2020-06-04

Abstractions of Graph Models

Charles Addison Johnson
Brigham Young University

Abstractions of Graph Models

Charles Addison Johnson

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Sean Warnick, Chair
David Wingate
Dennis Ng

Department of Computer Science

Brigham Young University

ABSTRACT

Abstractions of Graph Models

Charles Addison Johnson
Department of Computer Science, BYU
Master of Science

Building models, whether to explain or to predict observed data, is an exercise of describing how the values of observed variables depend on those of others. Black box models only describe relationships between observed variables, and they are evaluated by their ability to accurately describe the values of observed variables in new situations not previously available to the model–such as the output response to a new set of inputs, for example. Black box models describe the observed *behavior* of the underlying system, but they may not correctly describe the way in which the system computes this behavior. White box models, on the other hand, describe the observed behavior and also incorporate hidden, intermediate variables that are used to describe the specific computation the underlying system uses to generate its observed behavior. In this sense, we say the white box model captures the *structure* of the system, in addition to its observed behavior. Since a given white box model may be accurately described by an infinite variety of black box models, all computing the same observed behavior but using different structures to do so, we say that any of these black box models is an *abstraction* of the white box model. This thesis constructs foundational pieces of a unifying theory of linear mathematical abstractions that are central to scientific modeling. It offers a precise description of the spectrums of grey box models linking any white and black box representation.

There are various motivations for having this rich variety of representations of a given system. One key motivation is that of *consilience*, that is, to deepen our understanding of the modeling process by connecting various well developed theories under the umbrella of a broader theory. This work offers a precise relationship between Mason's signal flow graphs [34] and Willem's behavioral systems theory [66], in addition to linking the classical transfer function theory used by Nyquist [44], Bode [3], and Weiner [65] to the state space theory preferred by Kalman [27]. Another motivation comes from the application of *identification* or *learning* a model of the system from data. Learning problems trade off the number of a priori assumptions that one must make about a system, as well as the richness of available data, with the complexity of a model that one is able to confidently learn from measured observations. This work offers insight into these tradeoffs by characterizing them precisely over entire spectrums of grey box models of increasing complexity. A third motivation comes from the application of *vulnerability analysis*, which is the study of sensitivities of system behavior to structural perturbations in a grey-box model describing the attack surface, or representation of the system as visible to a potential attacker.

The main results of this work, and its specific contributions, are as follows:

1. We define new graph-theoretic constructs and use them to create a unified framework for structural abstractions,
2. We demonstrate that there will always exist a complete, structure preserving, acyclic abstraction for every single-input, fully-connected system,
3. We define structural controllability of an abstraction of a system and argue why our definition is good, and
4. We show how complete abstractions preserve structural controllability.

These results were accepted for publication in two papers at the 2020 International Federation of Automatic Control World Congress, each submitted to a different special invited session. These papers comprise Chapters 2 and 3, respectively, of the thesis presented here, and they are expected to appear in print July 2020.

Keywords: Abstractions, Graph-Based Models, Dynamic Systems

ACKNOWLEDGMENTS

I feel like it is a very special and uncommon thing to be able to work on the foundations to a new mathematical area which can describe a concept as fundamental and relevant as abstraction. It has been an eye-opening delight to build and rebuild from the ground up definitions that have to be the standing blocks for a great body of mathematical work. I've got to say thanks to Sean Warnick for enabling this experience and for being an example of how one thinks through the process of inventing new math.

I would furthermore like to thank my committee members for their flexibility and for enabling me to support my wife's educational and personal goals as I worked on my own.

Of course, like many before me, I've also got to thank my brilliant and loving wife for her support and willingness to let me work through this degree and thesis, especially when I had to (chose to) burn that midnight oil.

Table of Contents

List of Figures

Chapter 1

Introduction

This work is concerned with abstractions of a special class of graph based models. Abstractions are coarser-grained models which preserve some properties of the finer-grained models. The models in this thesis can describe the causal relationships between measured variables in a linear time-invariant (LTI) dynamic system, as well as computational dependency in other contexts as well.

The abstractions often represent the same dynamic system or context, in which fewer variables are measured. In a sense the abstractions are different perspectives on the same system. The ability of these abstractions to represent various perspectives has been leveraged in the past to model interesting phenomena. For example, in the context of cyber-infrastructure security, a system user and a malignant attacker may be working with the same model, however, the attacker can only see (or access) a coarser abstraction of the system for the sake of designing attacks [22]. The applications of abstractions, of course, go well beyond system vulnerability analysis.

1.1 Motivation

In many cases, the point of abstraction is to safely ignore the details of how exactly a computation is made so that we can focus our resources on other, perhaps higher-level, problems. The use of abstractions in this manner is hardly restricted to the discipline of computer science. A classic, near-universal example in our day, is driving[1]. When I drive a

[1]This example was taken from Chapter 10 of [47].

car, I don't think about the complex mechanical details behind how the inputs that I give to the car (turning the steering wheel, pushing down on the gas and brake pedals, etc.) are transformed into how the car behaves on the road. The use of abstractions explains how the finite human mind can function in a world with infinite or near-infinite complexity [18, 54].

However, our abstractions break down. Sometimes we need to break open the black-box and understand how the computation is made. I can ignore the inner workings of my car as long as it functions as I expect it to. However, a number of environmental conditions, mechanical or software failures or trauma to the vehicle can make my black-box understanding of the car insufficient. After an accident, or on an icy road, additional details regarding the function and location of various car parts may be essential to properly repair or control the vehicle.

In such circumstances it is almost never necessary to come face to face with the full complexity of your vehicle and the physics that govern its behavior. One is not likely to need to consider the car headlights to change out a flat tire. To properly navigate a complex world one must properly navigate and balance one's abstractions of the world.

Indeed, the world is becoming more complex and new paradigms for human-computer interaction are becoming necessary to take full advantage of the computing resources available. For example, a number of real-world strategic decisions are being made by complex, automated computations. However, humans, in spite of their limited capacity, need to take responsibility for these computer-suggested strategic decisions. Since a human is unable to understand the full details behind why a computer may have recommended such a decision, an abstracted representation of the computation process is a necessary requirement to allow a human decision maker to take responsibility for that decision. Put another way, abstractions of computation give humans alternatives to blindly accepting the computed decision.

In the field of systems theory a mathematical theory of abstraction is under development [29, 68]. It considers structurally distinct graph models which each constrain a system to have the same behavior. Complexity in this paradigm is, therefore, structural complexity,

specifically the number of edges and nodes in the graph model. While the application of this mathematical theory of abstraction has yet to be applied to abstractions in software development, this is one of the motivations underlying this work.

Previous work in development of this theory of abstractions limited the scope of these abstractions to frequency domain representations of linear time-invariant dynamic networks in both continuous and discrete time. However, it has been observed that the results extend quite naturally to time domain representations of the same systems when one redefines multiplication to be the convolution operator (defined as appropriate given a continuous-time or discrete-time formulation) [9]. This work extends this generalization by explicitly allowing the weights in the models to come from any mathematical field. This formulation, therefore includes systems whose weights for computation come from the binary field as well as other finite fields. This step is significant as it expands the potential reach of this theory of abstractions to include finite state-machine models (automata).

1.2 Overview

A key problem that spans a number of fields is that of finding the dependency relationships between variables of interest. For example, given a set of several timeseries, one may wish to model how one timeseries (perhaps representing a stock price or a protein concentration) influences another. In many cases the relationships between these timeseries may be asymmetric and full of nuances. For example, one may affect another only indirectly via a third timeseries, or there may be loops by which a timeseries indirectly affects itself through one or more intermediaries. Ideally, the resulting model would be a digraph whose nodes each represent one of the timeseries and whose edges indicate direct, one way, computational dependency between the timeseries. Each of these edges would have a weight to explain the nature of the dependency. This is known as the problem of network reconstruction. The model we seek is the signal structure of the interdependent timeseries. When there are unmeasured intermediary variables (unmeasured timeseries that are part of the true dependency structure),

the digraph that one learns will only be an abstraction of the true dependency structure. Identifying the dependency structure without measuring all the relevant variables leads to a number of challenges. For example, with no apriori information on the dependency structure, there will likely be an infinite number of weighted digraphs which will explain your measured data, even in the limit of infinite data! Abstractions of these models warrant special attention and care.

This thesis build upon a recent developing theory for structural abstractions of dynamic networks [29, 67, 68]. It does so by returning to and rigorously defining the fundamental structure on which the theory is built. In this community, LTI dynamic systems with partial measurements have been defined by their signal structure. The signal structure may be represented as a graph in which nodes are measured variables and edges represent causal relationships between the variables. These models are abstractions of the original system as structural information about the unmeasured variables (causal relationships between unmeasured variables and all the system variables) is lost in the signal structure.

Our central contribution (broadly speaking) is to more generally explore structural abstractions, including the signal structure already established for LTI dynamic systems. To do so a number of general concepts are defined. The key concept of *net effect* is introduced as one of the three foundational pillars of abstraction. The concepts of a *complete abstraction* and an *extraneous realization* are rigorously defined. The usefulness of this foundation is then explored by contributing results regarding the existence of complete acyclic subgraph abstractions as well as results on the structural controllability of abstractions in the special case of LTI dynamic networks. This work also contributes a discussion of the distinction between networked dynamic systems and dynamic networks.

Models in this thesis can be represented with a special type of graph. In this model, graph edges are directed and each edge has an associated weight. Similarly, each node has a label. Weights and labels are variables with values chosen from some mathematical field, $\mathbb{F}$. The field, $\mathbb{F}$, is a set endowed with two binary operations referred to as *addition* and

multiplication which satisfy a special set of technical conditions (for reference see [48]). In the graph, the direction of the edge implies computational dependency. We invented a concept of *well-labeled* nodes, whose labels satisfy the constraints imposed by the edges, edge weights and the labels themselves. These constraints will be defined in Chapter 2, but essentially, all paths from node x_i to x_j are summed. Each path in the sum is the product of the x_i's label and the path's edge weights. Then x_j is constrained to equal this sum for all $i \neq j$. So, if a graph had two nodes, x, y, and a single directed edge from x to y, y depends on x. This means that for each label x has, there is only one possible way to label y that keeps the node well-labeled.[2] A simple example of these concepts is shown in Figure 1.1.

In this work, abstractions of these models are structural abstractions, and so they involve making a new graph model with fewer edges which still preserves key model properties. The key property we preserve is *net effect*, which captures the way in which node labels are uniquely constrained by each other and edge weights. Net effect also is strongly tied to a graph (not the node labels) being well-labeled. For example, when our graph models are LTI dynamic graph models with clearly identified input and output nodes, the net effect becomes the transfer function matrix (a black-box input-output representation of an LTI dynamic system). You may refer to Figure 1.2 for a simple example of a graph abstraction in the case where $\mathbb{F} = \mathbb{R}$.

Chapter 2 of this work builds out of graph-based modeling efforts [34], often attributed to Samuel Mason, as well Jan Willems' behavioral theory whose modeling efforts focus on the relationship between manifest variables [66]. These perspectives appear to have some mutually incompatible viewpoints. For example, Willems argued strongly against the input-output paradigm associated with the transfer function, a key element of Mason's work and much of classic systems theory. However, dynamical structure function (DSF) theory has made a successful marriage of the concepts. DSF theory is a foundational work which this thesis builds upon. In combining signal-graph theory and behavioral theory, the innovators who

[2]The label of y must be the product of the label of x and the weight of the edge connecting the two nodes, otherwise the node is ill-labeled. This is all under the assumption that x itself is unconstrained.

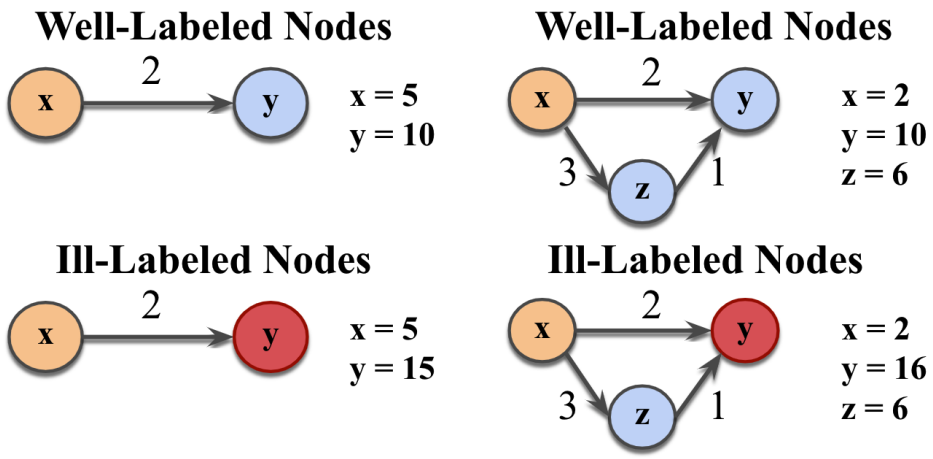

Figure 1.1: This figure depicts open directed graphs with nodes that are well and ill-labeled. We say that the upper left has well-labeled nodes as the label of y satisfies the constraint (imposed by the edge weight) that it be two times the label of x ($10 = 5 \times 2 \neq 15$). The graph in the upper right has well-labeled nodes because the labels of y and z are set to the label of x times the sum of the products of the edge weights on each path ($10 = 2 \times 2 + 6 \times 1 \neq 15$ and $6 = 2 \times 3 = 6$). Note here that x's label is considered an unconstrained variable, we refer to such variables as inputs in this work.

worked on DSF theory had to make some specific modeling and philosophical choices, many of which are discussed in [72]. This work will expose some of those modeling choices by examining and extending this theory of modeling (before applied more or less strictly to LTI systems) in a precise manner. Every element of these models will be defined and examined in a blend of behavioral and signal-flow graph theory with the aim to explore and understand model abstractions.

Chapter 3 of this work will then apply this theory of modeling and abstractions to the question of structural controllability and observability of complex networks (also referred to as networked dynamic systems). As the name suggests, structural controllability considers ones ability to control a network which has a specific graph structure, but arbitrary parameterizations (e.g. edge weights on the graph) of that structure. Since the current complex network community views interconnection in these graph models as interconnection between physically distinct systems (called subsystems) as opposed to the direct signaling

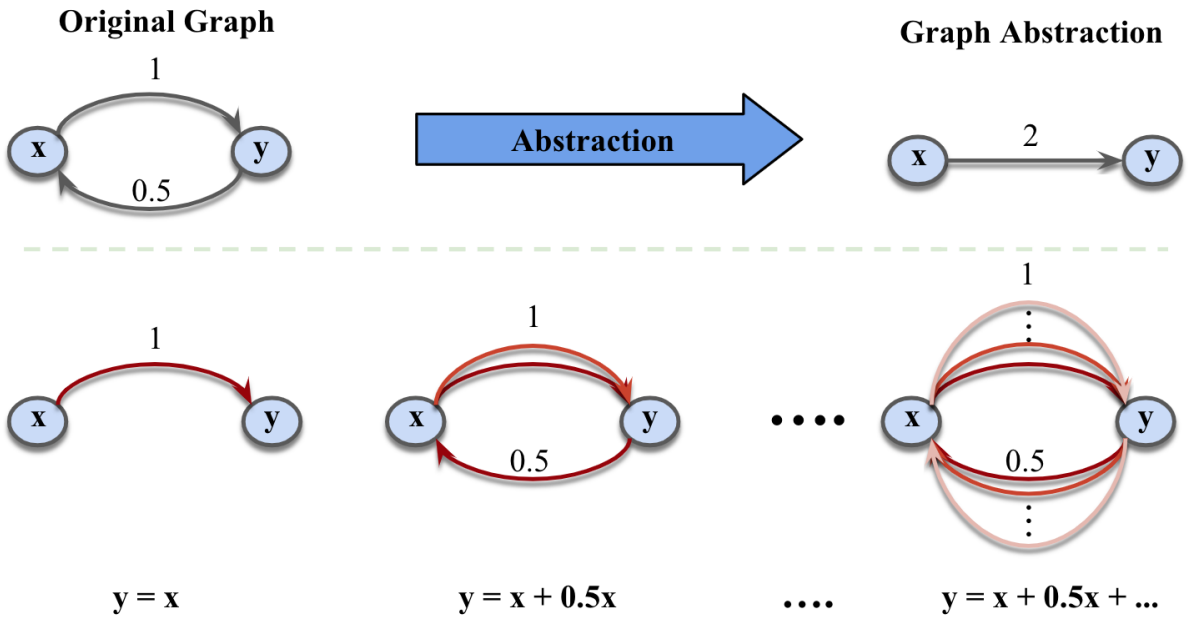

Figure 1.2: This figure depicts a graph as well as an abstraction of that graph. Below this it demonstrates the infinite paths which sum to constrain node y. Note that each constrains the nodes in the same way, but the abstraction has fewer edges and has lost information of how the left node is constrained by the right one. To for the graph to be well-labeled we must have $y = x + \frac{1}{2}x + \frac{1}{4}x + ... = (1 + \frac{1}{2} + \frac{1}{4} + ...)x = (\sum_{n=0}^{\infty}(\frac{1}{2})^n)x = 2x$, where the last equality is due to the fact that the sum described is a geometric series. One sees that the net effect, in this case, from x to y is 2.

relationships between measured variables, this chapter will illustrate how our class of graph based models is distinct. This is a subtle and yet key point as one could possibly conflate these two model classes and, therefore, draw erroneous conclusions based off of valid abstractions made with the wrong model class in mind.

Note that Chapters 2 and 3 will have been published as their own papers in the 2020 IFAC World Congress in Berlin. However, they are united in the sense that the first lays a foundation for the second and the second addresses a prominent community of complex systems theorists with the basics of this framework. It also shows how this framework can be used to characterize structural controllability and observability of abstractions of dynamic networks. These two chapters are presented as the main body of this thesis.

1.3 Related Work

This section does not discuss all relevant work to this thesis as each of the following chapters will cite publications with specific relevance to the content of the chapter. However, I hope that this brief survey of the literature can help to further contextualize and motivate the contributions of this work.

The class of graph models that inspired the contributions of this thesis originally arose to address the question of reconstructing a dynamic network from data, this structured model is commonly referred to as the Dynamic Structure Function [21]. The notion of abstracting away the structural complexity associated with hidden state variables was always key to the concept and derivation of the Dynamic Structure Function. This connection to abstractions has become more explicit in recent years.

Indeed, a theory of model abstractions is advantageous to a data scientist trying to reconstruct the true underlying computation structure of a system ($f(x)$), instead of just an input-output equivalent model. Past work has defined, for example, strict identifiability conditions for reconstructing the signal structure of measured variables from a dynamic system [19] as well as a network of interconnected subsystems from data [6, 59]. Furthermore,

[5] has shown that, when the identifiability conditions are not met, that any structurally admissible computation structure can equally explain the input-output dynamics observed. This is especially daunting as a network's possible structures grow combinatorially with the size of the network.

One line of relevant research on model abstractions focuses on network immersions (node abstractions followed by hollow abstractions). This research includes algorithms for computing such abstractions [64], and leveraging them to answer questions about network identifiability (necessary conditions for network reconstruction) [1, 19, 63]. These abstractions are very nice to work with as the resulting structure of the abstraction will always have a corresponding set of edge weights which will make the graph a true abstraction of the original system. Other methods of abstraction need not satisfy this property. This is why results that prove the existence of abstractions with desirable properties (such as structure-preserving, or acyclic) may be useful.

Much recent work studying network immersions has focused on adding additional noise assumptions and constraints to the fundamental questions associated with reconstructing LTI dynamic networks and then providing additional solutions and insights to those circumstances [36, 38, 40]. In other cases it is assumed that the network topology is known but that the dynamics still need to be identified [58]. In contrast, this work chiefly expands the theoretical framework of this graph-based model class at a very fundamental level.

1.4 Contributions

This work considers models in which the graph structure constrains the node values as dependant variables. Information in these graphs pass along paths in a multiplicative fashion and combines paths via addition. This does not restrict these models to standard addition and multiplication, but rather allows one to choose definitions for multiplication and addition that meet the field properties [48]. For example, if we define addition as the XOR operation and AND as the multiplication operation and our set of node and edge labels is $\{0, 1\}$, then

our network is defined over the binary field. Claude Shannon's thesis demonstrated the flexibility and power of modeling over the binary field [53]. These models, in general, include the full descriptive power of discrete-time and continuous-time LTI systems and much more. Structuring the framework for these models to be so general is the first contribution of this thesis.

This work extends previous theory of graph modeling and abstractions (see [29, 68]) from the instance of LTI dynamic networks to general graphical models with node and edge values taken from an arbitrary field. This is in contrast to the almost universal thrust of the body of work which focuses on structural abstractions of signal-structure between observed variables. The general body of the literature focuses nearly exclusively on results and formulations of LTI networks. In the presented, framework LTI networks are special cases of these graph models when the node and edge values are taken from the field of rational functions over $\mathbb{C}$, or the field constructed from such rational functions via the Laplace or Z Transform with multiplication defined as convolution and addition defined as pointwise function addition.

Chapter 3 will discuss a notion of controllability and observability for these graph models. It is somewhat surprising that there is any connection between these models and controllability in the first place. This is due to the fact that system concepts such as controllability and observability were originally defined to be state-space concepts. This work is not the first to connect graph representations of systems to concepts like controllability. For example, [31] began a stream of results continuing into the modern day on structural controllability. This work's contribution to that set of results is to characterize the structural controllability of abstractions.

Another contribution of this work is to introduce and motivate the importance of complete abstractions. This notion separates abstractions which drop computation and those which preserve it in a structurally simpler manner. On graphs, the interpretation of a complete abstraction is simple and we see that computation is lost when source and sink

nodes (or connected components) are abstracted away from the model. Chapter 3 of his work argues that incomplete abstractions do not permit the preservation of the property of structural control for LTI networks.

This work also proves that all single input networks have their own complete acyclic abstraction which preserves the original structure of the network. This means that our complete abstraction's structure is acquired only by deleting edges, none are added as often occurs when abstracting via network immersions. This problem is not trivial as these abstractions must preserve net effect (the input-output relationship of the entire system).

These contributions provide a framework for reasoning about and comparing structural model abstractions. This framework in and of itself is a valuable lens for analyzing the abstraction process. This is a key contribution as understanding the quality of an abstraction enables one to make good abstraction decisions. Abstractions can be judged not only on the amount of complexity that they remove or keep, but also on other desirable properties that they may have, such as being structure-preserving, acyclic, or complete. For example, the value of the quality of completeness depends on one's goals for the model. Chapter 3's main results indicate that complete abstractions are very valuable when one wants to preserve structural controllability of the abstraction.

Chapter 2

Graph Theoretic Foundations of Cyclic and Acyclic Linear Dynamic Networks

Accepted as a Special Session Paper to be presented in the 2020 IFAC World Congress in Berlin

Dynamic Networks are signal flow graphs explicitly partitioning *structural* information from *dynamic* or *behavioral* information in a dynamic system. This paper explicitly 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.

2.1 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 [52] and [34]. 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, [66].

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 ([19]). Incorporating various influences from information theory ([16], [49], and [55]), computer science ([46]), Bayesian statistics ([30]), and system identification ([14]), 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 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.

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.

2.2 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 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. [17], 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.

2.2.1 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:*

$$x = Wx. \tag{2.1}$$

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. 2.1. 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* 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.*

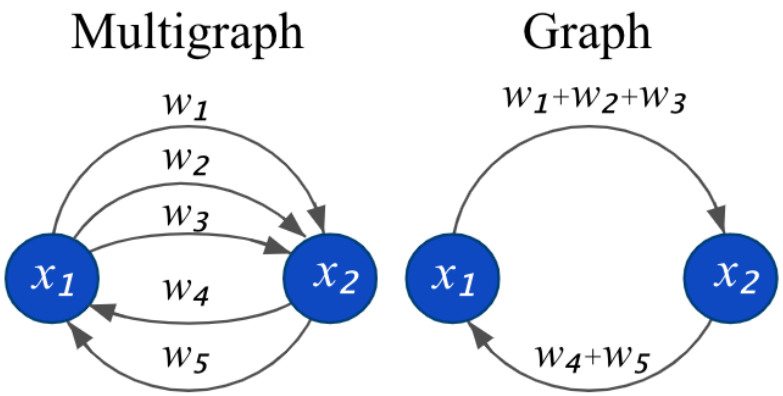

Figure 2.1: Multidigraphs (left) can be associated with nominal digraphs with corresponding edge weights (right) such that the graphs are *behaviorally equivalent*. This is possible because of the algebraic properties of the label set $\mathbb{F}$.

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

$$W = \begin{bmatrix} 0 & 0 & 0 \\ w_{21} & 0 & 0 \\ w_{31} & w_{32} & 0 \end{bmatrix} \tag{2.2}$$

with net-effect matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 \\ w_{21} & 1 & 0 \\ w_{21}w_{32} + w_{31} & w_{32} & 1 \end{bmatrix}. \tag{2.3}$$

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.

2.2.2 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*

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

- *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*

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

  - *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 $\mathcal{D}_B$.*

Definition 11 *A* well-labeled, open *(or just* open *for short) digraph of order $p + n + m$ is the graphical union of a well-labeled (closed) digraph of order n $\mathcal{D}_C = (\mathcal{X}, \mathcal{E}_C, x, W)$ a bipartite graph of order $n + m$ $\mathcal{D}_{B1} = (\mathcal{X} \cup \mathcal{U}, \mathcal{E}_{B1}, \begin{bmatrix} x \\ u \end{bmatrix}, \begin{bmatrix} 0 & V \\ 0 & 0 \end{bmatrix})$, a bipartite graph of order $n + p$ $\mathcal{D}_{B2} = (\mathcal{Y} \cup \mathcal{X}, \mathcal{E}_{B2}, \begin{bmatrix} y \\ x \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ R & 0 \end{bmatrix})$ and a bipartite graph of order $p + m$ $\mathcal{D}_{B12} = (\mathcal{Y} \cup \mathcal{U}, \mathcal{E}_{B12}, \begin{bmatrix} y \\ u \end{bmatrix}, \begin{bmatrix} 0 & D \\ 0 & 0 \end{bmatrix})$, yielding $\mathcal{D}_O = (\mathcal{Y} \cup \mathcal{X} \cup \mathcal{U}, \mathcal{E}_C \cup \mathcal{E}_{B1} \cup \mathcal{E}_{B2} \cup \mathcal{E}_{\mathcal{B}\infty\in}, \begin{bmatrix} y \\ x \\ u \end{bmatrix}, \begin{bmatrix} 0 & R & D \\ 0 & W & V \\ 0 & 0 & 0 \end{bmatrix})$.*

- *We characterize the* order *of the open digraph, $\mathcal{D}_O$, by the tuple, (p, n, m), and sometimes distinguish p as the* output order*, m as the* input order *and n as the latent order.*
- *The* net effect matrix *of a well-labeled open digraph is computed to be*

$$G \triangleq \begin{bmatrix} I & R(I-W)^{-1} & R(I-W)^{-1}V + D \\ 0 & (I-W)^{-1} & (I-W)^{-1}V \\ 0 & 0 & I \end{bmatrix}.$$

- *The behavior of the open digraph, $\mathcal{D}_O$, is given by*

$$\mathcal{B}(\mathcal{D}_O) \triangleq \{[y'\ u']' \in \mathbb{F}^{p+m} | y = (R(I-W)^{-1}V + D)u\}.$$

  - *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, $G_{13} = R(I - W)^{-1}V + D$. This is only possible, though, when the digraph is well-labeled, ensuring that the relation between y and u has a unique solution.*
  - *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 $\mathcal{D}_{B1}, \mathcal{D}_{B2}$ and $\mathcal{D}_{B12}$.*

Note observe the following special cases of the general open digraph:

1. $p = 0$, in this case we have the open digraph: $\mathcal{D}_O = (\mathcal{X} \cup \mathcal{U}, \mathcal{E}_C \cup \mathcal{E}_{B1}, \begin{bmatrix} x \\ u \end{bmatrix}, \begin{bmatrix} W & V \\ 0 & 0 \end{bmatrix})$.
   As before this is the network function.
2. $n = 0$, in this case we have the open digraph: $\mathcal{D}_O = \mathcal{D}_{B12}$.
3. $m = 0$, in this case we have the open digraph: $\mathcal{D}_O = (\mathcal{Y} \cup \mathcal{X}, \mathcal{E}_C \cup \mathcal{E}_{B2}, \begin{bmatrix} y \\ x \end{bmatrix}, \begin{bmatrix} 0 & R \\ 0 & W \end{bmatrix})$.

Different combinations of W, R, V and D may yield the same net effect and behavior. For simplicity, when $p = 0$ or $R = I$ and $D = 0$, we often may refer to $\mathcal{D}_O$, by just (x, u, W, V),

called the *network function*, or sometimes the *dynamic network function* (i.e. DNF) when $\mathbb{F}$ is a functional field.

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 [36].

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 [41].

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,

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

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.

2.2.3 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.

2.3 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.

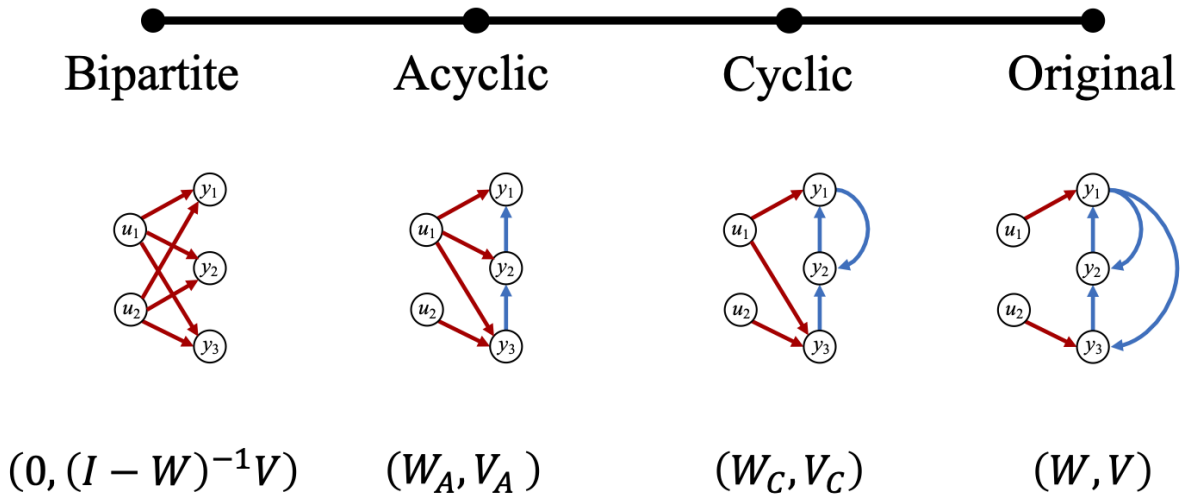

Figure 2.2: A dynamic network (i.e. "Original"), characterized by the DNF, (W, V), generates a rich spectrum of abstractions, characterized by fewer internal (i.e. blue) edges. The most extreme of these abstractions is a unique bipartite network, the *black box* representation of the system, characterized 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 2.2) 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 12 (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 [67], [12], [13], [32], and [68] for more details on these abstractions.

2.3.1 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 13 *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 2.3. 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)

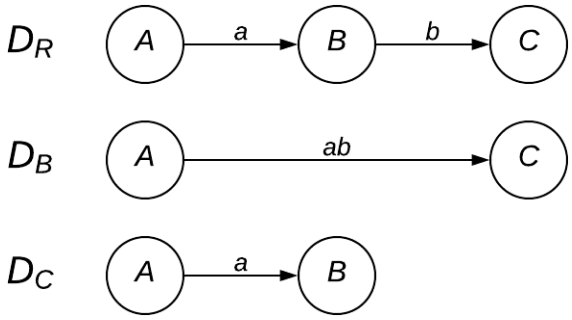

Figure 2.3: A network $\mathcal{D}_R$ with an abstraction that is complete ($\mathcal{D}_B$) and an abstraction that is not complete ($\mathcal{D}_C$).

information into the network that was not present in the original network. Thus, a minimal network realization must necessarily be non-extraneous, see [25].

2.4 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.

2.4.1 Acyclic Subgraphs of Digraphs

Definition 14 *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 15 *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 16 *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 [28]. 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 [2], [23] and [4].

Definition 17 *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 [10] and [15].

2.4.2 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 18 *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 $\mathcal{D}$.

2.4.3 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.*

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.*

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.*

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.

2.4.4 DAG Abstractions and Conditions on Completeness

We now return to the DAG abstraction defined in Section 2.4 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.*

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.*

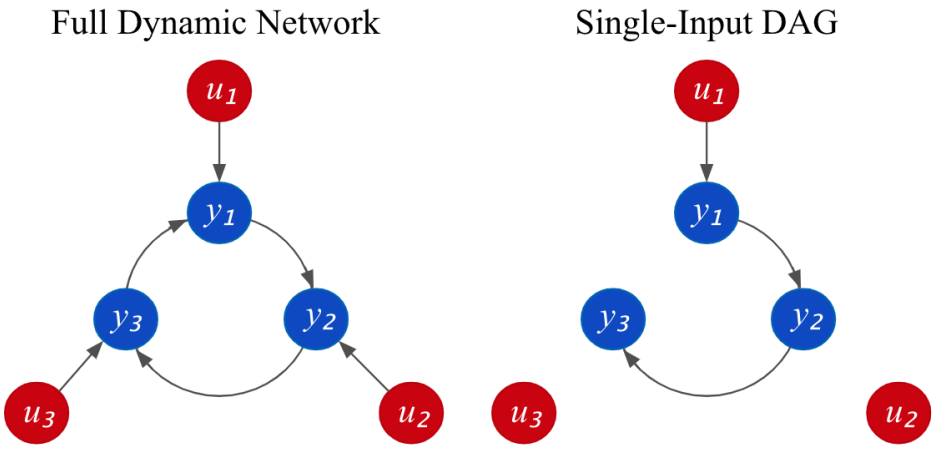

Figure 2.4: On the left is a dynamic network and on the right a single-input DAG abstraction. Note that the graphical union of all three such abstractions includes all edges in the original network.

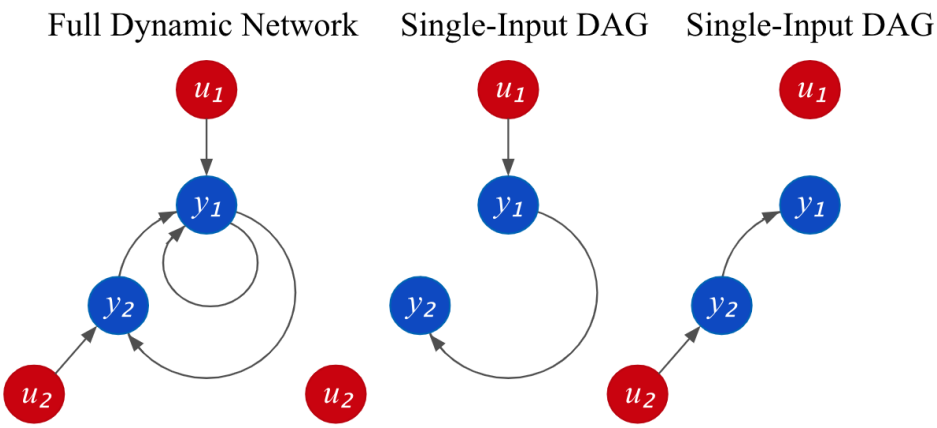

Figure 2.5: On the left is a dynamic network and on the right are its two single-input DAG abstractions. Note that their graphical union does not include all loop dynamics.

2.5 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 [25]).

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 2.4 illustrates this point for a ring structure. In others (see Figure 2.5) 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 [35], [36] and [37] 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 [5]).

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?

Chapter 3

Characterizing Network Controllability and Observability for Abstractions and Realizations of Dynamic Networks

Accepted as a Special Session Paper to be presented in the 2020 IFAC World Congress in Berlin

One method for managing the complexity of a dynamic network is to abstract some of this complexity away by using a simpler, yet behaviorally equivalent, mathematical model. A theory for such abstractions is currently under development (cf. [29] as well as [68]). While recent work has considered concepts of controllability and observability for networked dynamic systems (cf. [70] and [33]), this paper analyzes these concepts for *abstractions* of dynamic networks. In particular, we present the notion of a *complete* abstraction and an *extraneous* realization of a dynamic network and show that these concepts characterize the controllability and observability properties of a class of abstractions of dynamic networks.

3.1 Introduction

Dynamic networks are a useful representation of dynamic systems. They not only reflect the behavior of the system, but also capture a notion of structure in the way the system computes its behavior. In particular, nodes in a dynamic network represent variables, or signals, while edges between nodes describe dynamic relationships between signals. In linear time-invariant (LTI) systems, these edges are labeled with SISO transfer functions (or their time-domain equivalent, a convolution kernal) characterizing the dependency of one variable

on another. When considering directed edges, we align the direction of an edge with the causal dependency among variables.

A number of interesting systems questions can be posed in terms of dynamic networks. For example, work on characterizing informativity conditions for discovering network structure and edge dynamics under various conditions has been analyzed in [20], [1], [6], [7] and [45]. Likewise, a rich body of literature is emerging on the identification of part or all of a dynamic network, see [35], [38], [49], [61] and [57]. Distributed control problems have been explored where the dynamic network of the controller is constrained to have a particular network structure (see [50]), and the security of cyber-physical-human systems has been explored in terms of the underlying dynamic network characterizing the system in [51], [8] and [22].

Another set of interesting questions deals with the relationship between the graph structure of a dynamic network and its behavior. Some of these questions include structural controllability and reachability (first introduced in [31]), which extend the classic notions of controllability and observability of state-space models. Structural controllability analysis may be applied to all the fields where controllability is relevant and is especially of interest as exact knowledge of the model parameters are not required [70].

However, the complexity and scale of modern-day networks means that such analysis can be prohibitively expensive. Several strategies for managing this complexity arise from considering *abstractions* (behaviorally equivalent, though structurally simplified models) of dynamic networks. A rich theory of such abstractions is currently under development, see [26], [68] and [29].

This paper highlights key similarities and differences between dynamic networks and *networked dynamic systems*, a prominent model class for considering structured complex networks. It then contributes definitions and characterizations of network controllability and observability of abstractions of linear dynamic networks. These results highlight the advantages of considering network abstractions when modeling, as well as conditions (we namely

consider an abstraction condition called *completeness*) on which doing so fundamentally changes the interpretation of the abstracted (simplified) model.

3.2 Background

The terms *networked dynamic systems* and *dynamic networks* are used in the literature to mean very different things. Likewise, different notions of controllability and observability have also been developed for networks. This section briefly surveys these concepts and establishes the focus of the results presented in this paper.

3.2.1 Networked Dynamic Systems vs. Dynamic Networks

Networked dynamic systems are interconnections of subsystems that result in a larger, more complicated system ([6, 60]). Often such models are used to describe multi-vehicle systems ([42]), power systems ([24]), or other systems interconnected by communication networks ([43]). The typical linear model for such a system would consider linear state equations for each agent,

$$\begin{aligned} \dot{x}_i(t) &= A_i x_i(t) + B_i u_i(t) \\ y_i(t) &= C_i x_i(t) \end{aligned}$$

where $i \in 1, 2, ..., N$ indicates the agent index (in a system with N agents); $x_i(t) \in \mathbb{R}^{n_i}$ is the agent's internal state vector; the agent's input vector, $u_i(t) \in \mathbb{R}^{m_i}$, receives input signals from either external sources or neighbors in an interconnection (possibly directed) graph, $\mathcal{G}$, characterizing the structure of the entire system; the agent's output vector, $y_i(t) \in \mathbb{R}^{p_i}$, sends information to p_i neighbors defined by the same interconnection graph, $\mathcal{G}$; and t may be either continuous or discrete (in the discrete case, $\dot{x}(t)$ should be interpreted as $x(t+1)$ instead of $\frac{dx}{dt}$). Note that stacking the equations for each agent leads to a linear state space model characterizing the entire system, although the structure of $\mathcal{G}$ might get lost in such a representation. In such networks, *nodes* of $\mathcal{G}$ represent *systems* while *edges* carry *signals*

transmitting information from one subsystem to another. We often call $\mathcal{G}$ the *subsystem structure* of the network, since it represents the interconnection structure of subsystems.

Dynamic networks, on the other hand, are representations of dynamic systems that also appeal to a graph, $\mathcal{G}$, to characterize their structure, and they have been used to model biochemical reaction networks ([73]), financial networks ([39]), the attack surface for the security of cyber-physical-human systems ([22] and [8]), and to address different network reconstruction and identification problems, see [63], [56], [40] and [20]. Nevertheless, in these graphs, in contrast to networked dynamic systems, *nodes* represent *signals* while *edges* represent *systems*. Such systems are characterized by equations of the form

$$y = Wy + Vu$$

where y is a vector of length p and represents manifest measurements from the system and u is a vector of length m representing stochastic or deterministic inputs to the system. Note that, similar to networked dynamic systems, these equations may be considered in either the time or frequency domain and over discrete or continuous time. Thus, for example, to model a continuous time system in the frequency domain, entries of y, u, W and V would be real rational functions of the Laplace variable, $s \in \mathbb{C}$, while modeling a discrete time system in the time domain suggests that entries of these variables are functions of (discrete) time, $t \in \mathbb{Z}$, and multiplication would become the convolution operation. The graph structure, in this case, however, is revealed by the adjacency structure of W and V, and entries of these matrices reveal the dynamics associated with the system, called a *module* (see [62]), represented by each edge of the graph. Because we restrict our attention to *causal modules* and *well-posed representations* (see [69]), the resulting graph $\mathcal{G}$ can be interpreted as revealing the *causal dependencies* among the signals y and u; thus we call $\mathcal{G}$ the *signal structure* of the system.

Every causal, linear time-invariant system has both a subsystem structure and a signal structure, but in general these structures are different, see [71]. One example of this difference is due to the possibility of a *shared hidden state* between modules, while subsystems do not share states. The existence of shared hidden state implies that distinct modules on the dynamic network may not be separated in the dynamic system from which the dynamic network was computed; this may complicate the interpretation of a dynamical network as a physical system.

3.2.2 Structural Controllability and Observability

Consider an LTI state-space model:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx. \end{aligned} \tag{3.1}$$

We call this model (A, B, C) for convenience.

We restate the classic definition of structural controllability given by [31] via the notion of *structural zeros*[1] explained in [26].

To consider the structural controllability of (A, B, C) from a graph-theoretic standpoint we represent the system with a weighted digraph. We do so by considering the states, x, and the inputs, u, to be graph nodes and then characterize the system graph as the graph composition of a bipartite digraph with weighted adjacency matrix W_u and a second digraph with weighted adjacency matrix W_x, where

$$W_u = \begin{bmatrix} 0 & B \\ 0 & 0 \end{bmatrix} \text{ and } W_x = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}.$$

[1] Loosely speaking, structural zeros correspond to the absence of a connection between variables as opposed to the existence of a potentially zero-strength connection.

This gives us a new graph to consider, whose weighted adjacency matrix is of the form:

$$W_{x+u} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}.$$

The first n nodes in this resulting digraph are the state-variables, x, the remaining m nodes are the inputs, u. Each zero entry in A or B can either correspond to an edge with a zero weight or to a non-existent edge. One can think of each non-existent edge on this graph as a structural zero.

Definition 19 (Structurally Controllable) *The model in Equation 3.1 is structurally controllable if there exists a controllable system* $(\bar{A}, \bar{B}, \bar{C})$ *which has the same structural zeros as the system* (A, B, C).

Graph structure conditions on when a system is structurally controllable (specifically path connectivity from input nodes and the absence of graph dilations) are then given for the graph structure in [31].

For these LTI systems, the same graph results on structural observabillity may be defined on a sort of graphic dual of W_{x+u}. By graphic dual we mean the $n+p$ node graph (where nodes are the states and outputs) with the weighted adjacency matrix of the form:

$$W_{x+y} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}.$$

Structural observability can then be posed as a structural controllability problem on the transpose digraph given by

$$W_{x+y}^T = \begin{bmatrix} A^T & C^T \\ 0 & 0 \end{bmatrix}.$$

Thus we extend our definition of structural controllability to cover structural observability.

Definition 20 (Structurally Observable) *The model in Equation 3.1 is structurally observable if there exists a controllable system* $(\bar{A}^T, \bar{C}^T, \bar{B}^T)$ *which has the same structural zeros as the system* (A^T, C^T, B^T).

3.2.3 The Computational Dynamic Network Function

The *computational dynamic network function* for LTI systems is a model derived from Equation 3.1, as follows. By taking the Laplace transform (or Z transform depending on if time is continuous or discrete) of Equation 3.1 and assuming zero initial conditions we get:

$$\begin{aligned} sX &= AX + BU \\ Y &= CX \end{aligned}$$

Now, dividing by s yields: (3.2)

$$\begin{aligned} X &= \frac{1}{s}AX + \frac{1}{s}BU \\ Y &= CX. \end{aligned}$$

The equation tuple $(X = \frac{1}{s}AX + \frac{1}{s}BU, Y = CX)$ is the computational dynamic network function (DNF) of (A, B, C).

We now define structural controllability and observability on the computational DNF. Since there is a one-to-one correspondence between the LTI system (A, B, C) and its computational DNF, it inherits structural controllability and observability from the LTI system it corresponds to.

Definition 21 *The computational DNF,* $(X = \frac{1}{s}AX + \frac{1}{s}BU, Y = CX)$*, is* structurally controllable *if* (A, B, I) *is structurally controllable, and it is* structurally observable *if* (A^T, C^T, I) *is structurally controllable.*

The computational DNF may be interpreted as a digraph with the same structure as (A, B, C), but with the weights in A and B scaled by $\frac{1}{s}$, a first-order dynamic system.

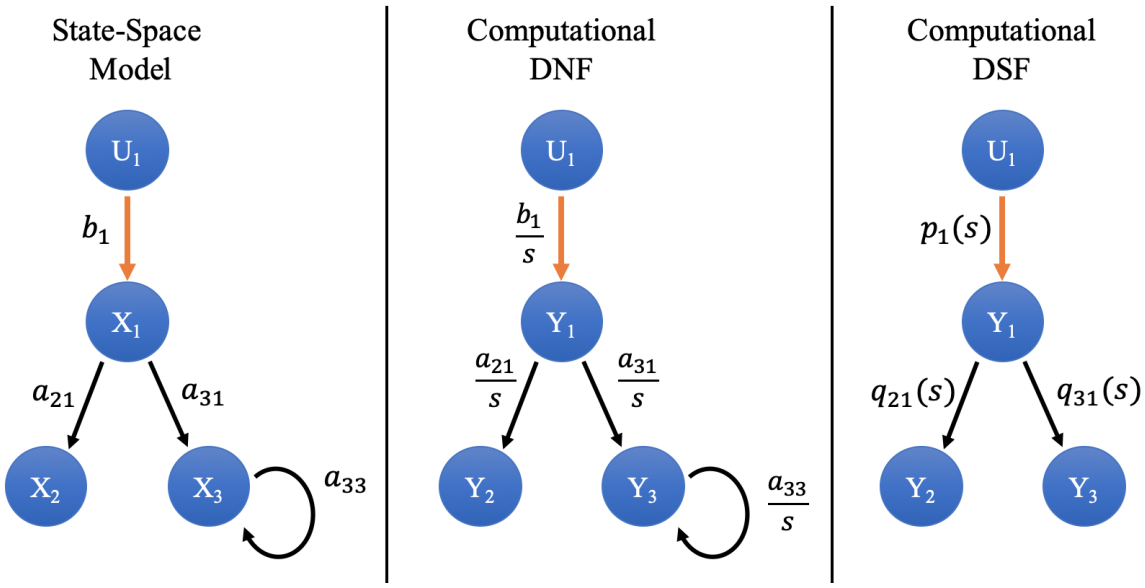

Figure 3.1: Example of graphical representations of the state-space, computational DNF and computational DSF models of an LTI system. Note that there is a one-to-one correspondence between a state-space model with $C = I$ and its computational DNF, thus the two perfectly share network controllability characteristics. Note in this case that the state-space (and therefore computational DNF) model has no dilations and has path connectedness from the input to all states, so all three models (including the computational DSF) are structurally controllable.

3.2.4 General LTI Dynamic Networks

General LTI dynamic networks may be interpreted as arbitrary digraphs whose edge weights are themselves SISO LTI systems (typically represented by rational polynomials over $\mathbb{C}$). One such example is the computational dynamical structure function (DSF), see [20].

Example 3 *Consider the state-space system:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & 0 & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \\ 0 \end{bmatrix} u_1$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

This system's computational DSF is the tuple $(Q(s), P(s))$*, where* $Y(s) = Q(s)Y(s)+P(s)U(s)$*. It is computed by taking an immersion (see Definition 22) with* $C_A = I$*. Specifically:*

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \\ Y_3(s) \end{bmatrix} = \frac{1}{s} \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ \frac{a_{31}s}{s-a_{33}} & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1(s) \\ Y_2(s) \\ Y_3(s) \end{bmatrix} + \frac{1}{s} \begin{bmatrix} b_1 \\ 0 \\ 0 \end{bmatrix} U_1(s).$$

Note that the self-loop in the computational DNF has now been incorporated into the relationship between Y_1 *and* Y_3*. This means that the network is structurally controllable (like the state-space model and computational DNF it was computed from), although the graph contains a dilation.*

Figure 3.1 demonstrates the digraphs associated with a simple LTI system, its computational DNF and its computational DSF.

3.3 Network Abstractions and Realizations

A theory of abstraction has been developed for dynamic networks. This theory results in a spectrum of representations linking input-output or behavioral descriptions to state space models of the same underlying system, see [67] and [64]. Moreover, while previous work has considered the controllability and observability of networked dynamical systems (such as [70], [33] and [11]), this work considers how these concepts relate to dynamic networks, in general, and their corresponding spectrum of abstractions in particular.

An abstraction of a dynamic network is a behaviorally equivalent model with fewer structural parameters. While there are many forms of abstractions, we will concern ourselves

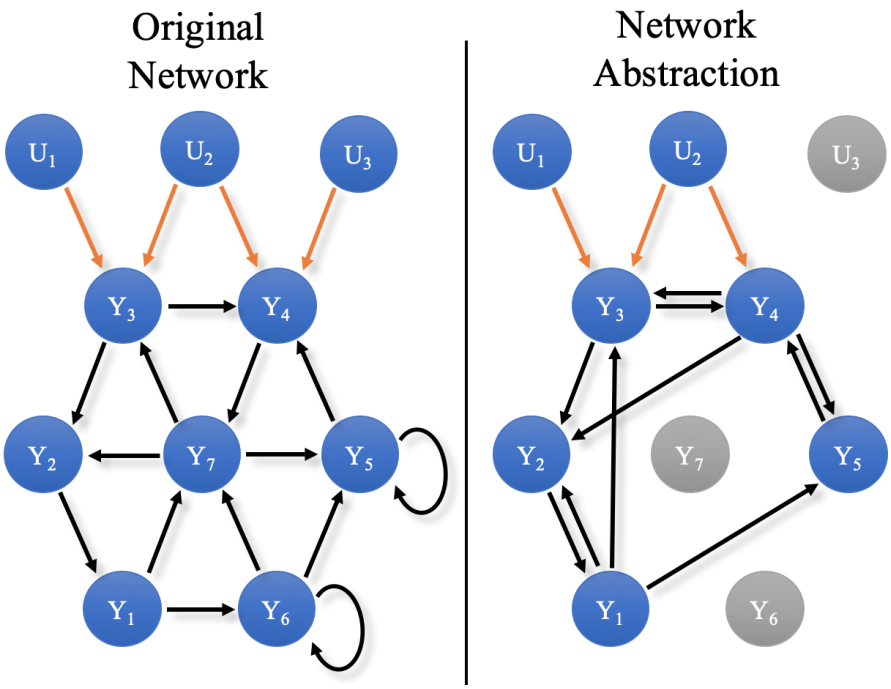

Figure 3.2: A visual demonstration of the concept of a network abstraction. We apply an immersion and an input immersion. By ignoring nodes U_3, Y_6, and Y_7, we reduce the number of parameters needed to learn the simplified network model from data. Note that the edge weights (which are dynamic systems) between the two models often will not be the same.

with immersions in this work. Formally we define an immersion as follows (note the relationship of this definition to those given by [68] and [64]).

Definition 22 (Immersion and Realization) *An* immersion *of an LTI network model, $(W_R(s), V_R(s))$, with input nodes $u_R(s)$, manifest nodes $x_R(s)$ and transfer function $G_R(s)$, is a second LTI network model $(W_A(s), V_A(s))$, with input nodes $u_A(s)$, manifest nodes $x_A(s)$ and transfer function $G_A(s)$ where:*

1. *W_A is hollow (has zeros on the diagonal),*
2. *$x_A(s) = C_A x_R(s)$ and*
3. *$G_A(s) = C_A G_R(s)$,*

where C_A is a set of rows from a permutation matrix of appropriate size.

We call $(W_R(s), V_R(s))$ is the realization *of $(W_A(s), V_A(s))$.*

Additionally, to simplify one's network models one may abstract away a network input (many things could motivate the desire to remove or ignore a network actuator). To do so we define a second form of abstraction.

Definition 23 (Input Immersion and Realization) *An* input immersion *of an LTI network model, $(W_R(s), V_R(s))$, with input nodes $u_R(s)$, manifest nodes $x_R(s)$ and transfer function $G_R(s)$, is a second LTI network model $(W_A(s), V_A(s))$, with input nodes $u_A(s)$, manifest nodes $x_A(s)$ and transfer function $G_A(s)$ where:*

1. *W_R is hollow (has zeros on the diagonal),*
2. *$u_A(s) = C_A u_R(s)$ and*
3. *$G_A(s) = G_R(s) C_A^T$,*

where C_A is a set of rows from a permutation matrix of appropriate size.

We call $(W_R(s), V_R(s))$ is the realization *of $(W_A(s), V_A(s))$.*

3.3.1 Structural Controllability and Observability of Dynamic Networks and their Immersions

The graphical results characterizing structural controllability and observabillity (originating with the introduction of structural control in [31]) do not extend to a reasonable definition of structural control on general dynamic networks (a parallel argument for the failure of these results to extend is given for networked dynamic systems in [11]).

Indeed, when we define structural control to be the existence of edge dynamics that permit a structurally controllable realization, we find that path connectedness is the only notion required for structural control. However, LTI systems that are not structurally controllable may be represented (as abstractions) by Dynamic Networks that are path connected.

To see that past graph results on structural controllability do not directly carry over to LTI dynamic networks, examine the dynamic network given by the DSF in Example 3 and

shown in Figure 3.1. When one examines the graph topology of the DSF there appears to be a graph dilation, which implies that structural controllability of the network is impossible, a conclusion shown in [31]. However, when one examines the computational DNF, or the state-space model which can be realized from the same network, there is no dilation (due to the existence of a self-loop on X_3), and the system is therefore structurally controllable.

Alternatively, we can restrict our definition to constrain the choice of edge dynamics with the same graph structure and then consider if a realization of the current dynamics is itself structurally controllable. But, even this working definition has its issues. It still allows a network that was derived from a structurally uncontrollable state-space model to be realized as structurally controllable. See Example 4 and Figure 3.3. Further restriction on which realizations we may consider for structural controllability is necessary.

Example 4 *(Uncontrollable System, Controllable Realizations) Consider the structurally uncontrollable state-space model*

$$(A, b) = \left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right).$$

Taking the trivial immersion of the system results in,

$$(Q_A(s), P_A(s)) = \left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \frac{1}{s} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right).$$

The transfer function of this immersion is: $G(s) = P(s)$. However, we may realize this system as an abstraction of the system

$$(A, B) = \left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \right),$$

which is clearly controllable (B is full row rank).

Note that the transfer function of this controllable realization,

$$G_{CR}(s) = \frac{1}{s}\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix},$$

has a higher Smith McMillan degree than that of the original model with transfer function $G_A(s) = P_A(s)$. *This difference in Smith McMillan degree motivates the below definition of structural controllability of a dynamic network.*

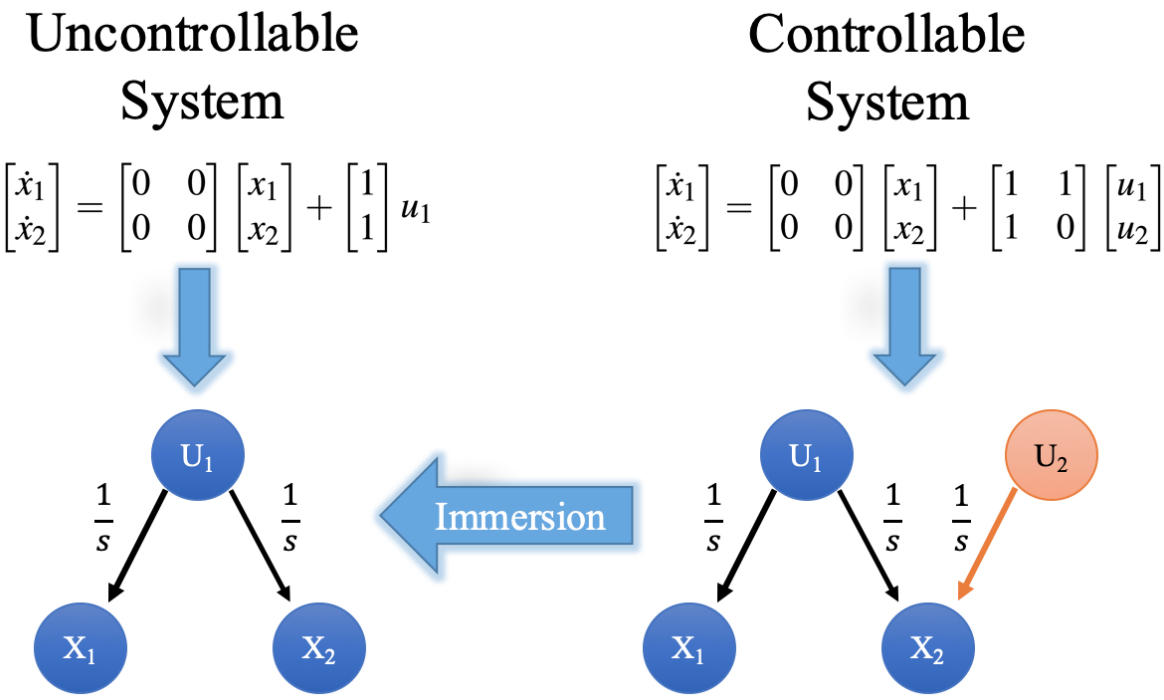

Figure 3.3: In this example we see that one network can be derived from a structurally uncontrollable state-space model, but also be the abstraction of a network that was derived from a structurally controllable state-space model (hence the same network is also derivable from a structurally controllable state-space model). We thereby justify holding the Smith McMillan Degree constant when considering state-space realizations of our abstractions.

A definition of structural controllability for dynamic networks needs avoid situations like that demonstrated in Example 4. To avoid this issue we propose restricting the set of admissible realizations to those which share the same Smith McMillan degree. This will allow

the realization of a class of systems which captures the same dynamic information as our network in a clear, tractable way.

Definition 24 (Structural Controllability) *An LTI dynamic network,* $(W_A(s), V_A(s))$*, is structurally controllable if there exists a structurally controllable computational dynamic network function in the set of all possible realizations of* $(W_R(s), V_R(s))$ *which are of the same Smith-McMillan degree of* $(W_A(s), V_A(s))$*.*

Under any of the definitions we considered above (including the one we chose) we cannot necessarily trust all the basic graph conditions (see Figure 3.1) used to characterize structural controllability on general LTI dynamic networks because the model we reference may be derived from an equivalent model with a more complex graph structure. However, we can ask the question of when properties like structural controllability carry over from realization to immersion. As a prelude to some results that answer this question we introduce the notion of a *complete immersion* and an *extraneous realization.*

3.4 Complete Immersions

We argue that not all network immersions are equal in their effect on the resulting model. There are some immersions which require no more than the removal of rows and columns from $W_R(s)$ and rows from $V_R(s)$, and there are some input immersions that require no more than removal of some columns of $V_R(s)$. We will refer to these as *incomplete immersions.* Incomplete immersions dramatically obfuscate the structural controllability and observability of dynamic networks by removing both structural and dynamic information from the resulting model.

Complete immersions, on the other hand, preserve underlying dynamics and, while they may appear to create dilations in the network, loop dynamics are encoded on to the edges and path connectivity is never disrupted.

Definition 25 (Complete Immersion) *An (input) immersion, $(W_A(s), V_A(s))$ is complete with respect to a subset of nodes in $w_R(s)$, $w_R^*(s)$, and a subset of nodes in $u_R(s)$, $u_R^*(s)$, if $w_A(s)$ contains all the nodes $w_R^*(s)$ and $u_A(s)$ contains all the nodes of $u_R^*(s)$.*

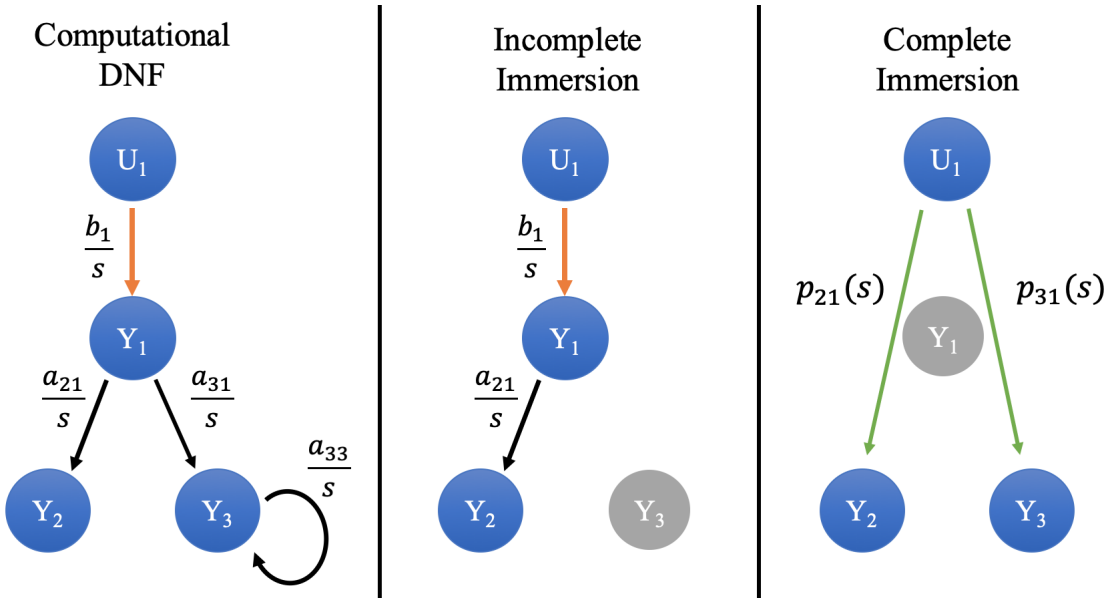

Figure 3.4: An example of an incomplete and a complete immersion with respect to the set of sink nodes of the same computational DNF. When performing an immersion we throw out the abstracted nodes (hence the abstracted node is represented in gray). Note that the complete immersion required the creation of new edges. The dynamics corresponding to the edges leading in to and out of Y_1 are placed on the edges $p_{21}(s)$ and $p_{31}(s)$ to maintain equivalent transfer functions from the realization to the immersion. The preservation of these dynamics on these new edges keeps key information in the simplified model.

Example 5 (Complete and Incomplete Immersion) *Consider the state-space model given in Example 3. We perform two distinct immersions on that state-space model. The first with*

$$C_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ and the second with } C_A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The first is an incomplete immersion with respect to the set of sink nodes. It results in the DSF $(Q_I(s), P_I(s))$*, where*

$$Q_I(s) = \frac{1}{s}\begin{bmatrix} 0 & 0 \\ a_{21} & 0 \end{bmatrix}, P_I(s) = \frac{1}{s}\begin{bmatrix} b_1 \\ 0 \end{bmatrix}.$$

Note that Q_1 *and* P_1 *are each submatricies of the* Q *and* P *associated with the computational DSF of this state-space system. This implies that no information from the edges into and from* Y_3 *have been included in this immersion.*

The second is a complete immersion with respect to the set of sink nodes. It results in the DSF $(Q_C(s), P_C(s))$*, where*

$$Q_C(s) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, P_C(s) = \frac{1}{s^2}\begin{bmatrix} b_1 a_{21} \\ \frac{b_1 a_{31}}{s - a_{33}} \end{bmatrix}.$$

In this model the entries in P_C *are distinct from those of* P *from the computational DSF. The edge weights in* P_C *have been modified to encode the dynamics of the entire model.*

By definition all input immersions are incomplete with respect to source nodes.

3.5 Results

Below we present preliminary results on the structural controllability of immersions and realizations of LTI dynamic networks.

Result 1 (Edge Weights in Incomplete Immersions) *Any single-node immersion,* $(W_A(s), V_A(s))$*, of an LTI dynamic network,* $(W_R(s), V_R(s))$*, that is incomplete with respect to sink nodes where* $C_A w_R(s) = w_A(s)$ *may always satisfy* $W_A(s) = C_A W_R(s) C_A^T$*.*

Without loss of generality assume that we abstract away the last node and it is a sink node. So $W_R(s)$ has the block structure: $W_R(s) = \begin{bmatrix} W_{11} & 0 \\ W_{12} & w_{22} \end{bmatrix}$, with $V_R(s) = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$,

where W_{12} and V_2 are row vectors. So the transfer function of the realization is: $G_R(s) = \begin{bmatrix} (I - W_{11})^{-1}V_1 \\ \frac{1}{1-w_{22}}(W_{12}(I - W_{11})^{-1}V_1 + V_2) \end{bmatrix}$, while $G_A(s) = (I - W_{11})^{-1}V_1$. This may be achieved by an abstraction of the form: $W_A(s) = W_1 1(s)$ and $V_A(s) = V_1(s)$.

Result 2 (Edge Weights in Incomplete Input Immersions) *Any single-node input immersion, $(W_A(s), V_A(s))$, of an LTI dynamic network, $(W_R(s), V_R(s))$, where $C_A u_R(s) = u_A(s)$ may always satisfy $V_A(s) = V_R(s)C_A^T$.*

Without loss of generality assume that we abstract away the last input node. So $V_R(s)$ has the block structure: $V_R(s) = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$, where V_2 is a column vector. So the transfer function of the realization is: $G_R(s) = (I - W_R)^{-1}\begin{bmatrix} V_1 & V_2 \end{bmatrix}$, while $G_A(s) = (I - W_R)^{-1}V_1$. This may be achieved by an abstraction of the form: $W_A(s) = W_R(s)$ and $V_A(s) = V_1(s)$.

Note that results 1 and 2 imply that in these cases the Smith McMillan Degree of the network necessarily decreases as poles in the transfer function associated with the abstracted edges will not be used to compute any other input-output relationship in the abstraction.

Result 3 (Complete Smith McMillan Degree) *The Smith McMillan degree of a single-node immersion $(W_A(s), V_A(s))$, of a weakly connected LTI dynamic network, $(W_R(s), V_R(s))$ equals that of $(W_R(s), V_R(s))$ if and only if it is complete with respect to its source and sink nodes.*

We prove the first direction by the contrapositive. Assume that the immersion, $(W_A(s), V_A(s))$, is not complete with respect to the set of source and sink nodes. Without loss of generality assume that we abstract away the last input node. So $W_R(s)$ has the block structure: $W_R(s) = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & w_{22} \end{bmatrix}$ and $V_R(s) = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$, where W_{12} and V_2 is a column vector and W_{21} is a row vector. Then, we have two cases. In the first case we have abstracted away at least one source node, so $W_{21} = 0$. In the second case we have abstracted away at least one sink

node, so $W_{12} = 0$. In both cases, results 1 and 2 imply that the transfer function of the abstraction is of a lower Smith McMillan degree than the original network.

Now assume that $(W_A(s), V_A(s))$ is a complete immersion with respect to source and sink nodes of $(W_R(s), V_R(s))$. Then, to preserve transfer function equivalence, all the dynamics of the abstracted edges pass over to the transfer function of the abstraction and so the Smith McMillan degree is the same.

Result 4 (Structural Control of Complete Immersions) *Complete immersions with respect to source and sink nodes of a structurally controllable LTI dynamic network are structurally controllable.*

This is true because the Smith-McMillan degree of the immersion is that of the realization and the immersion and the immersion dynamics are consistent (the transfer function entries are equal), so the realization may be constructed from the immersion. Since that realization was structurally controllable so is the immersion.

Analogous results for structural observability may be acquired by applying immersions to dynamic networks generated from the dual network.

3.6 Conclusions

We have explored the difference between networked dynamic systems and dynamic networks. We have highlighted how dynamic networks have an established theory of abstraction which allows one to greatly reduce the topological complexity of their use and of learning them from data. However, the interpretation of the abstracted model (especially immersions of the original model) may have a significant difference in interpretation due to the existence of shared hidden state. Regardless, such abstractions preserve all of the predictive power of their topologically complex realizations when they retain the property of completeness. Specifically, we defined structural controllability of dynamic networks and noted how completeness is a necessary requirement for preserving structural controllability through the abstraction

process. This informed application of dynamic network abstraction theory identifies how graph-based model topology simplification methods influences the interpretation of models of networked dynamic systems.

References

[1] J. Adebayo, T. Southwick, V. Chetty, E. Yeung, Y. Yuan, J. Gonçalves, J. Grose, J. Prince, G. B. Stan, and S. Warnick. Dynamical structure function identifiability conditions enabling signal structure reconstruction. In *IEEE Conference on Decision and Control*, Maui, 2012.

[2] Bonnie Berger and Peter W Shor. Approximation alogorithms for the maximum acyclic subgraph problem. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 236–243. Society for Industrial and Applied Mathematics, 1990.

[3] Hendrik W Bode et al. Network analysis and feedback amplifier design. 1945.

[4] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. On the advantage over random for maximum acyclic subgraph. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 625–633. IEEE, 2007.

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

[6] V. Chetty and S. Warnick. Necessary and sufficient conditions for identifiability of interconnected subsystems. In *Conference on Decision and Control*, Melbourne, Australia, 2017.

[7] V. Chetty, D. Hayden, S. Warnick, and J. Gonçalves. Robust signal-structure reconstruction. In *Conference on Decision and Control*, Florence, Italy, 2013.

[8] V. Chetty, N. Woodbury, E. Vaziripour, and S. Warnick. Vulnerability analysis for distributed and coordinated destabilization attacks. In *Conference on Decision and Control*, Los Angeles, 2014.

[9] V. Chetty, J. Eliason, and S. Warnick. Passive reconstruction of non-target-specific discrete-time LTI systems. In *American Control Conference*, Boston, MA, 2016.

[10] Yoeng-Jin Chu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14: 1396–1400, 1965.

[11] Noah J Cowan, Erick J Chastain, Daril A Vilhena, James S Freudenberg, and Carl T Bergstrom. Nodal dynamics, not degree distributions, determine the structural controllability of complex networks. *PloS one*, 7(6):e38398, 2012.

[12] Arne Dankers, Paul MJ Van den Hof, Xavier Bombois, and Peter SC Heuberger. Identification of dynamic models in complex networks with prediction error methods: Predictor input selection. *IEEE Transactions on Automatic Control*, 61(4):937–952, 2016.

[13] Arne Dankers, Paul MJ Van den Hof, Donatello Materassi, and Harm HM Weerts. Conditions for handling confounding variables in dynamic networks. *IFAC-PapersOnLine*, 50(1):3983–3988, 2017.

[14] Arne G Dankers, Paul MJ Van den Hof, and Xavier Bombois. Direct and indirect continuous-time identification in dynamic networks. In *53rd IEEE Conference on Decision and Control*, pages 3334–3339. IEEE, 2014.

[15] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards, B*, 71:233–240, 1967.

[16] Jalal Etesami and Negar Kiyavash. Directed information graphs: A generalization of linear dynamical graphs. In *2014 American Control Conference*, pages 2563–2568. IEEE, 2014.

[17] Joseph A Gallian. A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, 16(6):1–219, 2009.

[18] Gerd Gigerenzer and Daniel G Goldstein. Reasoning the fast and frugal way: models of bounded rationality. *Psychological Review*, 103(4):650, 1996.

[19] Jorge Gonçalves and Sean Warnick. Necessary and sufficient conditions for dynamical structure reconstruction of LTI networks. *IEEE Transactions on Automatic Control*, 53 (7):1670–1674, 2008.

[20] J. Gonçalves and S. Warnick. Necessary and sufficient conditions for dynamical structure reconstruction of LTI networks. *IEEE Transactions on Automatic Control*, August 2008.

[21] J. Gonçalves, R. Howes, and S. Warnick. Dynamical structure functions for the reverse engineering of LTI networks. In *Conference on Decision and Control*, New Orleans, 2007.

[22] D. Grimsman, V. Chetty, N. Woodbury, E. Vaziripour, S. Roy, D. Zappala, and S. Warnick. A case study of a systematic attack design method for critical infrastructure cyber-physical systems. In *American Control Conference*, Boston, MA, 2016.

[23] Refael Hassin and Shlomi Rubinstein. Approximations for the maximum acyclic subgraph problem. *Inf. Process. Lett.*, 51(3):133–140, 1994.

[24] David J Hill and Guanrong Chen. Power systems as dynamic networks. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4–pp. IEEE, 2006.

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

[26] Woodbury Nathan Johnson, Charles A. and Sean Warnick. Graph theoretic foundations of cyclic and acyclic linear dynamic networks. 2020.

[27] Rudolf Emil Kalman. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 1(2):152–192, 1963.

[28] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[29] EMM Kivits and Paul MJ Van den Hof. On representations of linear dynamic networks. *IFAC-PapersOnLine*, 51(15):838–843, 2018.

[30] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[31] Ching-Tai Lin. Structural controllability. *IEEE Transactions on Automatic Control*, 19 (3):201–208, 1974.

[32] Jonas Linder and Martin Enqvist. Identification and prediction in dynamic networks with unobservable nodes. *IFAC-PapersOnLine*, 50(1):10574–10579, 2017.

[33] Yang-Yu Liu and Albert-László Barabási. Control principles of complex systems. *Reviews of Modern Physics*, 88(3):035006, 2016.

[34] Samuel J Mason. *On the logic of feedback*. PhD thesis, Massachusetts Institute of Technology, 1952.

[35] D. Materassi and G. Innocenti. Topological identification in networks of dynamical systems. *IEEE Transactions on Automatic Control*, 55(8):1860–1871, Aug 2010. ISSN 0018-9286. doi: 10.1109/TAC.2010.2042347.

[36] D. Materassi and M. V. Salapaka. On the problem of reconstructing an unknown topology via locality properties of the wiener filter. *IEEE Transactions on Automatic Control*, 57(7):1765–1777, July 2012. ISSN 0018-9286. doi: 10.1109/TAC.2012.2183170.

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

[38] Donatello Materassi. Reconstruction of topologies for acyclic networks of dynamical systems, 08 2011.

[39] Donatello Materassi and Giacomo Innocenti. Unveiling the connectivity structure of financial networks via high-frequency analysis. *Physica A: Statistical Mechanics and its Applications*, 388(18):3866–3878, 2009.

[40] Donatello Materassi and Murti V Salapaka. Identification of network components in presence of unobserved nodes. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 1563–1568. IEEE, 2015.

[41] Donatello Materassi and Murti V Salapaka. Signal selection for estimation and identification in networks of dynamic systems: a graphical model approach. *arXiv preprint arXiv:1905.12132*, 2019.

[42] Richard M Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 2007.

[43] Michael J Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211, 2010.

[44] Harry Nyquist. Regeneration theory. *Bell System Technical Journal*, 11(1):126–147, 1932.

[45] P. E. Paré, V. Chetty, and S. Warnick. On the necessity of full-state measurement for state-space network reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing: Symposium on Network Theory*, Austin, 2013.

[46] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Elsevier, 2014.

[47] Jordan B Peterson. *12 Rules for Life: An Antidote to Chaos*. Random House Canada, 2018.

[48] Charles C Pinter. *A Book of Abstract Algebra*. Courier Corporation, 2010.

[49] C. J. Quinn, N. Kiyavash, and T. P. Coleman. Directed information graphs. *IEEE Transactions on Information Theory*, 61(12):6887–6909, Dec 2015. ISSN 0018-9448. doi: 10.1109/TIT.2015.2478440.

[50] A. Rai and S. Warnick. A technique for designing stabilizing distributed controllers with arbitrary signal structure constraints. In *European Control Conference*, Zürich, Switzerland, 2013.

[51] A. Rai, D. Ward, S. Roy, and S. Warnick. Vulnerable links and secure architectures in the stabilization of networks of controlled dynamical systems. In *American Control Conference*, pages 1248–1253, Montréal, Canada, 2012.

[52] Claude Shannon. *Collected Papers*. IEEE Information Theory Society, 1993.

[53] Claude E Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12):713–723, 1938.

[54] Herbert A Simon. Rational choice and the structure of the environment. *Psychological Review*, 63(2):129, 1956.

[55] Venkat Ram Subramanian, Andrew Lamperski, and Murti V Salapaka. Network topology identification from corrupt data streams. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1695–1700. IEEE, 2017.

[56] Saurav Talukdar, Deepjyoti Deka, Donatello Materassi, and Murti Salapaka. Exact topology reconstruction of radial dynamical systems with applications to distribution system of the power grid. In *2017 American Control Conference (ACC)*, pages 813–818. IEEE, 2017.

[57] Paul M J Van den Hof, Arne Dankers, Peter S. C. Heuberger, and Xavier Bombois. Identification of dynamic models in complex networks with prediction error methods—basic methods for consistent module estimates, 2013. URL `http://www.sciencedirect.com/science/article/pii/S0005109813003592`. ID: 271426.

[58] Henk J van Waarde, Pietro Tesi, and M Kanat Camlibel. Identifiability of undirected dynamical networks: a graph-theoretic approach. *IEEE Control Systems Letters*, 2(4): 683–688, 2018.

[59] Henk J Van Waarde, Pietro Tesi, and M Kanat Camlibel. Necessary and sufficient topological conditions for identifiability of dynamical networks. *IEEE Transactions on Automatic Control*, 2019.

[60] Henk J van Waarde, Pietro Tesi, and M Kanat Camlibel. Topology identification of heterogeneous networks of linear systems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 5513–5518. IEEE, 2019.

[61] H. Weerts, P. M. J. Van den Hof, and A. Dankers. Identification of dynamic networks operating in the presence of algebraic loops. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4606–4611, Dec 2016. doi: 10.1109/CDC.2016.7798970.

[62] Harm HM Weerts, Arne G Dankers, and Paul MJ Van den Hof. Identifiability in dynamic network identification. *IFAC-PapersOnLine*, 48(28):1409–1414, 2015.

[63] Harm HM Weerts, Paul MJ Van den Hof, and Arne G Dankers. Identifiability of linear dynamic networks. *Automatica*, 89:247–258, 2018.

[64] Harm HM Weerts, Jonas Linder, Martin Enqvist, and Paul MJ Van den Hof. Abstractions of linear dynamic networks for input selection in local module identification. *arXiv preprint arXiv:1901.00348*, 2019.

[65] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. MIT press, 2019.

[66] Jan C Willems and Jan W Polderman. *Introduction to Mathematical Systems Theory: a Behavioral Approach*, volume 26. Springer Science & Business Media, 2013.

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

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

[69] N. Woodbury, A. Dankers, and S. Warnick. On the well-posedness of LTI networks. In *Conference on Decision and Control*, Melbourne, Australia, 2017.

[70] Linying Xiang, Fei Chen, Wei Ren, and Guanrong Chen. Advances in network controllability. *IEEE Circuits and Systems Magazine*, 19(2):8–32, 2019.

[71] E. Yeung, J. Gonçalves, H. Sandberg, and S. Warnick. Representing structure in linear interconnected dynamical systems. In *Conference on Desicion and Control*, Atlanta, 2010.

[72] Enoch Yeung, Jorge Gonçalves, Henrick Sandberg, and Sean Warnick. The meaning of structure in interconnected dynamic systems. *arXiv preprint arXiv:1108.2755*, 2011.

[73] Enoch Yeung, Jongmin Kim, Jorge Gonçalves, and Richard M Murray. Global network identification from reconstructed dynamical structure subnetworks: Applications to biochemical reaction networks. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 881–888. IEEE, 2015.