a bipartite digraph.

Definition 10. A *well-labeled*, *open* (or just *open* for short) digraph of order $p + m$ is the graphical union of a well-labeled (closed) digraph of order p and size $N \leq p^2$, $\mathcal{D}_C = (\mathcal{Y}, \mathcal{E}_C, y, W)$, and a bipartite graph of order $p+m$, $\mathcal{D}_B = (\mathcal{Y} \cup \mathcal{U}, \mathcal{E}_B, \begin{bmatrix} y \\ u \end{bmatrix}, V)$, yielding

$$\mathcal{D}_O = (\mathcal{Y} \cup \mathcal{U}, \mathcal{E}_C \cup \mathcal{E}_B, \begin{bmatrix} y \\ u \end{bmatrix}, [W \ V]).$$

- We characterize the *order* of the open digraph, $\mathcal{D}_O$, by the tuple, (p, m), and sometimes distinguish m as the *input order* and p as the *output order*.
- The *internal size* of an open digraph is defined to be N, the number of edges in $\mathcal{D}_C$.
- The *net effect matrix* of a well-labeled open digraph is computed to be $G \triangleq (I - W)^{-1}V$. The matrix G characterizes how u affects y, and can be found as the appropriate entry in the inverse of $(I - \begin{bmatrix} W & V \\ 0 & 0 \end{bmatrix})$.
- The behavior of the open digraph, $\mathcal{D}_O$, is given by

$$\mathfrak{B}(\mathcal{D}_O) \triangleq \{[y' \ u']' \in \mathbb{F}^{n+m} | y = (I - W)^{-1}Vu\}. \quad \square$$

  - Characterizing behavior of the open digraph is only possible when the digraph is well-labeled, ensuring that the relation between y and u has a unique solution.
  - Like in bipartite graphs, the behavior of a well-labeled open digraph is an r-dimensional subspace in $\mathbb{F}^{p+m}$, where r is the rank of the matrix, $G_{12} = (I - W)^{-1}V$.
  - Interpretation of u as an independent variable and y as an observed variable suggests directionality of edges in the graph correspond to *computational dependence*, and even the edges in W inherit this interpretation from $\mathcal{D}_B$.

Different combinations of W and V may yield the same net effect and behavior. For simplicity, we often may refer to $\mathcal{D}_O$, by (y, u, W, V) or just (W, V), called the *network function*, or sometimes the *dynamic network function* (i.e. DNF) when $\mathbb{F}$ is a functional field, e.g. rational functions of a complex variable. Dynamic network functions where $V = I$ and inputs u are (unobserved) independent random processes are referred to as *linear dynamic graphs* (LDG) by Materassi and Salapaka (2012).

Sometimes we only consider network functions with no self loops, suggesting the diagonal elements of W are necessarily zero. In this case we use the notation (Q, P) for (W, V) and call the pair (Q, P) the *structure function* or the *dynamical structure function* (i.e. DSF) when appropriate. Dynamical structure functions for which the inputs are not observable are also called *linear dynamic influence models* (LDIM) after the work of Materassi and Salapaka (2019).

We often slightly abuse notation by using the network functions, i.e. (W, V), or (Q, P), or (Q, I), to refer to the open digraph characterized by these parameters. Moreover, notice that every well-labeled open digraph,

$$\mathcal{D}_O = (\mathcal{Y} \cup \mathcal{U}, \mathcal{E}_C \cup \mathcal{E}_B, \begin{bmatrix} y \\ u \end{bmatrix}, [W \ V])$$

characterizes an associated *bipartite* digraph over the same input and output node sets, given by:

$$\mathcal{D}_B = (\mathcal{Y} \cup \mathcal{U}, \mathcal{E}_C \cup \mathcal{E}_B, \begin{bmatrix} y \\ u \end{bmatrix}, (I - W)^{-1}V).$$

These digraphs share the same net effect and behavior, leading naturally to the notions of compatibility, abstractions and realizations of open digraphs.

2.3 Digraphs as Models of Dynamic Systems

Digraphs are effective models of linear dynamic systems when one chooses $\mathbb{F}$ to be a functional field, such as rational functions of a complex variable. In this case, values of node variables should be understood as the Laplace transform of the corresponding value of of the node variable as a function of time, and values of edge weights should be interpreted as the transfer function of the corresponding single-input, single-output dynamic system. In this way digraphs can represent dynamic systems in the frequency domain, and their net effect matrix is each system's corresponding transfer function matrix.

Alternatively, one could choose $\mathbb{F}$ to be the set of time distributions obtained as the inverse Laplace transforms of rational functions, with multiplication understood as convolution. Then the digraph would represent a dynamic system in the time domain, and its net effect matrix becomes the system's convolution kernel matrix.

Thus we see that the graph formalism described here enables a study of the structural properties of systems, regardless of details about whether the model is defined over the frequency domain or the time domain, or whether the system is dynamic or not, or stochastic or not, etc. As long as the labels belong to a field, all of the properties derived here, and in the subsequent analysis, will hold.

3. ABSTRACTIONS AND REALIZATIONS

Open digraphs include closed and bipartite digraphs as special cases, and thus they become the focus of our analysis. For example, a closed digraph characterized by adjacency matrix W is well modeled by an open digraph with network function (W, I), where I is the appropriate sized identity matrix. We see here that in both cases the net effect is given by $(I - W)^{-1}$. Similarly, a bipartite digraph characterized by adjacency matrix V is well modeled by an open digraph with network function $(0, V)$, as both will have the same net effect matrix, given by V. The key to modeling one system with another is that the net effect, and hence the behavior, is preserved.

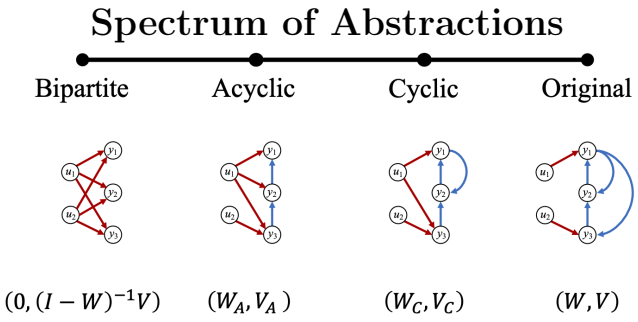

Fig. 2. A dynamic network (i.e. "Original"), characterized by the DNF, (W, V), generates a rich spectrum of abstractions, characterized by fewer internal (i.e. blue) edges. The most extreme of these abstractions is a unique bipartite network, the *black box* representation of the system, characterized by the DNF, $(0, (I - W)^{-1}V)$ and *no* internal (i.e. blue) edges. Along the way are (generally) both cyclic and acyclic abstractions; examples of each are shown.

Given any network function, (W, V), there are many other combinations, $(\bar{W}, \bar{V})$, that characterize the same behavior. We think of these digraphs as different computation structures that result in the same behavior, the way a transfer function and its different state-space realizations all have the same input-output behavior.

In particular, notice that given any network function, we can form a spectrum (see Figure 2) of behaviorally equivalent network functions spanning from the original to its unique digraph with minimal internal structure, given by $(0, (I - W)^{-1}V)$. Every step along this spectrum is characterized by the number of internal edges, preserved from W, which leads us to the notions of *abstraction* and *realization*.

Definition 11. (Abstraction and Realization). Given two open digraphs characterized by $D_R = (W_R, V_R)$ and $D_A = (W_A, V_A)$, D_A is an *abstraction* of D_R if:

(1) (Nodes) $\mathcal{D}_A$'s input and output node sets are (not necessarily strict) subsets of the input and output node sets of $\mathcal{D}_R$,
(2) (Behavior) The net effect of $\mathcal{D}_A$ equals the appropriate submatrix of the net effect of $\mathcal{D}_R$, and
(3) (Internal Size) the internal size (i.e. number of internal edges, see Definition 10) of $\mathcal{D}_A$ is strictly less than the internal size of $\mathcal{D}_R$.

When $\mathcal{D}_A$ is an abstraction of $\mathcal{D}_R$, then $\mathcal{D}_R$ is called a *realization* of $\mathcal{D}_A$. □

One way to characterize the three points of the definition of an abstraction is through full row-rank matrices C_y and C_u, whose rows are orthogonal indicator vectors. Then, given open digraphs $D_R = (y_R, u_R, W_R, V_R)$ and $D_A = (y_A, u_A, W_A, V_A)$, D_A is an *abstraction* of D_R if:

(1) $C_y y_R = y_A$, $C_u u_R = u_A$,
(2) $C_y(I - W_R)^{-1}V_R C_u^T = (I - W_A)^{-1}V_A$ and
(3) $N_A < N_R$.

When C_y and C_u are appropriate sized identity matrices, we see that $(0, (I - W_R)^{-1}V_R)$ is a special abstraction, because it is 1) unique, 2) bipartite, and 3) an abstraction of all other abstractions of $\mathcal{D}_R$, making it the "most abstract" of all. This "black box" representation of the system grounds the spectrum of behaviorally equivalent networks constructed as subgraphs from the original digraph.

We finally note that the above definition and characterization of abstraction is consistent with a number of network abstractions discussed in the literature. For example,

(1) An *edge abstraction* is an abstraction where C_y and C_u are both square,
(2) A *node abstraction* is an abstraction where C_y and/or C_u are not square,
(3) A *hollow abstraction* is an edge abstraction where W_A is a hollow matrix,
(4) An *immersion* is a node abstraction followed by a hollow abstraction, if necessary, to end up with no self-loops. See Woodbury (2019), Dankers et al. (2016), Dankers et al. (2017), Linder and Enqvist (2017), and Woodbury and Warnick (2019) for more details on these abstractions.

3.1 Complete Abstractions and Extraneous Realizations

Abstractions provide a systematic way to produce structurally simplified representations of complex systems. Nevertheless, they can retain full information about the system's effect, but not all do. This section details one of the most important property of abstractions: completeness.

Definition 12. An abstraction, $\mathcal{D}_A = (y_A, u_A, W_A, V_A)$ of $\mathcal{D}_R = (y_R, u_R, W_R, V_R)$ is *complete with respect to* y_R^* *and* u_R^*, where y_R^* is a subset of nodes in y_R, i.e. $y_R^* \subset y_R$, and $u_R^* \subset u_R$, if y_A contains all the nodes y_R^* and u_A contains all the nodes in u_R^*. □

When an abstraction, $\mathcal{D}_A$, of a given network $\mathcal{D}_R$ is complete with respect to *all* nodes u_R and all the *sink* nodes in y_R, then we say it is a *complete abstraction*, and $\mathcal{D}_R$ is a *non-extraneous realization* of $\mathcal{D}_A$. If $\mathcal{D}_A$ is *not* a complete abstraction of $\mathcal{D}_R$, then $\mathcal{D}_R$ is an *extraneous realization* of $\mathcal{D}_A$.

The basic intuition about complete abstractions is that information about the edges in the original network is never entirely lost in the abstraction process; it is only compressed and moved around within the network. On the other hand, extraneous realizations of a network add new edge information that was lost in the given network as an abstraction of these realizations.

For example, consider Figure 3. Suppose $\mathcal{D}_R$ is the original network with $\mathcal{D}_B$ and $\mathcal{D}_C$ as two separate abstractions. The weighted digraph $\mathcal{D}_B$ is a complete abstraction; note that the resulting edge weight of the single edge in the abstraction contains information from both of the realization's edges $(A, B) = a$ and $(B, C) = b$. However, the edge weight in the abstraction $\mathcal{D}_C$, which is not complete, contains no information about edge (B, C) from the original network, $\mathcal{D}_R$.

Non-extraneous realizations are critical to understanding *network minimality*. If the realization of a network is extraneous, it has introduced additional (hence extraneous) information into the network that was not present in the original network. Thus, a minimal network realiza-

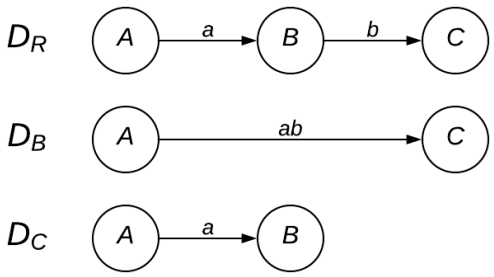

Fig. 3. A network $\mathcal{D}_R$ with an abstraction that is complete ($\mathcal{D}_B$) and an abstraction that is not complete ($\mathcal{D}_C$).

tion must necessarily be non-extraneous, see Johnson and Warnick (2020).

4. ACYCLIC ABSTRACTIONS

Abstractions are structurally simplified representations of systems that preserve net effect, or the behavior of the system. One way to simplify the structure of a system is to remove cycles, resulting in an acyclic abstraction. Acyclic structures are critical for many graph algorithms, and so developing acyclic abstractions may be a first step in applying such algorithms to dynamic systems. This section details when such abstractions can be constructed, when they're complete, and how to build them.

4.1 Acyclic Subgraphs of Digraphs

Definition 13. A *directed acyclic graph* (DAG) is a weighted digraph $\mathcal{D}$ which admits a node ordering so that the adjacency matrix of $\mathcal{D}$ is lower triangular. □

Definition 14. Any node in a digraph, $v' \in \mathcal{V}$, such that $\forall v \in \mathcal{V}, (v, v') \notin \mathcal{E}$, is a *source node.* □

Definition 15. An *arborescence* is a connected, directed acyclic graph, $\mathcal{D}'$, in which any two nodes are connected by at most one path with only one source node, called the *root node.* □

DAGs lend themselves to a number of applications such as allowing one to preform a topological sort on the network, to execute special algorithms such as belief propagation, and to conduct critical path analysis. The problem of deriving the largest acyclic subgraph of an arbitrary graph, $\mathcal{D}$ (referred to as the maximal acyclic subgraph problem), has been proven to be NP-hard, see Karp (1972). Some focus in the literature has been to derive polynomial time techniques for choosing a subgraph with proven bounds on the number of remaining edges, see Berger and Shor (1990), Hassin and Rubinstein (1994) and Charikar et al. (2007).

Definition 16. A digraph *spans* node set $\mathcal{V}$ if there exists a node, $v' \in \mathcal{V}$ for which there is a path from v' to every other node in $\mathcal{V}$.

A *spanning subgraph* of digraph, $\mathcal{D}$, is a digraph, $\mathcal{D}'$ so that $\mathcal{V} = \mathcal{V}'$, $\mathcal{E} \subset \mathcal{E}'$ and $\mathcal{D}'$ spans $\mathcal{V}$. □

Another existing class of acyclic subgraph finding problems is that of finding the minimal spanning arborescence. This problem is source centered, one chooses a root node of a graph and then computes an arborescence which spans the node set from which there are paths from the root node. This is the digraph version of the minimal spanning tree problem. It has had a polynomial solution since the 1960's, see Chu (1965) and Edmonds (1967).

4.2 Single-Source Open Digraphs and DAG Subgraphs

Given a closed digraph, $\mathcal{D}^*$, we choose a node, v', and then attach an input node, v_0, with a weight 1 edge, making an open digraph. We call it $\mathcal{D}$.

Definition 17. A *single-source open digraph* of a closed digraph, $\mathcal{D}^*$, is a closed digraph, $\mathcal{D}$, which consists of $\mathcal{D}^*$ composed with a single input node, v_1 attached to a single node, $v' \in \mathcal{V}^*$ by a single unit-weight edge. □

In this section we consider a DAG subgraph $\mathcal{D}' \subset \mathcal{D}$ in which v_0 is the only source node. This resulting digraph shares node labels and its edges are a subset of the edges of $\mathcal{D}$. We demonstrate that when there is a path from v_0 to every other node in $\mathcal{D}'$ that there exists a labeling of the edge weights in $\mathcal{D}'$ which makes $\mathcal{D}'$ an abstraction of $\mathcal{D}$.

4.3 Directed Acyclic Abstractions

Once we have chosen our acyclic subgraph, $\mathcal{D}'$, we consider if it is possible to choose weights so that the net effect of $\mathcal{D}'$ equals the appropriate column of the net effect of $\mathcal{D}$.

Lemma 1. Every spanning subgraph arborescence of a single-source open digraph has a labeling which makes its net effect equal to the appropriate column of the net effect of the open digraph. □

PROOF Let $\mathcal{E}$ be the set of edges in the original open digraph, and let $\mathcal{E}'$ be the subset of the edges in the spanning subgraph arborescence. We note that the source node of the open digraph is the root node of the arborescence.

We now choose the weight of each edge in $\mathcal{E}'$ as follows. We start by choosing the weight of every edge (v_1, v_{j_1}) where v_{j_1} is distance 1 from v_1 to be $G_{j_1 1}$. Thus, $W'_{j_1 1} = G_{j_1 1}$.

We then assign each link from v_1 to v_{j_2} where v_{j_2} is distance 2 from v_1 to be: $W'_{j_2 1} = \frac{G_{j_2 1}}{G_{k1}}$. Where k is the index of the node so that (v_k, v_{j_2}) is in the unique path from v_1 to v_{j_2}.

We repeat this process until we have reached the set of nodes of maximal finite distance from v_1.

At this point we have that, for any index k, if we let j the distance of v_k from v_1 and m_i be the index of the i^{th} vertex on the unique path from v_1 to v_k, then

$$G'_{k1} = \prod_{i=1}^{j} \frac{G_{m_{i+1}1}}{G_{m_i 1}} = G_{k1}$$

and so $G = G'$. ■

Theorem 1. Every spanning subgraph DAG of a single-source open digraph has a labeling which makes its net effect equal to the appropriate column of the net effect of the open digraph. The resulting DAG with this labeling is an abstraction of the original digraph. □

PROOF We note first that every spanning subgraph DAG has a subset of edges which constitute a spanning subgraph arborescence. We partition the set of edges in the spanning subgraph DAG into this subset and the remaining edges.

Let $\mathcal{E}$ be the set of edges in the original open digraph, and let $\mathcal{E}'$ be the subset of the edges in the spanning subgraph DAG.

Furthermore let $\mathcal{E}^*$ be the edges which are part of the spanning arborescence rooted at v_i and let $\dot{\mathcal{E}}$ be the rest of the edges in $\mathcal{E}'$.

Begin by adding all the edges in $\mathcal{E}^*$ and choose their weights as in the proof of Lemma 1 so that $G' = G$. At this point we add, one at a time, the edges of, $\dot{\mathcal{E}}$. Each time we add an edge, call it (v_s, v_t), we first calculate the weight of (v_s, v_t) to be $\frac{1}{e_s}$, where e_s is the (current) net-effect of v_i on v_s (a finite sum of products, since we are working with a DAG) and then, recursively recalculate the weight of each edge $(v_{s_1}, v_{t_1}) \in \dot{\mathcal{E}} - \{(v_s, v_t)\}$ the same fashion including recalculating the weight of each edge in $(v_{s_2}, v_{t_2}) \in \dot{\mathcal{E}} - \{(v_s, v_t), (v_{s_1}, v_{t_1})\}$ and so on and so forth. The recursion will never result in circular dependencies as the graph is a DAG. Thus in each iteration we preserve the relation that $G = G'$. ■

Corollary 1. Every spanning subgraph DAG of an open digraph has a labeling which makes it an abstraction of the open digraph. □

PROOF We simply note that every spanning subgraph DAG with a labeling which makes its net effect equal to the appropriate column of the net effect of the open digraph fulfills the definition of an abstraction. ■

4.4 DAG Abstractions and Conditions on Completeness

We now return to the DAG abstraction defined in Section 4 discuss completeness of such abstractions. We note that DAG abstractions are edge abstractions, meaning that C_1 and C_2 are both square.

Theorem 2. (Completeness of DAG Abstractions). A spanning subgraph DAG abstraction with a path from the source node to every other node in the digraph is complete. □

PROOF Let W_R and W_A represent the weighted adjacency matrices of the original single-source weighted digraph and the DAG abstraction respectively. With H_R the net effect matrix of the original digraph.

There is only one source node in the DAG abstraction, v_1. For every entry $[W_R]_{ij}$ there exists one entry in H_R which is a function of $[W_R]_{ij}$. Since there is a path from v_1 to every other node in the DAG abstraction, for all index k, there exists at least one edge in the DAG abstraction whose corresponding label in W_A is a function of $[G_R]_k$. Therefore, for all $[G_R]_{ij}$ there is at least one entry of W_A which is a function of $[W_R]_{ij}$ and the abstraction is complete. ■

The following result is a consequence of Theorem 1. It implies that every dynamic network spanned by a DAG has an abstraction whose edges match those of said DAG.

Corollary 2. Given any source node of a DSF there exists a single-input DAG abstraction of the DSF which is complete with respect to all nodes spanned by the source node. □

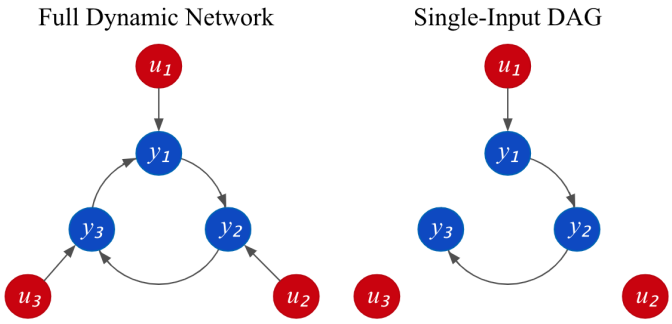

Fig. 4. On the left is a dynamic network and on the right a single-input DAG abstraction. Note that the graphical union of all three such abstractions includes all edges in the original network.

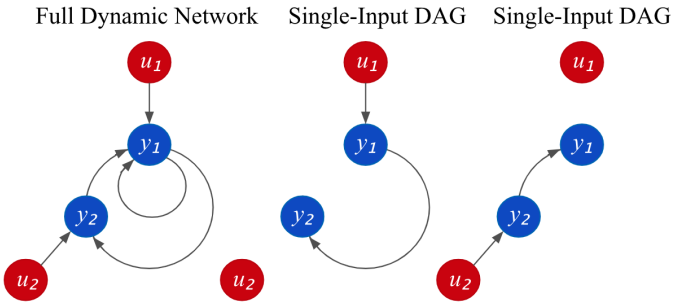

Fig. 5. On the left is a dynamic network and on the right are its two single-input DAG abstractions. Note that their graphical union does not include all loop dynamics.

5. CONCLUSION

This paper demonstrated how open digraphs can represent dynamic networks through careful choice of algebraic field from which to draw node and edge labels. New notions of net effect and behavior were introduced, leading to definitions of abstraction and realization and the associated spectrum of abstractions, effectively linking (in the case of linear time invariant dynamic networks) state space realizations to their transfer functions.

Completeness of abstractions was also defined, along with the associated concept on non-extraneous realization, and graphical characterizations of these properties were given. These notions were argued to be essential for minimality and their associated concepts of controllability and observability of a network, although these ideas are detailed elsewhere (see Johnson and Warnick (2020)).

Finally, directed acyclic graphical abstractions were introduced. Methods for generating DAG abstractions, and conditions for their completeness, were also given.

Note that the structural information contained in each single-input DAG abstraction can be combined to infer several acyclic structures. In some cases, the combined information reveals the entire topology. Figure 4 illustrates this point for a ring structure. In others (see Figure 5) some or none of the cyclic structure may be inferred.

Previous work has given passive data-driven algorithms for the reconstruction of acyclic networks (see Materassi

and Innocenti (2010), Materassi and Salapaka (2012) and Materassi and Salapaka (2013) for such work on LDGs). However, there are strong indications that such techniques when used on a network with cycles need not reconstruct a true DAG abstraction of the network (see, for example Chetty and Warnick (2015)).

Thus we pose to two related questions for future work:

(1) What information regarding cyclic structure can be acquired from complete acyclic abstractions of a dynamic network?
(2) To what extent can acyclic reconstruction techniques give us true acyclic abstractions of a reconstructed cyclic network?

REFERENCES

Berger, B. and Shor, P.W. (1990). Approximation alogorithms for the maximum acyclic subgraph problem. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, 236–243. Society for Industrial and Applied Mathematics.

Charikar, M., Makarychev, K., and Makarychev, Y. (2007). On the advantage over random for maximum acyclic subgraph. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, 625–633. IEEE.

Chetty, V. and Warnick, S. (2015). Network semantics of dynamical systems. In *Conference on Decision and Control*. Osaka, Japan.

Chu, Y.J. (1965). On the shortest arborescence of a directed graph. *Scientia Sinica*, 14, 1396–1400.

Dankers, A., Van den Hof, P.M., Bombois, X., and Heuberger, P.S. (2016). Identification of dynamic models in complex networks with prediction error methods: Predictor input selection. *IEEE Transactions on Automatic Control*, 61(4), 937–952.

Dankers, A., Van den Hof, P.M., Materassi, D., and Weerts, H.H. (2017). Conditions for handling confounding variables in dynamic networks. *IFAC-PapersOnLine*, 50(1), 3983–3988.

Dankers, A.G., Van den Hof, P.M., and Bombois, X. (2014). Direct and indirect continuous-time identification in dynamic networks. In *53rd IEEE Conference on Decision and Control*, 3334–3339. IEEE.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards, B*, 71, 233–240.

Etesami, J. and Kiyavash, N. (2014). Directed information graphs: A generalization of linear dynamical graphs. In *2014 American Control Conference*, 2563–2568. IEEE.

Gallian, J.A. (2009). A dynamic survey of graph labeling. *The electronic journal of combinatorics*, 16(6), 1–219.

Gonçalves, J. and Warnick, S. (2008). Necessary and sufficient conditions for dynamical structure reconstruction of lti networks. *IEEE Transactions on Automatic Control*, 53(7), 1670–1674.

Hassin, R. and Rubinstein, S. (1994). Approximations for the maximum acyclic subgraph problem. *Inf. Process. Lett.*, 51(3), 133–140.

Johnson, C.A. and Warnick, S. (2020). Characterizing network controllability and observability for abstractions and realizations of dynamic networks. In *IFAC World Congress 2020*.

Karp, R.M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations*, 85–103. Springer.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Linder, J. and Enqvist, M. (2017). Identification and prediction in dynamic networks with unobservable nodes. *IFAC-PapersOnLine*, 50(1), 10574–10579.

Mason, S.J. (1952). *On the logic of feedback*. Ph.D. thesis, Massachusetts Institute of Technology.

Materassi, D. and Innocenti, G. (2010). Topological identification in networks of dynamical systems. *IEEE Transactions on Automatic Control*, 55(8), 1860–1871. doi:10.1109/TAC.2010.2042347.

Materassi, D. and Salapaka, M.V. (2012). On the problem of reconstructing an unknown topology via locality properties of the wiener filter. *IEEE Transactions on Automatic Control*, 57(7), 1765–1777. doi:10.1109/TAC.2012.2183170.

Materassi, D. and Salapaka, M.V. (2013). Reconstruction of directed acyclic networks of dynamical systems. In *2013 American Control Conference*, 4687–4692. doi:10.1109/ACC.2013.6580562.

Materassi, D. and Salapaka, M.V. (2019). Signal selection for estimation and identification in networks of dynamic systems: a graphical model approach. *arXiv preprint arXiv:1905.12132*.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

Quinn, C.J., Kiyavash, N., and Coleman, T.P. (2015). Directed information graphs. *IEEE Transactions on Information Theory*, 61(12), 6887–6909. doi:10.1109/TIT.2015.2478440.

Shannon, C. (1993). *Collected papers*. IEEE Information Theory Society.

Subramanian, V.R., Lamperski, A., and Salapaka, M.V. (2017). Network topology identification from corrupt data streams. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 1695–1700. IEEE.

Willems, J.C. and Polderman, J.W. (2013). *Introduction to mathematical systems theory: a behavioral approach*, volume 26. Springer Science & Business Media.

Woodbury, N. (2019). *Representation and Reconstruction of Linear, Time-Invariant Networks*. Ph.D Dissertation, Brigham Young Univ., Provo, UT.

Woodbury, N. and Warnick, S. (2019). Abstractions and realizations of dynamic networks. In *American Control Conference*. Philadelphia, PA.