

ABSTRACT

Title of dissertation: **GENERALIZED SYNCHRONIZATION
TREES**

James R. Ferlez
Doctor of Philosophy, 2019

Dissertation directed by: Professor Steven I. Marcus
Department of Electrical and Computer Engineering
and
Professor W. Rance Cleaveland
Department of Computer Science

We propose a novel framework for modeling cyber-physical systems (CPSs) that we call Generalized Synchronization Trees (GSTs). GSTs provide a rich framework for modeling systems that contain both discrete and continuous behavior in any combination, as well as enabling novel methods for analyzing such systems. GSTs were inspired by Milner’s successful use of Synchronization Trees (STs) to model interconnected computing processes, and GSTs afford a means of applying those same perspectives to CPSs. In particular, STs – and thus GSTs – provide a very natural setting for studying bisimulation and composition. In this thesis, we study both matters from a number of different perspectives: different notions of bisimulation over GSTs are defined and their (unexpected) semantic differences are established; the relationship of GSTs to behavioral, state-based systems is demonstrated; a simple modal logic for GSTs is defined and its relationship to bisimulation

is established, in particular with respect to Hennessy-Milner classes of GSTs; and finally, bisimulation is demonstrated to be a congruence for several different composition operators.

First, we contribute a novel counterexample to the semantic equivalence of two common notions of bisimulation, as they are naturally adapted to GSTs; this is in contrast to the situation for discrete processes, where these two notions of bisimulation coincide. This example – and the definitions of bisimulation on which it is based – thus provides crucial guiding intuition for further study.

Second, we demonstrate how GSTs may be regarded as “unrollings” of conventional state-based behavioral systems, in direct analog to the way STs may be regarded as “unrollings” of ordinary labeled transitions systems. Crucially, conditions are established under which this unrolling procedure is shown to preserve a notion of bisimulation, as it does in the discrete case.

Third, we study the relationship between bisimulation for GSTs and a generalization of Hennessy-Milner Logic (HML) that we call Generalized Hennessy-Milner Logic (GHML). In particular, we establish a relationship between *maximal* Hennessy-Milner classes of discrete structures and *maximal* Hennessy-Milner classes of GSTs; a Hennessy-Milner class is a class of models for which modal equivalence implies bisimulation equivalence, and this direction of implication is seldom studied in the context of CPSs. Moreover, we translate Linear, Time-Invariant Systems – regarded as behavioral systems – into GSTs, and study the relationship between these GSTs and maximal Hennessy-Milner classes.

Finally, we study the congruence properties of bisimulation with respect to

several composition operators over GSTs. One such composition operator mirrors a synchronous composition operator defined over behavioral systems, so the relationship between GSTs and state-based systems leads to a natural congruence result. Another composition operator we consider is a novel generalization of the CSP parallel composition operator for discrete systems; for this operator, too, we establish that bisimulation is a congruence, although the operator itself has subtle, implicit restrictions that make this possible. We also show that *discrete* GSTs can capture the bisimulation semantics of HyPA, a hybrid process algebra; consequently, this is implicitly a congruence result over compositions obtained using HyPA terms for which bisimulation is a congruence.

GENERALIZED SYNCHRONIZATION TREES

by

James R. Ferlez

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:

Professor Steven I. Marcus, Chair/Advisor
Professor W. Rance Cleaveland, Co-Advisor
Professor P.S. Krishnaprasad
Professor John Baras
Professor Yasser Shoukry
Professor Michael C. Fu

© Copyright by
James R. Ferlez
2019

Acknowledgments

I would first like to thank my advisors Steve Marcus and Rance Cleaveland for their support, both financial and otherwise, throughout the duration of my PhD. I would also like to thank Professors P.S. Krishnaprasad, John Baras, Yasser Shoukry and Michael C. Fu for agreeing to serve on my committee. I am especially grateful to Michael Fu both for our interactions across many group meetings and for offering me a quiet spot in which to write this dissertation. I also owe an additional debt of gratitude to Professor Krishnaprasad for our many inspiring conversations over the years.

I would particularly like to thank Professor André Tits for always taking the time to hear me out on any matter, technical or not, and for his remarkable course on optimal control, which was undoubtedly the single most formative experience in my time here. I am also indebted to Professor C. Robert Warner, not only for teaching a less-than-ideal student three semesters of analysis but also for a thoughtful bit of encouragement that has sustained me in many difficult times. I would like to thank Professor Adrian Papamarcou for mentoring me as a co-teacher of ENEE 222 and for his many kind and thoughtful comments in the time since.

This dissertation would not have been possible without the support of my parents, my family and my friends. I am forever indebted to my brother Bryan, whose patience, thoughtfulness and sense of humor turned around many a lousy day for me; my brother Michael who never hesitated to lend his support many times during this process; and my brother Jonathan for his encouraging optimism and

for helping me get away on a vacation when I needed it most. I particularly want to thank my friends, Graham Kaland (né Alldredge), Eduardo Arvelo and Adam Tabeling for always keeping me honest (in matters technical and not) and each for showing me an uncommon humanity at times when my depression nearly got the better of me. I also want to thank Marcos Vasconcelos both for his friendship and for helping me through a health issue. I am also indebted to Andrew Beckwith not only for his friendship but also for a small, innocuous comment that nevertheless sustained my confidence as a writer through many trying times. Finally, I would like to thank all of my other friends and colleagues at UMD for their help and support over the years: Alborz Alavian, Marie Chau, Zamira Daw, Karamatou Yacoubou Djima, Peter Fontana, David Hartman, Meiyun He, Sam Huang, Yunlong Huang, Cheng Jie, Yunchuan Li, Kun Lin, Van Sy Mai, Waseem Malik, Ion Matei, Evripidis Paraskevas, L.A. Prashanth, Huashuai Qu, Bhaskar Ramasubramanian, Christoph Schulze, Guowei Sun, Yongqiang Wang, and David Ward (I apologize in advance to anyone I may have missed!).

Finally, I am immensely grateful to the staff in both ECE and ISR who have repeatedly gone out of their way to help me along in my graduate studies. I am particularly indebted to Melanie Prange, who has patiently helped me literally from the time I arrived at UMD until just yesterday. I also want to thank Beverly Dennis, Vivian Lu, Carla Scarbor and Robert McMullen, each of whom has made a special effort on my behalf; I am truly grateful.

Table of Contents

1	Introduction	1
1.1	Cyber-Physical Systems (CPSs)	1
1.2	Traditional Cyber-Physical System Models	2
1.2.1	Discrete, Labeled-Transition-like CPS Models	3
1.2.2	Continuous-Like Synchronous-Composition Models	5
1.2.3	Limitations	7
1.3	A New CPS Modeling Framework: Generalized Synchronization Trees	7
1.4	Outline of Contributions	8
1.4.1	System Equivalence for GSTs (Bisimulation)	9
1.4.2	GST Representations of Existing CPS Models	9
1.4.3	Modal Logic and Bisimulation for GSTs	10
1.4.4	Asynchronous Composition of GSTs (CSP-Parallel Composition)	11
2	Background	12
2.1	Process Algebra and Synchronization Trees: an Algebraic, Compositional Theory for Discrete Systems	12
2.2	Behavioral Description of Dynamical Systems	16
2.2.1	Behavioral Dynamical Systems: Definitions	16
2.2.2	State Maps and a Notion of Bisimulation for Behavioral Dynamical Systems	18
2.3	Modal Logic and (Maximal) VHHM Classes of KSSs	22
2.4	Hybrid Process Algebra (HyPA)	26
2.4.1	HyPA Syntax	26
2.4.2	HyPA Semantics	30
2.4.3	Bisimulation for HyPA	35
3	Generalized Synchronization Trees	37
3.1	Generalized Synchronization Trees	37
3.1.1	Generalizing Trees	37
3.1.2	Generalized Synchronization Trees	39
3.2	Bisimulation for GSTs	41

3.2.1	Notions of Bisimulation for GSTs	41
3.2.2	Relating Strong and Weak Simulations	44
4	Generalized Synchronization Trees and State Systems	48
4.1	Introduction	48
4.2	Behavioral Dynamical Systems as GSTs	49
4.3	Comparison of Bisimulation Relations for Behavioral Dynamical Systems	54
5	GSTs and Modal Logic	60
5.1	Introduction	60
5.2	Generalized Hennessy-Milner Logic (GHML)	64
5.2.1	GHML for GSTs: Syntax and Semantics	64
5.2.2	VHHM Classes of GSTs	67
5.3	Linear Time-Invariant Systems as GSTs	71
5.3.1	LTI Systems	72
5.3.2	LTI Systems as GSTs	73
5.3.2.1	Admissible Input Functions	73
5.3.2.2	Construction of the GSTs	79
5.4	LTI Systems and Order-Equivalence Semantics	82
5.4.1	Preliminaries	84
5.4.2	LTI Systems and Order-Equivalence Non-determinism: Purely Analytic Inputs and “Local” Non-determinism Structure	86
5.4.3	LTI Systems and Order-equivalence Non-determinism: Piecewise Analytic Inputs and “Global” Non-determinism Structure	106
5.5	Modal Logic and Order-equivalence Induced Nondeterminism	111
5.5.1	The Structure of Nondeterminism in LTI-derived Surrogate KSSs	112
5.5.2	Quasi-discreteness and VHHM Classes for Weak Dense, Transitive, Right-linear Structures	118
5.5.3	Implications for LTI Systems	134
6	Congruence Results	142
6.1	Synchronous Interconnection of GSTs Derived from Behavioral Systems	142
6.2	CSP-type Parallel Composition	145
6.2.1	Preliminaries	145
6.2.2	Congruence and Weak Bisimulation	146
6.2.3	Congruence and Strong Bisimulation	151
6.2.4	Anomalies in CSP-Parallel Composition for GSTs	154
6.3	A Process Algebra for Hybrid Systems: HyPA	159
7	Conclusions and Future Research	165
7.1	Conclusions	165
7.2	Future Research	167
	Bibliography	169

Chapter 1: Introduction

1.1 Cyber-Physical Systems (CPSs)

The ever-increasing complexity of real-world cyber-physical systems (CPSs) has revealed the inadequacy of traditional design and analysis techniques: neither design techniques from the physical domain (control theory) nor those from the cyber domain (computer science) have been proven to be sufficiently broad and scalable to address modern, real-world CPSs. This is a critical situation, as human safety and well-being increasingly depend on the bug-free design of CPSs. In fact, the failures of modern CPSs are increasingly expensive and newsworthy: for example, Toyota recently recalled nearly 700,000 Prius vehicles to fix a software bug that could cause the hybrid system to “shutdown while the vehicle is being driven, resulting in a loss of power.” [1]

The gulf between traditional design methodologies and modern CPSs is widening largely due to the desire for increased automation both across devices and within them. The number of control laws running simultaneously in real-world devices has been increasing steadily, and those controllers are increasingly tied together by inexpensive wired/wireless networks in addition to the dynamics of the device itself. The automotive industry is one notable example of this trend. The first automo-

bile to contain a microcontroller was released in 1977, and it employed a single microcontroller to adjust ignition timing [2]. By contrast, a modern luxury car typically contains in excess of 100 microcontrollers, most of which are interconnected by various networking features in addition to the dynamics of the car itself [2]. The proliferation and interconnectedness of such controllers has exposed the limitations of the traditional design approach where individual subsystems are designed independently and then subsequently integrated with limited analytical evaluation of the complete system. The complexity of modern CPSs is such that this design strategy increasingly leads to unanticipated emergent behavior, either through software bugs or unanticipated interactions between the control laws themselves.

The research in this dissertation is an effort to address deficiency. In particular, we take the point of view that the main impediment to safe, robust CPS design is a dearth of *expressive* yet *composable* CPS models. Thus, our approach is to propose a new CPS modeling framework that facilitates the accommodation of composability ideas from the analysis of discrete time systems, specifically process algebra and synchronization trees (STs). To appreciate the value of this approach, though, it is necessary to appreciate the current state of CPS modeling.

1.2 Traditional Cyber-Physical System Models

There are abundant CPS models in the literature, but virtually all of these models can be said to fit in at least one of the following categories: either the model and/or composition semantics is discrete and LTS-like, or the model treats

composition exclusively through synchronization. Examples of the former include HA [3] and HyPA [4], as well as [5–10]. Examples of the latter include [11, 12]. In this section, we summarize both types of models, and we conclude with a summary of the limitations thereof.

1.2.1 Discrete, Labeled-Transition-like CPS Models

Hybrid Automata (HA) [3] were among the earliest system models that were endowed with enough generality to significantly intersect the modern notion of a CPS. A HA specifies the time evolution of a fixed number of continuous (state) variables just like the aforementioned state equations; however, this evolution is not governed by a fixed system of state equations, but rather by a system of state equations that changes with the evolving (discrete) state of an automaton. This automaton can (nondeterministically) make a transition whenever prescribed conditions on the continuous state variables are satisfied (these conditions may depend on the state of the automaton); this *couples* the state of the automaton to the value of the continuous state variables. Thus, HA bridge the gap between differential equation models and discrete computational models, but the semantics of HA can be described by a particular LTS dubbed a *timed transition system* [3]. In a timed transition system, the set of locations is a subset of the (Cartesian) product of the automaton’s states and the state space of the continuous state variables: for a discrete state and a continuous state to comprise a location, the continuous states must satisfy all so-called invariant predicates associated with the discrete

state. Transitions from the automaton are lifted to transitions in the LTS, provided they emanate from a location that satisfies the transition condition described above. The semantics of the differential equations are captured by incorporating additional transitions. Such a transition is added between a pair of locations if and only if two conditions are met: first, both locations have a common discrete state, and second, the state equations in that discrete state admit a finite-duration solution whose initial and final states are consistent with the continuous states of the source and target locations, respectively. A solution that meets these conditions is called a *witness*, and each of these additional transitions is labeled with the duration of its witness.

Hybrid Process Algebra (HyPA) [4] can be thought of as a process algebra that captures and extends the semantics of Hybrid Automata. As such, HyPA is a compositional algebraic framework for modeling CPSs; the signature of HyPA reflects this by including flow clauses and atomic discrete events. As in traditional process algebra, semantic derivation rules describe how transitions transform one term in the HyPA algebra into another; in HyPA, as with hybrid automata, such transitions correspond to either a discrete action or a finite-duration continuous flow (in HyPA, such transitions are labeled by a witness to the transition). However, the semantics of HyPA, like the semantics of HA, must be sensitive to the valuation of the continuous model variables, so the aforementioned derivation rules deviate from traditional process algebra in that they transform a HyPA term/valuation pair into another HyPA term/valuation pair. This connects the semantic derivation rules of HyPA to the timed transition system in the context of hybrid automata: that is

the derivation rules in HyPA can be used to construct a labeled transition system where each state (location) is specified by both a HyPA term and a valuation of the model variables. Though hybrid automata and HyPA do not share strictly identical semantics, this modeling choice clarifies that a HyPA process, like a hybrid automaton, can be semantically captured by a labeled transition system.

1.2.2 Continuous-Like Synchronous-Composition Models

Historically, control theorists, physicists and others have regarded dynamical systems as synonymous with differential equations; often, those differential equations are a system of coupled first order differential equations, each of which specifies the first derivative of one variable. Such equations are known as state equations, and each variable that appears in differentiated form is called a state variable. For example, the dynamics of a falling body with mass m can be described in terms of the object's height above the ground, h , and its vertical velocity, v ; for convenience, we define the state variables $x_1 = h$ and $x_2 = v$. Using Newton's laws, we can then write a pair of differential equations in x_1 and x_2 : that is $\dot{x}_1(t) = x_2(t)$ and $\dot{x}_2(t) = g$, where the constant g is the acceleration due to gravity near the surface of the earth. Such state equations can also be endowed with free variables that represent external influences on the state: in the previous example, we might instead write $\dot{x}_2(t) = g + (1/m) \cdot u(t)$ to indicate that there is a time-dependent external force $u(t)$, which also acts on the falling body. In control theory, such a differential equation is then referred to as (a model of) the *plant*, and the control problem

is to design $u(t)$ in such a way as to beneficially alter the dynamical properties of the state, x_1 and x_2 . Such control problems also typically have constraints on how $u(t)$ may be designed, such as requiring that $u(t) = y(x_1(t), x_2(t))$ for some prescribed sensor output function y . Constraints of this sort also suggest a notion of composition, namely that of input to output connection (or synchronization): in the previous example, the states x_1 and x_2 are fed as inputs into the function y , and the result is connected to the control input u . It is in fact the case that all traditional methods of composition for differential equations are of this type.

Recently, Willems et. al. [13, 14] began advocating an alternate point of view where dynamical systems – including hybrid systems and CPSs – are defined completely by the time trajectories of the variables involved. From this point of view, a dynamical system is defined in terms of a signal space \mathbb{W} , which contains all possible values for the system variables. For an object in free fall (see above), we need to keep track of its height and velocity, so we might take $\mathbb{W} = \mathbb{R} \times \mathbb{R}$. A dynamical system is also ascribed a totally ordered time set \mathbb{T} over which the system variables evolve. The dynamics of the system are then described entirely in terms of the time trajectories of these variables or *behaviors*, that is functions from $\mathbb{T} \rightarrow \mathbb{W}$ (commonly denoted by $\mathbb{W}^{\mathbb{T}}$). Such models encapsulate the state equation models described previously, but they can also capture executions of hybrid automata, HyPA processes and other CPS-relevant hybrid frameworks. Unfortunately, composition for behavioral models is (to date) restricted to a generalized type of synchronization between variables, much like the input to output connections described in the context of traditional state-space models.

1.2.3 Limitations

On the one hand, continuous-like CPS models permit rich and expressive abstractions of CPS models that capture true continuous behavior and choice, but this traditionally comes with the restriction of *synchronous*-only composition operators. On the other hand, LTS-like CPS models allow for a broad range of compositional abstractions such as *asynchronous* composition, but only by forgoing the ability to faithfully represent arbitrary continuous-time behavior and choice. These complementary deficiencies are a serious bottleneck to compositional CPS modeling and hence, to robust, scalable CPS system design.

1.3 A New CPS Modeling Framework: Generalized Synchronization Trees

In this context, the goal of this dissertation is to advance the mathematical foundations of CPS modeling by addressing these specific limitations. To this end, we have developed a novel framework that offers two important features for modeling CPSs: enough *generality* to encompass many different CPS models and enough *compositionality* to model complex CPSs in a scalable way (i.e. as a composition of smaller, more manageable systems). In other words, the objective is to develop a modeling framework that transcends the deficits of current CPS models, as they are elucidated in [Section 1.2](#).

This framework, which we have called Generalized Synchronization Trees (GSTs)

[15], was inspired by Milner’s use of Synchronization Trees (STs) to model interconnected computing processes [16] (see [Section 2.1](#) for background on STs). As such, GSTs generalize the mathematical structure of their forebears to include a wide range of *non-discrete* systems – including many classes of CPSs – while also maintaining the structural compositionality properties that make STs such effective discrete-system models. Together, these features distinguish GSTs from the existing modeling frameworks in the literature, and ameliorate the limitations of those models as described in [Section 1.2](#). This unique *generality* of GSTs – both in ability to represent different CPSs and in ability to specify different types of composition – is in some sense a consequence of the novel choice to mirror the relatively simple tree-like structures that underpin STs. Moreover, with such a simple and flexible underpinning, GSTs are also particularly well suited to studying CPS models in an *abstract* way: it possible not only to represent many CPSs models as GSTs, but also to imagine GSTs *independent of any concrete underlying CPS model*. This makes GSTs further valuable as a tool to understand what is possible in very generic dynamical systems, and hence, extremely complicated CPSs.

1.4 Outline of Contributions

The core contribution of this dissertation is a new, ST-inspired CPS model in the form of GSTs: this material comprises the initial content of [Chapter 3](#). In addition, this dissertation also contains the following contributions, each of which is identified with its location in this document.

1.4.1 System Equivalence for GSTs (Bisimulation)

Milner [16] is credited with defining an important notion of system equivalence between discrete systems: this notion would eventually become known as bisimulation (see Section 2.1). However, in the context of (non-discrete) CPS models, two main notions of bisimulation developed simultaneously in the literature. It was a central contribution of [15] that these two notions are in fact different, and in general, irreconcilable. This material fills out the remainder of Chapter 3.

1.4.2 GST Representations of Existing CPS Models

For GSTs to be a useful tool in the study of *specific* CPS models, those CPS models must be representable as GSTs in some meaningful – and precise – way. In fact, the same sort of reconciliation is necessary between STs and (more) general discrete models; that is a conventional discrete model, a Labeled Transition System (LTS) for example, may be “unrolled” into a ST that is *bisimilar* to the original model (again, see Section 2.1). Emulating this connection, we have shown that several different CPS models can be “unrolled” into *bisimilar GSTs*: in [15], such a procedure was exhibited for HyPA processes (see Section 2.4), which have essentially Hybrid Automata semantics; and in [17], Behavioral Dynamical System models [13] were likewise shown to be “unrollable” into bisimilar GSTs. Together, these results establish the validity of GSTs as a tool for studying existing CPS models. This material is found predominantly in Chapter 4; the HyPA related material appears in Chapter 6, since it involves the inherent composition of a process algebra.

1.4.3 Modal Logic and Bisimulation for GSTs

Hennessy and Milner noticed early on that bisimulation equivalence for a special class of STs could be characterized in terms of a very simple modal logic, known as Hennessy-Milner logic (HML) [18] (see also [Section 2.3](#)); for this reason, HML has become an important tool in the formal description and verification of discrete systems. Of course new tools for the formal description and verification of *CPS models* are critical, so we developed a generalization of HML for GSTs – called Generalized Hennessy-Milner Logic (GHML) [19] – in order to study bisimulation for GSTs in terms of simple, HML-like modal formulas. Importantly, GHML can be shown to (partially) characterize *maximal* classes of GSTs with respect to bisimulation [19]: abstractly, this means that the failure of bisimulation between two such GSTs can be *explained* by a simple GHML “witness” formula. However, if one of those GSTs represents a *specification* and the other an *implementation*, then such a GHML “witness” provides a tractable explanation of *why* the implementation is buggy. Moreover, we have brand new work that describes the relationship between ubiquitous Linear Time-Invariant (LTI) system models and this idea of GST classes where bisimulation is captured by GHML: this is an important bridge between the *abstract* study of GHML as it pertains to bisimulation and the use of GHML formulas to get useful debugging information for real CPS models. Together, this material is the subject of [Chapter 5](#).

1.4.4 Asynchronous Composition of GSTs (CSP-Parallel Composition)

Also in [15] was a generalization of the well-known asynchronous parallel composition operator for discrete systems (also known as CSP-parallel composition). This parallel composition takes two discrete systems and composes them asynchronously to form a third (discrete) system. Importantly, the parallel composition of two particular (discrete) systems will be bisimilar to the result of composing any other pair of systems that are bisimilar to original two. We have proven a similar “substitutivity” result for our GST generalization of this composition operator, although not without caveats. This is work in preparation for submission, and it appears in [Chapter 6](#) alongside the compositional HyPA results described before.

Chapter 2: Background

2.1 Process Algebra and Synchronization Trees: an Algebraic, Compositional Theory for Discrete Systems

Process algebra was born in the early 1980's out of the then nascent interest in concurrent computing. Because of this heritage, process algebra was a compositional framework from its inception: two independent computing processes can interfere with each other when run concurrently, so it is crucial to model and understand possible *interactions* between two such concurrently running processes. The study of such interactions motivated the seminal work of process algebra: Milner's *Calculus of Communicating Systems* (CCS) [16].

The most important feature of Milner's work was its algebraic character: the interaction – or *composition* – of two computing processes was described solely by a third computing process. Specifically, Milner semantically formalized computing processes as a set; a particular notion of interaction between two processes is then described entirely by a function that maps two elements from this set into a third. In process algebra, composition operators really are mathematically analogous to other well-known algebraic operators such as addition and multiplication over the reals.

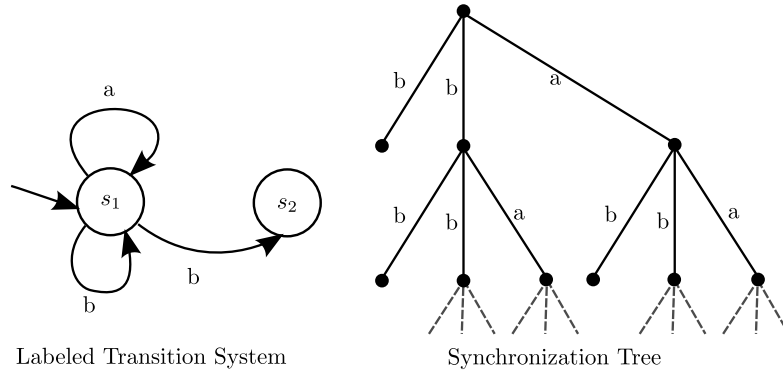


Figure 2.1: A labeled transition system and its corresponding synchronization tree

This feature is highly desirable in the study of composition, since such operators can be *nested*: that is the result of one composition may again be composed with another process using exactly the same operators, and the result of that composition may appear in yet another composition and so on.

Milner’s particular formalization of computing processes as synchronization trees (STs) was itself a crucial innovation of his theory. As a model for computing processes, STs can be understood conceptually as a special type of nondeterministic labeled transition system (LTS): a ST can be thought of as a loop-free LTS where every location (node) lies on a unique path from a special node called the root. The following quotes the formal, mathematical definition given by Milner on p. 16 of [16].

“A Synchronization Tree (ST) of sort L is a rooted, unordered, finitely branching tree each of whose arcs is labelled by a member of $L \cup \{\tau\}$.”¹

Consequently, a ST contains an explicit prefix hierarchy – formally, a partial

¹ Milner also introduces the notion of *rigid synchronization tree*, which limit edge labels to the set L . We elide this distinction, as we do not consider τ in this thesis.

order – on its nodes: each node in a ST represents a unique sequence of transitions, and the children (or successors) of a node represent *extensions* of this sequence. An ordinary LTS can be represented as a ST by simply tracing every one of its executions while uniquely relabeling each location every time it is visited. Conceptually, this amounts to “unrolling” a LTS as depicted in Fig. 2.1. The advantage of this structure is that every node in a ST can be viewed as the root of another ST; thus, the semantics of the underlying LTS can be encoded by rules that transform one ST into another. These rules are called *semantic derivation rules*, and they can also be used to describe the semantics of certain *composition operators* as transformations over STs.

The self-similar structure of STs also led Milner to define the concept of *bisimulation equivalence* between computational processes; this equivalence expresses precisely when one ST has the same behavior as another. Specifically, two STs are bisimilar if they *simulate* each other: given two STs A and B , B simulates A if every possible transition from the root of A can be matched by a transition from the root of B that has the same label, and after executing these transitions, the sub-ST of A is again simulated by the sub-ST of B . Bisimulation can be specified as in the following definition, adapted from [20]; recall that if R is a binary relation on a set $S \times T$ then R^{-1} , the inverse of R , is binary relation on $T \times S$ defined by $R^{-1} \triangleq \{\langle t, s \rangle \mid \langle s, t \rangle \in R\}$.

Definition 1 (Simulation and Bisimulation for Synchronization Trees). *Let L be a set of labels, and let ST_L be the set of STs whose labels come from L .*

1. Let $T, T' \in ST_L$ and $a \in L$. Then $T \xrightarrow{a} T'$ if there is an edge labeled by a from the root of T to the root of T' .
2. Relation $R \subseteq ST_L \times ST_L$ is a simulation if, whenever $\langle T_1, T_2 \rangle \in R$ and $T_1 \xrightarrow{a} T'_1$, then there exists T'_2 such that $T_2 \xrightarrow{a} T'_2$ and $\langle T'_1, T'_2 \rangle \in R$.
3. Relation $R \subseteq ST_L \times ST_L$ is a bisimulation if both R and R^{-1} are simulations.
4. Let $T_1, T_2 \in ST_L$. Then T_1 is simulated by T_2 (notation $T_1 \sqsubseteq T_2$) if there is a simulation relation R with $\langle T_1, T_2 \rangle \in R$.
5. Let $T_1, T_2 \in ST_L$. Then T_1 and T_2 are bisimulation equivalent, or bisimilar (notation $T_1 \sim T_2$), if there is a bisimulation R with $\langle T_1, T_2 \rangle \in R$.

The relation \sqsubseteq is often called *the simulation preorder*. It can be shown that the simulation preorder (bisimulation equivalence) itself is a simulation (bisimulation) relation, and indeed is the unique largest such relation. Note, however, that Milner's definition was only semi-formal: other authors [21–24] subsequently developed the underlying mathematics fully through coinductive constructions.

Conceptually, two bisimilar STs can certainly process the same sequences of labels, but crucially, both STs are able to process any such sequence and then *extend* it in the same way. Thus, bisimulation is an *interactive* equivalence that is stronger than language equivalence: this has ramifications specifically for system composition. For a number of interesting *composition* operators, this equivalence also describes precisely when one ST can be exchanged by another as an operand: Milner's synchronous composition and Hoare's *asynchronous* parallel composition

are notable examples.

2.2 Behavioral Description of Dynamical Systems

Since we will be working extensively with behavioral systems in [Chapter 4](#), we repeat a number of crucial definitions and results that will be necessary for regarding such systems as GSTs.

2.2.1 Behavioral Dynamical Systems: Definitions

Historically, control theorists, physicists and others have regarded dynamical systems as synonymous with differential equations. Recently, however, Willems et. al. (see e.g. [\[14\]](#) for a summary) began advocating an alternate point of view wherein dynamical systems are defined completely by the time trajectories of the variables involved. From this point of view, a dynamical system is defined in terms of a signal space \mathbb{W} , which contains all possible values for the system variables, and a totally ordered time set \mathbb{T} over which those system variables evolve. The dynamics of the system are then described entirely in terms of the time trajectories of these variables or *behaviors*, that is functions from $\mathbb{T} \rightarrow \mathbb{W}$ (commonly denoted by $\mathbb{W}^{\mathbb{T}}$).

Definition 2 (Behavioral Dynamical System [\[14\]](#)). A **dynamical system** Σ is a triple $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ where \mathbb{T} is a totally ordered time set and \mathbb{W} is the space in which the system variables take their values. The set $\mathcal{B} \subseteq \mathbb{W}^{\mathbb{T}}$ is the set of behaviors of the system and contains **all possible** time evolutions of the variables in the system.

We make several remarks about [Definition 2](#). First, systems defined in this way

need not distinguish between input and output variables. However, we're interested in bisimulation, which is an *interactive* notion, so we will often partition the signal space into input variables that take values in \mathbb{V} and internal/output variables that take values in \mathbb{D} ; hence, $\mathbb{W} = \mathbb{V} \times \mathbb{D}$. Finally, we make the further assumption that the time set \mathbb{T} can be embedded in a set which has an additive group structure (e.g. the integers or the reals), as in [25] where the definition of bisimulation depends on this fact in a strong way. In particular, that definition employs time-shifted behaviors: see [Definition 11](#).

Functions play a crucial role in the behavioral treatment of dynamical systems, so we introduce some convenient notation to facilitate manipulation of trajectories. With the exception of [Definition 4](#), all of this notation comes from [25].

Definition 3 (Truncation). *Given a behavior $w : \mathbb{T} \rightarrow \mathbb{W}$ and a time $t \in \mathbb{T}$, we define the truncation of w to t as that $w|_t \in \mathbb{W}^{\{\tau \in \mathbb{T} : \tau \leq t\}}$ such that $w|_t(\tau) = w(\tau)$ for all $\tau \leq t$.*

Definition 4 (Function with restricted domain). *We denote a function $w : \{\tau \in \mathbb{T} : \tau \leq t\} \rightarrow \mathbb{W}$ with the notation $w|_t$.*

The additional notation in [Definition 4](#) is meant to serve as a reminder that the values of $w|_t$ beyond t are treated as forgotten; in other words, $w|_t$ agrees with w on times up to t , but its domain cannot be expanded to get agreement with w for times later than t .

Definition 5 (Concatenation). *Given two behaviors $w_1 : \mathbb{T} \rightarrow \mathbb{W}$ and $w_2 : \mathbb{T} \rightarrow \mathbb{W}$ as well as two times $t_1, t_2 \in \mathbb{T}$, we introduce the following notation for the function*

obtained by gluing the suffix of w_1 starting at t_1 to the prefix of w_2 ending at t_2 .

That is, we define

$$w_2 \wedge_{t_1}^{t_2} w_1 := \begin{cases} w_2(\tau) & \tau \leq t_2 \\ w_1(\tau + t_1 - t_2) & \tau > t_2 \end{cases}.$$

Definition 6 (Projection). *In the sequel, π will denote a projection that returns a particular element of a tuple. There will be two varieties of this notation:*

- *when π is specified with a natural number, π_n will return the n^{th} element in the tuple; and*
- *when π is specified with a set A , π_A will return the element of the tuple taking values in the set A . We use this form only for readability: it is meant to have an exact counterpart in the other version of this notation.*

2.2.2 State Maps and a Notion of Bisimulation for Behavioral Dynamical Systems

In order to define a notion of bisimulation for behavioral dynamical systems, Julius et. al. [25] worked not with the raw trajectories, but rather with the output of a so-called state map. The reason for this will become clear subsequently. We repeat the definition below.

Definition 7 (State Map [25]). *A **state map** for a dynamical system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$, is a surjective map $\mathfrak{d} : \mathcal{B} \times \mathbb{T} \rightarrow \Delta$ with the following property:*

- Let $w_1 \in \mathcal{B}$ and $t_1 \in \mathbb{T}$. Then for all $w_2 \in \mathcal{B}$ and $t_2 \in \mathbb{T}$ such that $\mathfrak{d}(w_1, t_1) = \mathfrak{d}(w_2, t_2)$, there exists a $w'_1 \in \mathcal{B}$ with the property that $w'_1(t) = w_1(t) \forall t \leq t_1$ and $w'_1(t) = w_2(t + t_2 - t_1) \forall t > t_1$; in other words, $w_1 \wedge_{t_2}^{t_1} w_2 \in \mathcal{B}$.

Δ is a set associated with the state map \mathfrak{d} and that different state maps are allowed to have different codomains. Δ is called the **state space** of the state map \mathfrak{d} ; elements of Δ are called **states**.

Definition 8 (State System [25]). A state system (Σ, \mathfrak{d}) is a behavioral system $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ along with a state map $\mathfrak{d} : \mathcal{B} \times \mathbb{T} \rightarrow \Delta$.

The general idea of Definition 7 can be summarized as follows. When a state system is in a certain state, it doesn't matter what trajectory the system followed to get there: the legitimate future trajectories are always the same. This is also independent of *when* the system enters the state. In the sequel, we will need to refine the notion of a state map further in order to obtain meaningful results. Hence, we re-introduce the following two definitions from [25].

Definition 9 (Past induced state map [25]). Let $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ with a state map $\mathfrak{x} : \mathcal{B} \times \mathbb{T} \rightarrow X$. The state map \mathfrak{x} is **past induced** if for any $w_1, w_2 \in \mathcal{B}$ and $t \in \mathbb{T}$, $w_1|_t = w_2|_t$ implies that $\mathfrak{x}(w_1, t) = \mathfrak{x}(w_2, t)$.

Definition 10 (Markovian state map [25]). Let $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ with a state map $\mathfrak{x} : \mathcal{B} \times \mathbb{T} \rightarrow X$. The state map \mathfrak{x} is **Markov** if for any $w_1, w_2 \in \mathcal{B}$ and $t_1, t_2 \in \mathbb{T}$, then $\mathfrak{x}(w_1, t_1) = \mathfrak{x}(w_2, t_2)$ and $w_1(\tau - t_2 + t_1) = w_2(\tau) \forall \tau \in [t_2, t_2 + \delta t]$ implies that $\mathfrak{x}(w_1, \tau - t_2 + t_1) = \mathfrak{x}(w_2, \tau)$ for all $\tau \in [t_2, t_2 + \delta t]$.

Now we introduce the definition of bisimulation presented in [25]. However, we take a somewhat different approach: we bootstrap that definition of bisimulation from an appropriate definition of a simulation relation. In this way, the development parallels the previous discussion of bisimulation.

Definition 11 ((Bi)Simulation for Behavioral Systems with State Maps [25]). *Let $\Sigma_1 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_1, \mathcal{B}_1)$ and $\Sigma_2 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_2, \mathcal{B}_2)$ be two dynamical systems which share the same external signal space \mathbb{V} . Further suppose that each dynamical system has an associated state map $\mathfrak{r}_i : \mathbb{V} \times \mathbb{D}_i \rightarrow X_i$. Then a binary relation ${}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2} \subseteq X_1 \times X_2$ is a **simulation relation** (indicating Σ_2 **simulates** some portion of Σ_1) if and only if the following statement holds.*

- For any $(\chi_1, \chi_2) \in {}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2}$, $w_1 := (v_1, d_1) \in \mathcal{B}_1$, $t_1 \in \mathbb{T}$, $w_2 := (v_2, d_2) \in \mathcal{B}_2$ and $t_2 \in \mathbb{T}$ such that $(\mathfrak{r}_1(w_1, t_1), \mathfrak{r}_2(w_2, t_2)) = (\chi_1, \chi_2)$, there exists a $d'_2 \in \pi_{\mathbb{D}_2}\mathcal{B}_2$ such that:

1. $w'_2 := (v_2 \wedge_{t_1}^{t_2} v_1, d'_2) \in \mathcal{B}_2$
2. $\mathfrak{r}_2(w'_2, t_2) = \chi_2$
3. $d'_2|_{t_2} = d_2|_{t_2}$
4. $(\mathfrak{r}_1(w_1, \tau - t_2 + t_1), \mathfrak{r}_2(w'_2, \tau)) \in {}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2} \quad \forall \tau \geq t_2$.

If such a simulation relation ${}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2}$ has the property that $\forall \chi_1 \in X_1$ there exists $\chi_2 \in X_2$ such that $(\chi_1, \chi_2) \in {}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2}$, then we say $(\Sigma_2, \mathfrak{r}_2)$ **simulates** $(\Sigma_1, \mathfrak{r}_1)$. For such relations we will usually use the notation ${}_{\mathfrak{r}_1}\sqsubseteq_{\mathfrak{r}_2}$, and write $\Sigma_{1\mathfrak{r}_1}\sqsubseteq_{\mathfrak{r}_2}\Sigma_2$ to indicate that Σ_1 is **simulated** by Σ_2 .

A **bisimulation relation** is a simulation relation ${}_{\mathfrak{r}_1}\mathbf{B}_{\mathfrak{r}_2} \subseteq X_1 \times X_2$ for which the relation ${}_{\mathfrak{r}_1}\mathbf{B}_{\mathfrak{r}_2}^{-1} := \{\langle \chi_2, \chi_1 \rangle \in X_2 \times X_1 : \langle \chi_1, \chi_2 \rangle \in {}_{\mathfrak{r}_1}\mathbf{B}_{\mathfrak{r}_2}\}$ is also a simulation relation. $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ are then said to be **bisimilar** if there is a bisimulation relation ${}_{\mathfrak{r}_1}\sim_{\mathfrak{r}_2}$ such that $(\Sigma_2, \mathfrak{r}_2)$ simulates $(\Sigma_1, \mathfrak{r}_1)$ according to ${}_{\mathfrak{r}_1}\sim_{\mathfrak{r}_2}$ and $(\Sigma_1, \mathfrak{r}_1)$ simulates $(\Sigma_2, \mathfrak{r}_2)$ according to ${}_{\mathfrak{r}_1}\sim_{\mathfrak{r}_2}^{-1}$.

Intuitively, [Definition 11](#) says the following about states χ_1 and χ_2 that are related according to ${}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2}$: whenever a behavior from Σ_1 passes through state χ_1 , any behavior in Σ_2 that passes through *the related state* χ_2 – no matter *when* it does so – will have a kind of parity with the behavior from Σ_1 . In particular, the portion of the trajectory from Σ_1 that comes *after* state χ_1 can always be matched to a (possibly different) continuation of the trajectory that led to χ_2 in Σ_2 ; the “matching” is with respect to the external variables (item 3 in [Definition 11](#)) and membership of intermediate states in the relation ${}_{\mathfrak{r}_1}\mathbf{R}_{\mathfrak{r}_2}$ (item 4 in [Definition 11](#)).

The *system* Σ_2 *simulates* Σ_1 when we have a simulation relation with the additional property that every state from Σ_1 is related to some state in Σ_2 . Thus, Σ_2 can match the external behavior of Σ_1 no matter what state Σ_1 is in; in this sense, Σ_2 contains something like a “superset” of the (external) behavior of Σ_1 . Given this idea of what it means for one system to be a “superset” of the external behaviors of another, it is natural to specify a relation that indicates when two systems are externally “equivalent” to each other. This is the idea behind *bisimulation*.

[Definition 11](#) appears to be agnostic to the state map property of \mathfrak{r}_1 and \mathfrak{r}_2 , however this is actually not the case. If \mathfrak{r}_1 and \mathfrak{r}_2 weren’t state maps, then it might

be possible for two states χ_1 and χ_2 to satisfy the properties of [Definition 11](#) for some behaviors and times but not others. In fact, the state map property is necessary to assert that bisimulation is an equivalence relation because it assures the existence of certain trajectories; see Lemma 8 in [\[25\]](#).

2.3 Modal Logic and (Maximal) VHHM Classes of KSs

In this section, we describe the relevant background on HML, modal logic and (maximal) Hennessy Milner Classes. The material in this section is based largely on [\[26, 27\]](#). To assist with the intelligibility of this and subsequent sections, we reintroduce notation from [\[19\]](#) in [Table 2.1](#).

Hennessy-Milner Logic is defined as follows, and it is interpreted with the usual semantics over STs or Kripke Structures. Note that HML lacks propositional variables, so we have the subsequent definition for an HML-like logic that does include propositional variables.

Definition 12 (Hennessy-Milner Logic (HML)/Modal Logic). *Given a set of labels, L , **Hennessy-Milner Logic (HML)** [\[18\]](#) is the set of formulas $\Phi_{HML}(L)$ specified*

$s \supseteq_X t$: t simulates s (w.r.t. simulation notion X)
$s \leftrightarrow_X t$: s and t are bisimilar (w.r.t. simulation notion X)
$s \approx_Y t$: s and t satisfy the same formulas of logic Y

Table 2.1: Notation for simulation, bisimulation and modal equivalence.

as follows, where $\ell \in L$:

$$\varphi := \top \quad | \quad \neg\varphi \quad | \quad \varphi_1 \wedge \varphi_2 \quad | \quad \langle \ell \rangle \varphi. \quad (2.1)$$

When propositional variables are added to HML the result will be called a **modal logic**; the modal formulas with propositional variables Θ and labels L will be denoted by $\Phi_\Theta(L)$.

Of course in this chapter, we are primarily interested in studying the relationship between modal logics like HML and bisimulation: in particular, we are interested in when modal equivalence (according to HML, say) *implies* bisimulation equivalence. Hence we have the following definition.

Definition 13 (Visser/Hollenberg Hennessy-Milner Property [27]). *Let \mathfrak{K} be a set of Kripke structures. \mathfrak{K} is said to satisfy the **Visser/Hollenberg Hennessy-Milner Property (VHHM property)** with respect to $\Phi_\Theta(L)$ if for any two Kripke structures $\mathbf{S}, \mathbf{T} \in \mathfrak{K}$ and **any two states** $s' \in S$ and $t' \in T$*

$$s' \simeq t' \quad \Leftrightarrow \quad s' \approx_{\Phi_\Theta(L)} t'. \quad (2.2)$$

The foundation of *maximal* VHHM classes over KSs is the so-called canonical model [26], alternately referred to as the Henkin model in [27]. The canonical model is a special KS that is one of the most important tools in the study of (normal) modal logics. The most important feature of the canonical model is that its states (worlds) are maximally consistent sets of formulas, and the outgoing transitions from a particular state are selected in a way so as to be consistent with the formulas *in* that state: see the Henkin property in [Theorem 1](#).

Definition 14 (Canonical (Henkin) Model [26, 27]). Let $\Lambda \subseteq \Phi_{\Theta}(L)$ be a normal logic. Then the **canonical model** is the Kripke structure $\mathbf{C}^{\Lambda} = (S^{\Lambda}, \{R_{\ell}^{\Lambda} : \ell \in L\}, V^{\Lambda})$ defined as follows:

- the set of states (worlds), S^{Λ} , is the collection of maximally consistent sets of formulas with respect to the logic Λ ;
- for each $\ell \in L$, the transition relation $R_{\ell}^{\Lambda} \subseteq S^{\Lambda} \times S^{\Lambda}$ is defined such that $sR_{\ell}t$ if and only if $\varphi \in t$ implies that $\langle \ell \rangle \varphi \in s$.
- the valuation $V^{\Lambda} : \Theta \rightarrow S^{\Lambda}$ is defined such that $V(p) = \{s \in S^{\Lambda} : p \in s\}$.

As alluded to above, the primary reason to care about the canonical model is that the constituent formulas of a state tell you something about the formulas that are *true* in that state. This is the so called *Henkin property*: that is a world in S^{Λ} satisfies a modal formula if and only if that formula is an element of the world (recall that S^{Λ} is a collection of sets of formulas; that is $s \in S^{\Lambda}$ is a maximally consistent set of formulas with respect to the logic Λ).

Theorem 1 (The Canonical Model Satisfies the Henkin Property [26, 27]). For any state s in the canonical model \mathbf{C}^{Λ} and any formula $\varphi \in \Phi_{\Theta}(L)$: $s \models \varphi \iff \varphi \in s$ (the aforementioned **Henkin property**).

Importantly, though, the canonical model is not the *unique* KS over S^{Λ} to satisfy the Henkin Property. Thus we refer to such models using the following terminology from [27].

Definition 15 (Henkin-like model [27]). *Let \mathbf{C}^K be the canonical model associated with the smallest normal logic K . Then a **Henkin-like** model is any Kripke structure $\mathbf{HC}^K = (S^K, \{R_\ell^{\mathbf{HC}^K} \subseteq R_\ell^K : \ell \in L\}, V^K)$ that satisfies the Henkin property (see [Theorem 1](#) and the discussion preceding it).*

It is through such Henkin-like models that an elegant characterization of *maximal* VHHM classes is possible, as reported in [27].

Theorem 2 (Maximal VHHM Classes [27]). *Let \mathbf{HC}^K be any Henkin-like model, and let $\mathbf{S}(\mathbf{HC}^K)$ be the set of generated sub-models of \mathbf{HC}^K . Then*

1. *The set of all Kripke structures that are bisimilar to a model in $\mathbf{S}(\mathbf{HC}^K)$ is a maximal VHHM class; that is it is maximal in a set-theoretic sense. We denote such a class by $\mathbf{BS}(\mathbf{HC}^K)$.*
2. *Let \mathfrak{H} be any set of Kripke structures that satisfies the VHHM property. Then $\mathfrak{H} \subseteq \mathbf{BS}(\mathbf{HC}^K)$ for at least one Henkin-like model \mathbf{HC}^K .*

[19] offer the following description of the essence of [Theorem 2](#):

“... a class of models $\mathbf{BS}(\mathbf{HC}^K)$ is necessarily a VHHM class because modal equivalence is a bisimulation relation over a single Henkin-like model (each maximal set of formulas is satisfied only by its own unique state in the model). Thus, Henkin-like models effectively “canonicalize” different VHHM classes because a given Henkin-like model associates a particular transition structure with each and every (maximal) set of formulas that can be satisfied in any Kripke structure (K is sound and complete with respect to Kripke structures).” [19]

2.4 Hybrid Process Algebra (HyPA)

Hybrid Process Algebra (HyPA) [4] is a compositional algebraic framework for modeling hybrid systems that permit instantaneous jumps in their continuous model variables; the signature of HyPA reflects this by including reinitialization operators, flow clauses and disrupt operators. As in traditional process algebra, semantic derivation rules describe how transitions transform one term in the HyPA algebra into another; in HyPA, such transitions correspond to either a discrete action or a finite-duration continuous flow. However, the semantics of HyPA must be sensitive to the valuation of the continuous model variables in order to be meaningful: transitions must be enabled only for certain valuations of the model variables, and transitions must also be able to alter the model variables when they execute. Thus, the aforementioned derivation rules actually transform a HyPA term, valuation pair into another HyPA term, valuation pair. This makes it possible to use the semantic derivation rules of HyPA to construct a labeled transition system where each state (location) is specified by a HyPA term and a valuation of the model variables. These labeled transition systems are coined *hybrid transition systems* in [4].

2.4.1 HyPA Syntax

Broadly speaking, a HyPA process (or term) is defined as a (nested) algebraic expression formed by composing HyPA operators; there are several “atomic” processes that form the leaves of such expression trees – that is such atomic processes can be viewed as algebraic operators with arity zero. In HyPA, an atomic process

is one of the following: the deadlock process, δ ; the empty process, ε ; an atomic (discrete) label $a \in \mathcal{A}$; or a flow clause. Flow clauses are defined in terms of the valuations for “continuous” (usually real-valued) variables: flow clauses specify differential equations that induce a “flow” over those variables. Continuous model variables also appear in reinitialization predicates: such a predicate is defined by a reinitialization maps that “reset” the valuation of the continuous variables. The terminology for defining flow predicates and reinitialization predicates is described below in [Table 2.2](#).

\mathcal{V}_m	Model variables.
$\dot{\mathcal{V}}_m = \{\dot{x} x \in \mathcal{V}_m\}$	“Derived” (differentiated) model variables.
$\mathcal{V}_m^- = \{x^- x \in \mathcal{V}_m\}$	Variables “storing” the current value of model variables.
$\mathcal{V}_m^+ = \{x^+ x \in \mathcal{V}_m\}$	Variables “storing” the immediate future values of model variables.
$\mathcal{V}(x), x \in \mathcal{V}_m$	Domain of the model variable x .
$\mathcal{V} = \cup_{x \in \mathcal{V}_m} \mathcal{V}(x)$	The union of all variable domains.
$Val = \mathcal{V}_m \rightarrow \mathcal{V}$	The set of all variable valuations.
T	Time set; totally ordered set with a least element, 0.
$\mathcal{F} = \left\{ f \in (T \rightarrow Val) \mid \begin{array}{l} \text{dom}(f) = [0, t) \text{ for} \\ \text{some } t \in T \end{array} \right\}$	The set of all flows.
\mathcal{P}_f	Set of flow predicates (i.e. differential equations).
$\models_f \subseteq \mathcal{F} \times \mathcal{P}_f$	Satisfaction relation for a flow predicate. The flow <i>false</i> satisfies no flow predicates.
$\mathcal{R} = Val \times Val$	The set of reinitializations.
\mathcal{P}_r	Reinitialization predicates.
$\models_r \subseteq \mathcal{R} \times \mathcal{P}_r$	Satisfaction relation for a reinitialization predicate. Assume <i>true</i> , <i>false</i> $\in \mathcal{P}_r$.

Table 2.2: Modeling notation for HyPA [4].

Given these definitions, the syntax of HyPA is specified inductively according to the following table, [Table 2.3](#). Flow and reinitialization clauses are defined explicitly in the subsequent [Table 2.4](#).

$p :=$	δ	deadlock
	ϵ	empty process
	$a \in \mathcal{A}$	discrete action
	$c = (V P_f) \in C$	flow clause
	$(d \gg p)_{d \in D}$	reinitialization operator (where $d = [V P_r] \in D$ is a reinitialization clause)
	$p \oplus q$	alternative composition
	$p \odot q$	sequential composition
	$p \blacktriangleright q$	disrupt
	$p \triangleright q$	left-disrupt
	$p \parallel q$	parallel composition
	$p \ll q$	left-parallel composition
	$p q$	forced-synchronization
	$\partial_H(p)_{H \subseteq \mathcal{A}}$	encapsulation operators

Table 2.3: HyPA syntax; p and q are arbitrary HyPA processes. [\[4\]](#)

To be precise, the flow clauses and reinitialization clauses used in Table 2.3 are defined in terms of flow and reinitialization predicates (see Table 2.2); both types of clauses are defined specifically in the following table, Table 2.4.

$C = \left\{ \left(V P_f \right) \mid V \subseteq \mathcal{V}_m, P_f \in \mathcal{P}_f \right\}$	Set of flow clauses. A flow clause is a pair in $\mathcal{V}_m \times \mathcal{P}_f$; the specified model variables are not allowed to jump .
$D = \left\{ [V P_r] \mid V \subseteq \mathcal{V}_m, P_r \in \mathcal{P}_r \right\}$	Set of reinitialization clauses. A reinitialization clause is a pair in $\mathcal{V}_m \times \mathcal{P}_r$; the specified model variables are the only ones that are allowed to change .

Table 2.4: Flow and Reinitialization Clauses [4].

2.4.2 HyPA Semantics

The semantics of HyPA is defined in terms of *hybrid transition systems* [4].

Definition 16 (Hybrid Transition System [4]). *A hybrid transition system is a tuple $(X, A, \Sigma, \mapsto, \rightsquigarrow, \checkmark)$, consisting of a state space X , a set of action labels A , a set of flow labels Σ , and transition relation $\mapsto \subseteq X \times A \times X$ and $\rightsquigarrow \subseteq (X \times \Sigma \times X)$. Lastly, there is a termination predicate $\checkmark \subseteq X$.*

If the set of all HyPA processes over a set of recursive process variables \mathcal{V}_p is given by $\mathcal{T}(\mathcal{V}_p)$, then we can define the semantics of a HyPA process in terms of a specific Hybrid Transition System.

Definition 17 (Hybrid Transition System for a HyPA Process [4]). *A Hybrid Transition System for a HyPA process has constituents defined as follows:*

- $X = \mathcal{T}(\mathcal{V}_p) \times Val$;
- $A = \mathcal{A} \times Val$;
- $\Sigma = \mathcal{F}$ (the set of flows; see Table 2.2).

Then the transition relations \mapsto and \rightsquigarrow and the termination operator \checkmark will be denoted by:

- $\langle x \rangle \xrightarrow{a} \langle x' \rangle$ for $x, x' \in X$ and $a \in A$ to signify $(x, a, x') \in \mapsto$;
- $\langle x \rangle \rightsquigarrow \langle x' \rangle$ for $x, x' \in X$ and $\sigma \in \Sigma$ to signify $(x, \sigma, x') \in \rightsquigarrow$; and
- $\langle x \rangle \checkmark$ for $x \in X$ to signify $x \in \checkmark$.

The actual constituents of \mapsto , \rightsquigarrow and \checkmark will of course be specified by the subsequent interpretation of HyPA terms.

Now, it is only left to define the interpretation of flow clauses and reinitialization clauses before the remaining semantics of HyPA can be specified using SOS rules.

Definition 18 (Solution of a Flow Clause [4]). *A pair $(\nu, \sigma) \in Val \times \mathcal{F}$ is defined to be a solution to a flow clause $c \in C$, i.e. $(\nu, \sigma) \models c$, if:*

- $(\nu, \sigma) \models \left(V | P_f \right)$ if $\sigma \models_f P_f$ and for all $x \in V$, $\nu(x) = \sigma(0)(x)$; and
- $(\nu, \sigma) \models c \wedge c'$ if $(\nu, \sigma) \models c$ and $(\nu, \sigma) \models c'$.

Definition 19 (Solution of a Reinitialization Clause [4]). A reinitialization $(\nu, \nu') \in \mathcal{R}$ is defined to be a solution of a reinitialization clause $d \in D$, i.e. $(\nu, \nu') \models d$, if:

- $(\nu, \nu') \models [V|P_r]$ if $(\nu, \nu') \models_r P_r$ and for all $x \notin V$, $\nu(x) = \nu'(x)$;
- $(\nu, \nu') \models d' \vee d''$ if $(\nu, \nu') \models d'$ or $(\nu, \nu') \models d''$;
- $(\nu, \nu') \models d' \wedge d''$ if $(\nu, \nu') \models d'$ and $(\nu, \nu') \models d''$;
- $(\nu, \nu') \models d' \sim d''$ if there exists $v \in Val$ with $(\nu, v) \models d'$ and $(v, \nu') \models d''$;
- $(\nu, \nu') \models d' \stackrel{?}{\sim}$ if $\nu = \nu'$ and there exists $v \in Val$ with $(\nu, v) \models d'$; and
- $(\nu, \nu') \models c_{jmp}$ if there exists $\sigma \in \Sigma$ such that $(\nu, \sigma) \models c$ and $\sigma(0) = \nu'$.

The remainder of the HyPA semantics is defined by SOS rules. For completeness, these are reproduced from [4] in [Table 2.5](#) - [Table 2.9](#).

$$\begin{array}{c}
\frac{}{\langle \epsilon, v \rangle \checkmark} \text{(1)} \quad \frac{}{\langle a, v \rangle \xrightarrow{a,v} \langle \epsilon, v \rangle} \text{(2)} \quad \frac{(v, \sigma) \models c, \text{dom}(\sigma) = [0, t]}{\langle c, v \rangle \rightsquigarrow \langle c, \sigma(t) \rangle} \text{(3)} \\
\frac{(v, v') \models d, \langle p, v' \rangle \checkmark}{\langle d \gg p, v \rangle \checkmark} \text{(4)} \quad \frac{(v, v') \models d, \langle p, v' \rangle \xrightarrow{l} \langle p', v'' \rangle}{\langle d \gg p, v \rangle \xrightarrow{l} \langle p', v'' \rangle} \text{(5)}
\end{array}$$

Table 2.5: Operational Semantics of HyPA. Reproduction of Table 1 in [4].

$$\begin{array}{c}
\frac{\langle p, v \rangle \checkmark}{\langle p \oplus q, v \rangle \checkmark} \text{(6)} \quad \frac{\langle p, v \rangle \xrightarrow{l} \langle p', v' \rangle}{\langle p \oplus q, v \rangle \xrightarrow{l} \langle p', v' \rangle} \text{(7)} \quad \frac{\langle p, v \rangle \checkmark, \langle q, v \rangle \checkmark}{\langle p \odot q, v \rangle \checkmark} \text{(8)} \\
\frac{\langle q, v \rangle \checkmark}{\langle q \oplus p, v \rangle \checkmark} \quad \frac{\langle q, v \rangle \xrightarrow{l} \langle q', v' \rangle}{\langle q \oplus p, v \rangle \xrightarrow{l} \langle q', v' \rangle} \\
\frac{\langle p, v \rangle \xrightarrow{l} \langle p', v' \rangle}{\langle p \odot q, v \rangle \xrightarrow{l} \langle p' \odot q, v' \rangle} \text{(9)} \quad \frac{\langle p, v \rangle \checkmark, \langle q, v \rangle \xrightarrow{l} \langle q', v' \rangle}{\langle p \odot q, v \rangle \xrightarrow{l} \langle q', v' \rangle} \text{(10)}
\end{array}$$

Table 2.6: Operational Semantics of HyPA: alternative and sequential composition.

Reproduction of Table 2 in [4].

$$\begin{array}{c}
\frac{\langle p, v \rangle \checkmark}{\langle p \blacktriangleright q, v \rangle \checkmark} \text{(11)} \quad \frac{\langle p, v \rangle \xrightarrow{l} \langle p', v' \rangle}{\langle p \blacktriangleright q, v \rangle \xrightarrow{l} \langle p' \blacktriangleright q, v' \rangle} \text{(12)} \\
\frac{\langle p \triangleright q, v \rangle \checkmark}{\langle p \blacktriangleright q, v \rangle \checkmark} \quad \frac{\langle p \triangleright q, v \rangle \xrightarrow{l} \langle p' \blacktriangleright q, v' \rangle}{\langle p \blacktriangleright q, v \rangle \xrightarrow{l} \langle p' \blacktriangleright q, v' \rangle} \\
\frac{\langle q, v \rangle \checkmark}{\langle p \blacktriangleright q, v \rangle \checkmark} \text{(13)} \quad \frac{\langle q, v \rangle \xrightarrow{l} \langle q', v' \rangle}{\langle p \blacktriangleright q, v \rangle \xrightarrow{l} \langle q', v' \rangle} \text{(14)}
\end{array}$$

Table 2.7: Operational Semantics of HyPA: disrupt operator. Reproduction of Table

3 in [4].

Operational semantics of HyPA, parallel composition

$\frac{\langle p, v \rangle \checkmark, \langle q, v \rangle \checkmark}{\langle p \parallel q, v \rangle \checkmark} \quad \langle p \mid q, v \rangle \checkmark} (15)$	$\frac{\langle p, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle, \langle q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle q', v' \rangle}{\langle p \parallel q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p' \parallel q', v' \rangle} \quad \langle p \mid q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p' \mid q', v' \rangle} (16)$
$\frac{\langle p, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle, \langle q, v \rangle \checkmark}{\langle p \parallel q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle} \quad \langle q \parallel p, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle} (17)$	$\frac{\langle p, v \rangle \overset{a, v'}{\mapsto} \langle p', v'' \rangle}{\langle p \parallel q, v \rangle \overset{a, v'}{\mapsto} \langle p' \parallel q, v'' \rangle} \quad \langle q \parallel p, v \rangle \overset{a, v'}{\mapsto} \langle q \parallel p', v'' \rangle} (18)$
$\frac{\langle p, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle, \langle q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle}{\langle p \parallel q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle} \quad \langle p \parallel q, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle} (19)$	
$\frac{\langle p, v \rangle \overset{a, v'}{\mapsto} \langle p', v'' \rangle, \langle q, v \rangle \overset{a', v'}{\mapsto} \langle q', v'' \rangle, a'' = a \gamma a'}{\langle p \parallel q, v \rangle \overset{a'', v'}{\mapsto} \langle p' \parallel q', v'' \rangle} \quad \langle p \mid q, v \rangle \overset{a'', v'}{\mapsto} \langle p' \mid q', v'' \rangle}$	

Table 2.8: Operational Semantics of HyPA: parallel composition. Reproduction of Table 4 in [4].

Operational semantics of HyPA, encapsulation and recursion

$\frac{\langle p, v \rangle \overset{a, v'}{\mapsto} \langle p', v'' \rangle, a \notin H}{\langle \partial_H(p), v \rangle \overset{a, v'}{\mapsto} \langle \partial_H(p'), v'' \rangle} (20)$
$\frac{\langle p, v \rangle \overset{\sigma}{\rightsquigarrow} \langle p', v' \rangle}{\langle \partial_H(p), v \rangle \overset{\sigma}{\rightsquigarrow} \langle \partial_H(p'), v' \rangle} (21) \quad \frac{\langle p, v \rangle \checkmark}{\langle \partial_H(p), v \rangle \checkmark} (22)$
$\frac{\langle p, v \rangle \checkmark}{\langle X, v \rangle \checkmark} (23) \quad X \approx p \in E \quad \frac{\langle p, v \rangle \xrightarrow{l} \langle p', v' \rangle}{\langle X, v \rangle \xrightarrow{l} \langle p', v' \rangle} (24) \quad X \approx p \in E$

Table 2.9: Operational Semantics of HyPA: encapsulation and recursion. Reproduction of Table 5 in [4].

2.4.3 Bisimulation for HyPA

In the context of hybrid transition systems, the standard notion of bisimulation for labeled transition systems with marked states may be trivially applied; this appears as Definition 4 in [4]. However, bisimulation as such is not a congruence for HyPA processes, because the valuation of the model variables plays a fundamental role in the semantic derivation rules of HyPA (see Table 2.5 - Table 2.9 in Section 2.4.2). This nuance has two specific facets. First, the semantics of HyPA are such that the label on a transition determines exactly the valuation of the model variables after such a transition is performed (this appears inconspicuously in Appendix A of [4] as Lemma A.1). This feature of the HyPA semantics considerably prunes the reachable states in a hybrid transition system; it also forces synchrony of valuations between processes that perform the same sequence of labels, and reduces nondeterminism in the valuation of the model variables to a choice that is made before a process starts performing transitions. Second, HyPA operators operate only on process terms, so non-unary operators are unable to maintain distinct trajectories of the model variables for each of the subprocesses supplied as operands. This is in stark contrast to traditional processes algebra, where all of the state information is contained in an algebraic expression; in that setting, a non-unary operator effectively maintains separate state spaces for *each* of the subprocesses supplied as operands. These factors work in concert to break bisimulation as a congruence for HyPA: two processes which are bisimilar need only agree about behavior on their reachable states (the first point), yet the parallel composition operator allows a third

process to introduce new states from which the original processes can execute by influencing the common (i.e. global) model valuations (the second point).

Unfortunately, it is a reasonable, even necessary, choice to require global variable valuations. This inflexibility suggests only one means to obtain congruence with respect to HyPA: in order to substitute a process q for a process p , a bisimulation relation must be exhibited that includes all of the subprocess, valuation pairs that are hidden by an initial choice of valuation. This is exactly the idea behind stateless bisimulation, which is indeed a congruence for HyPA [4]. This is an important conclusion in [4], although it appears only in relation to a more complicated notion of congruence, called robust bisimulation; it is proved in [4] that robust bisimulation is equivalent to stateless bisimulation in the context of HyPA. For this reason, we confine ourselves to the consideration of stateless bisimulation. The definition of stateless bisimulation is repeated below.

Definition 20 (Stateless bisimulation [4]). *Given a hybrid transition system with state space $\mathcal{T}(\mathcal{V}_p) \times \text{Val}$, a **stateless bisimulation relation** is a binary relation $R \subseteq \mathcal{T}(\mathcal{V}_p) \times \mathcal{T}(\mathcal{V}_p)$ such that for all $\nu, \nu' \in \text{Val}$ and pRq ,*

- $\langle p, \nu \rangle \in \checkmark$ implies $\langle q, \nu \rangle \in \checkmark$,
- $\langle q, \nu \rangle \in \checkmark$ implies $\langle p, \nu \rangle \in \checkmark$,
- $\langle p, \nu \rangle \xrightarrow{\ell} \langle p', \nu' \rangle$ implies $\exists q' \in \mathcal{T}(\mathcal{V}_p)$ such that $\langle q, \nu \rangle \xrightarrow{\ell} \langle q', \nu' \rangle \wedge p'Rq'$ and
- $\langle q, \nu \rangle \xrightarrow{\ell} \langle q', \nu' \rangle$ implies $\exists p' \in \mathcal{T}(\mathcal{V}_p)$ such that $\langle p, \nu \rangle \xrightarrow{\ell} \langle p', \nu' \rangle \wedge p'Rq'$.

Chapter 3: Generalized Synchronization Trees

3.1 Generalized Synchronization Trees

This section contains the foundational ideas on which much of the remainder of this dissertation depends: it is here that we formally define Generalized Synchronization Trees as a modeling structure that generalizes Milner's Synchronization Trees.

3.1.1 Generalizing Trees

This section presents basic background on partial and total orders and reviews a classical definition of tree in this setting.

Definition 21 (Partial Order). *Let P be a set, and let $\preceq \subseteq P \times P$ be a binary relation on P . Then \preceq is a partial order on P if the following hold.*

1. \preceq is reflexive: for all $p \in P$, $p \preceq p$.
2. \preceq is anti-symmetric: for all $p_1, p_2 \in P$, if $p_1 \preceq p_2$ and $p_2 \preceq p_1$ then $p_1 = p_2$.
3. \preceq is transitive: for all $p_1, p_2, p_3 \in P$, if $p_1 \preceq p_2$ and $p_2 \preceq p_3$ then $p_1 \preceq p_3$.

We abuse terminology and refer to $\langle P, \preceq \rangle$ as a partial order if \preceq is a partial order over set P . We write $p_1 \prec p_2$ if $p_1 \preceq p_2$ and $p_1 \neq p_2$ and $p_2 \succeq p_1$ if $p_1 \preceq p_2$.

We adapt the usual interval notation for numbers to partial orders as follows.

$$[p_1, p_2] \triangleq \{p \in P \mid p_1 \preceq p \preceq p_2\}$$

$$(p_1, p_2) \triangleq \{p \in P \mid p_1 \prec p \prec p_2\}$$

Half-open intervals, e.g. $[p_1, p_2)$ and $(p_1, p_2]$, have the obvious definitions.

Definition 22 (Upper/Lower Bounds). *Fix partial order $\langle P, \preceq \rangle$ and $P' \subseteq P$.*

1. $p \in P$ is an upper (lower) bound of P' if for every $p' \in P'$, $p' \preceq p$ ($p \preceq p'$).
2. $p \in P$ is the least upper (greatest lower) bound of P' if p is an upper (lower) bound of P' and for every upper (lower) bound p' of P' , $p \preceq p'$ ($p' \preceq p$).

When set P' has a least upper bound (greatest lower bound) we sometimes use $\sup P'$ ($\inf P'$) to denote this element.

Definition 23 (Total Order). *Let $\langle P, \preceq \rangle$ be a partial order. Then \preceq is a total or linear order on P if for every $p_1, p_2 \in P$, either $p_1 \preceq p_2$ or $p_2 \preceq p_1$.*

If $\langle P, \preceq \rangle$ is a partial order and $P' \subseteq P$, then we sometimes abuse notation and write $\langle P', \preceq \rangle$ for the partial order obtained by restricting \preceq to elements in P' . We say that P' is *totally ordered* by \preceq if \preceq is a total order for P' . We refer to P' as a *linear subset* of P in this case. Trees may now be defined as follows [28].

Definition 24 (Tree [28]). *A tree is partial order $\langle P, \preceq \rangle$ such that for each $p \in P$, the set $\{p' \in P \mid p' \preceq p\}$ is totally ordered by \preceq . If there is also a $p_0 \in P$ such that $p_0 \preceq p$ for all $p \in P$, then p_0 is called the root of the tree, and $\langle P, \preceq, p_0 \rangle$ is said to be a rooted tree.*

In [29], these structures are referred to as prefix orders. The distinguishing feature of a tree implies that \succsim defines a notion of ancestry. In a rooted tree, the root is an ancestor of every node, so every node has a unique “path” to the root. Since the subsequent development is modeled on synchronization trees, we will consider only rooted trees in the sequel.

We conclude this section by discussing a notion of discreteness for trees.

Definition 25 (Discrete Tree). *A tree $\langle P, \preceq, p_0 \rangle$ is discrete if and only if for every p , the set $[p_0, p]$ is finite.*

The following alternative characterization of discreteness is sometimes useful.

Proposition 1. *A tree $\langle P, \preceq, p_0 \rangle$ is discrete if and only if the following all hold.*

1. *For every $p \neq p_0$, $\sup[p_0, p] \in [p_0, p]$.*
2. *For every $p \in P$ and $p' \succ p$, $\inf(p, p'] \in (p, p']$.*
3. *Every nonempty linear subset P' of P has a greatest lower bound.*

3.1.2 Generalized Synchronization Trees

The impact of Milner’s work is hard to overstate; process algebra is a major field of study in computing, and the notions of simulation and bisimulation have had a substantial influence on other areas such as control-system modeling and systems biology, where the focus is on continuous, rather than discrete, behavior. However, the rich array of composition operators, and associated elegant metatheoretical results [30, 31] found in traditional process algebra have yet to emerge in these more

general contexts. Our motivation for generalized synchronization trees is to provide a flexible framework analogous to synchronization trees over which composition operations may be easily defined, and their algebraic properties studied, for this more general setting.

Synchronization trees are intended to model discrete systems that evolve via the execution of atomic actions. This phenomenon is evident in the fact that trees have edges that are labeled by these actions; each node in a tree is thus at most finitely many transitions from the root. For systems that have continuous as well as discrete dynamics, synchronization trees offer a problematic foundation for system modeling, since the notion of continuous trajectory is missing.

Generalized synchronization trees are intended to provide support for discrete, continuous, and hybrid notions of computation, where nondeterminism (branching) may also be discrete, continuous, or both.

Definition 26 (Generalized Synchronization Tree). *Let L be a set of labels. Then a generalized synchronization tree (GST) is a tuple $\langle P, \preceq, p_0, \mathcal{L} \rangle$, where:*

1. $\langle P, \preceq, p_0 \rangle$ is a tree in the sense of [Definition 24](#); and
2. $\mathcal{L} \in P \setminus \{p_0\} \rightarrow L$ is a (possibly partial) labeling function.

A GST differs from a synchronization tree in two respects. On the one hand, the tree structure is made precise by reference to [Definition 24](#). On the other hand, labels are attached to (non-root) nodes, rather than edges; indeed, a GST may not in general have a readily identifiable notion of edge.

In the rest of this section we show how different classes of systems may be encoded as GSTs. These examples contain different mixtures of discrete / continuous time and discrete / continuous nondeterminism (called “choice”).

3.2 Bisimulation for GSTs

[Section 2.1](#) defined standard notions of equivalence (bisimulation) and refinement (simulation) on synchronization trees. The goal of this section is to study adaptations of these notions for generalized synchronization trees. In the process of doing so, we highlight a subtlety that arises because of GSTs’ capability of modeling non-discrete time. As the notions of simulation and bisimulation are closely linked (see [Definition 1](#)), in what follows we focus our attention on simulation.

3.2.1 Notions of Bisimulation for GSTs

Simulation relations in [Definition 1](#) rely on a notion, labeled edges, that synchronization trees possess but GSTs do not. However, an intuition underlying the simulation relation is that if $T_1 \sqsubseteq T_2$, then every “execution step” of T_1 can be matched by T_2 in such a way that the resulting trees are still related. This observation provides a starting point for simulations on GSTs; rather than relying on edges to define computation, use the notion *trajectory* instead.

Definition 27 (Trajectory). *Let $\langle P, \preceq, p_0, \mathcal{L} \rangle$ be a GST, and let $p \in P$. Then a trajectory from p is a linear subset $P' \subseteq P$ such that for all $p' \in P'$:*

1. $p' \succ p$ and

2. $(p, p'] \subseteq P'$.

A trajectory from a node p in a GST is a path that starts from p , but for technical reasons, does not include p . A trajectory can be bounded with a maximal element as in the case of the interval $(p, p']$, or it can be bounded with a least upper bound as in the case of (p, p') . It is also possible for a trajectory to be bounded without a least upper bound or even unbounded.

Trajectories are analogous to computations and thus will form the basis of the simulation relations given below. In order to determine when two trajectories “match”, we introduce the concept of *order-equivalence*.

Definition 28 (Order Equivalence). *Let $\langle P, \preceq_P, p_0, \mathcal{L}_P \rangle$ and $\langle Q, \preceq_Q, q_0, \mathcal{L}_Q \rangle$ be GSTs, and T_p, T_q be trajectories from $p \in P$ and $q \in Q$ respectively. Then T_p and T_q are order-equivalent if there exists a bijection $\lambda \in T_P \rightarrow T_Q$ such that:*

1. $p_1 \preceq_P p_2$ if and only if $\lambda(p_1) \preceq_Q \lambda(p_2)$ for all $p_1, p_2 \in T_P$, and
2. $\mathcal{L}_P(p) = \mathcal{L}_Q(\lambda(p))$ for all $p \in T_P$.

When λ has this property, we say that λ is an order equivalence from T_P to T_Q .

Two trajectories that are order-equivalent can be seen as possessing the same “content”, as given by the labeling functions of the trees, in the same “order”. Note that in general, the bijections used to relate two order-equivalent trajectories need not be unique, although when the trees in question are discrete, they must be. The first notion of simulation may now be given as follows.

Definition 29 (Weak Simulation for GSTs). *Let $G_1 = \langle P, \preceq_P, p_0, \mathcal{L}_P \rangle$ and $G_2 = \langle Q, \preceq_Q, q_0, \mathcal{L}_Q \rangle$ be GSTs. Then $R \subseteq P \times Q$ is a weak simulation from G_1 to G_2 if, whenever $\langle p, q \rangle \in R$ and $p' \succeq p$, then there is a $q' \succeq q$ such that:*

1. $\langle p', q' \rangle \in R$, and
2. Trajectories $(p, p']$ and $(q, q']$ are order-equivalent.

$G_1 \sqsubseteq_w G_2$ if there is a weak simulation R from G_1 to G_2 with $\langle p_0, q_0 \rangle \in R$.

Weak bisimulation equivalence can be defined easily. Call a weak simulation R from G_1 to G_2 a weak bisimulation if R^{-1} is a weak simulation from G_2 to G_1 . Then $G_1 \sim_w G_2$ iff there is a weak bisimulation R with $\langle p_0, q_0 \rangle \in R$.

Weak simulation appears to be the natural extension of simulation to GSTs: for one node to be simulated by another, each bounded trajectory from the first node must be appropriately “matched” by an equivalent trajectory from the second node. However, one may impose a stronger condition on the trajectories emanating from related nodes, as follows.

Definition 30 (Strong Simulation for GSTs). *Let $G_1 = \langle P, \preceq_P, p_0, \mathcal{L}_P \rangle$ and $G_2 = \langle Q, \preceq_Q, q_0, \mathcal{L}_Q \rangle$ be GSTs. Then $R \subseteq P \times Q$ is a strong simulation from G_1 to G_2 if, whenever $\langle p, q \rangle \in R$ and T_p is a trajectory from p , there is a trajectory T_q from q and bijection $\lambda \in T_p \rightarrow T_q$ such that:*

1. λ is an order equivalence from T_p to T_q , and
2. $\langle p', \lambda(p') \rangle \in R$ for all $p' \in T_p$.

$G_1 \sqsubseteq_s G_2$ if there is a strong simulation R from G_1 to G_2 with $\langle p_0, q_0 \rangle \in R$.

Strong simulations strengthen weak ones by requiring that matching trajectories also pass through nodes that are related by the simulation relation, and by also considering potentially unbounded trajectories as well as bounded ones.

3.2.2 Relating Strong and Weak Simulations

This section now considers the relationships between weak and strong simulation. The first result indicates that the latter is indeed stronger than the former.

Theorem 3. *Let G_1 and G_2 be GSTs with $G_1 \sqsubseteq_s G_2$. Then $G_1 \sqsubseteq_w G_2$.*

The proof follows from the fact that every strong simulation is a weak simulation.

The next result, coupled with the previous one, establishes that for discrete trees, the two simulation orderings in fact coincide.

Theorem 4. *Suppose that G_1 and G_2 are discrete GSTs, and that $G_1 \sqsubseteq_w G_2$. Then $G_1 \sqsubseteq_s G_2$.*

The proof uses induction on transitions to show that any weak simulation is also strong.

We now show that $\sqsubseteq_w / \sqsubseteq_s$ coincides with the simulation ordering, \sqsubseteq , given for synchronization trees (i.e. discrete, finite-branching GSTs) in [Definition 1](#). The next definition defines a notion of \xrightarrow{a} for discrete GSTs.

Definition 31 (Transitions for Discrete GSTs). *Let $G = \langle P, \preceq, p_0, \mathcal{L} \rangle$ be a GST, with $p, p' \in P$.*

1. p' is an immediate successor of p if $p' \succ p$ and there exists no $p'' \in P$ such that $p' \succ p'' \succ p$.
2. $G[p$, the subtree of G rooted at p , is $\langle P', \preceq', p, \mathcal{L}' \rangle$, where $P' = \{p' \in P \mid p \preceq p'\}$, and \preceq' / \mathcal{L}' are the restrictions of \preceq and \mathcal{L} to $P' / P' \setminus \{p\}$.
3. Let $G' = \langle P', \preceq', p'_0, \mathcal{L}' \rangle$ be a GST. Then $G \xrightarrow{a} G'$ exactly when $p'_0 \in P$, p'_0 is an immediate successor of p_0 , $G' = G[p'_0$, and $\mathcal{L}(p') = a$.

Intuitively, $G \xrightarrow{a} G'$ if G' is an immediate subtree of G and the root of G' labeled by a . Based on this notion, [Definition 1](#) may now be applied to discrete GSTs. We have the following.

Theorem 5. *Let G_1, G_2 be discrete GSTs. Then the following are equivalent.*

1. $G_1 \sqsubseteq G_2$.
2. $G_1 \sqsubseteq_w G_2$.
3. $G_1 \sqsubseteq_s G_2$.

One might hope that \sqsubseteq_w and \sqsubseteq_s would coincide for general GSTs, thereby obviating the need for two notions. Unfortunately, this is not the case.

Theorem 6. *There exist GSTs G_1 and G_2 such that $G_1 \sqsubseteq_w G_2$ but $G_1 \not\sqsubseteq_s G_2$.*

Proof. Consider the GSTs depicted in [Fig. 3.1](#). It turns out that the trees G_1 and G_2 are such that $G_1 \sqsubseteq_w G_2$, but $G_1 \not\sqsubseteq_s G_2$.

Both G_1 and G_2 use a label set $\{\alpha, \beta\}$, and both are discrete except for their start states. Each tree is constructed from a basic “time axis” $T = \{1/n \mid n \in$

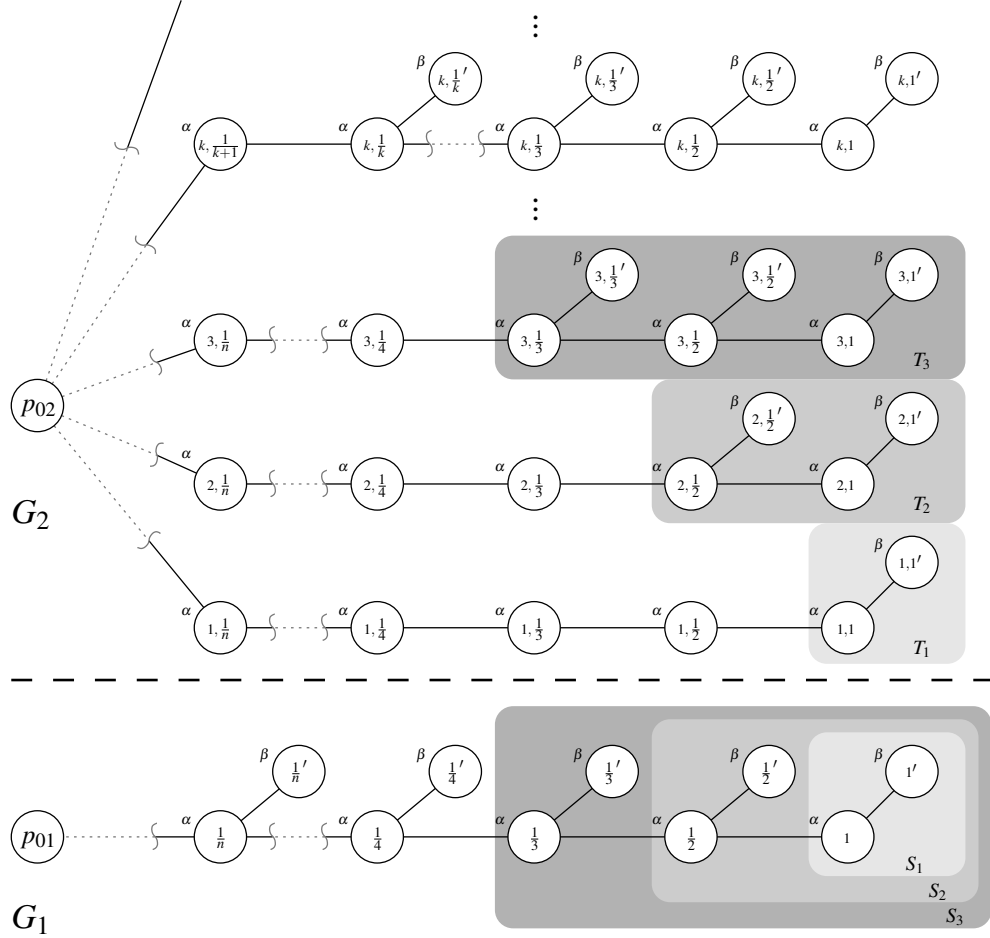


Figure 3.1: Visualization of the GSTs used in proof of [Theorem 6](#).

$\mathcal{N} \setminus \{0\} \cup \{0\}$, with the usual ordering \leq . The start state G_1 corresponds to time 0; each subsequent node is labeled by α if there is an edge to the next time point, or β if the node is maximal. In traditional synchronization-tree terms, each (non-start) node has an outgoing labeled α and another labeled β . G_2 is similar to G_1 except it contains an infinite number of branches from the start state, with branch k only enabling β transitions for the last k nodes.

The shading in the diagram illustrates a weak simulation that may be constructed and used to show that the start states in G_1 and G_2 are indeed related by

a weak simulation. Intuitively, every trajectory from the start node of G_1 leads to a node from which a finite number of α s are possible, with a β possible at each step also. This trajectory can be matched by one from the start state of G_2 that leads to a branch from which enough β -less nodes can be bypassed.

On the other hand, no strong simulation can be constructed relating the start node of G_1 with G_2 . The basic intuition is that any trajectory leading from the start node of G_1 has β s enabled at every intermediate node, and this behavior does not exist in any trajectory leading from the start node of G_2 . \square

The preceding result suggests that simulation is more nuanced for GSTs than for synchronization trees. One naturally may wonder which of the two notions proposed in this section is the “right” one. Our perspective is that the strong notion possesses a sense of invariance that one might expect for simulation; if one system is simulated by the other then any execution of the former can be “traced” by the latter following only related states. In this sense, strong simulation may be seen to have stronger intuitive justifications than the weaker one.

Chapter 4: Generalized Synchronization Trees and State Systems

4.1 Introduction

The advantages of STs have thus far been thoroughly celebrated. However, these advantages would do little good if STs bore no relationship to the study of other, more traditional discrete models. Thankfully, and as their popularity attests, this is not the case: as we have described already, an ordinary Labeled Transition System (LTS) that possesses an initial state may be “unrolled” into a ST – recall the example in [Fig. 2.1](#). The value of this procedure, however, is that a LTS and its “unrolling” (as a ST) are *bisimilar*. Thus, we can study bisimulation-invariant properties – including composition – with whichever structure is more convenient. And because each node in a ST has a *unique* incoming edge, “unrolling” a discrete system into a tree (or ST) also “unrolls” all of the loops in the former. This is an extremely useful structural alteration for various purposes, e.g. as a proof technique in modal logic [\[32\]](#).

Since the selling point of GSTs is to have a ST-like analog for continuous and hybrid system models, it is natural to ask whether GSTs, too, can be viewed as an “unrolling” of continuous system models – with some notion of bisimulation preserved in the process. This chapter addresses that question by showing how to

“unroll” system models from a modeling framework with a fully *non-discrete semantics*: that is the behavioral framework of Willems et. al. [14]. Most importantly, we also examine how bisimulation for such behavioral systems (see [25]; see also [Section 2.2](#)) is preserved by this unrolling procedure. On one hand, conversion to a GST is demonstrated to respect bisimulation between two behavioral systems: that is two behavioral systems with bisimilar state spaces are unrolled into GSTs that are bisimilar according to the definition of strong bisimulation in [15]. On the other hand, we exhibit a condition under which strong bisimulation between two unrolled behavioral systems implies that they have bisimilar state spaces. In this latter case, strong bisimulation between the GSTs is insufficient because GSTs (and their bisimulations) have no inherent awareness of (real) time; the extra condition requires that time information be preserved by the bisimulation relation. Others have recognized the importance of time in the context of bisimulation for CPSs (see e.g. [33]), but our work remains unique in the context of GSTs, and indeed, tree-like CPS models.

4.2 Behavioral Dynamical Systems as GSTs

In this section, we present a method for constructing a GST from a behavioral system $\Sigma = (\mathbb{T}, \mathbb{V} \times \mathbb{D}, \mathcal{B})$. As before, \mathbb{V} represents the external signal space and \mathbb{D} represents the internal signal space of Σ . **With a view to comparison with [25], we further assume that $\mathbb{T} = \mathbb{R}^{\geq 0}$.** According to [Definition 26](#), we need to identify a set P with a tree partial order, a set of labels L and a labeling function \mathcal{L} .

In order to create a tree partial order from a behavioral system, we appeal to the intuition of unrolling a LTS into a ST. This procedure, described in the introduction, establishes the nodes of the ST (i.e. its states) as sequences of transitions from the underlying LTS; moreover these nodes can be partially ordered by *prefixing*. This idea generalizes to a state map that can be defined for any behavioral system; this canonical state map then captures the possibility of partially ordering *states* by prefixing.

Proposition 2. *Let $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ be a dynamical system, and let $\mathcal{B}' := \{f|_t : \exists f' \in \mathcal{B}, t \in \mathbb{T} \text{ s.t. } f|_t = f'|_t\}$. For $\Phi = \mathcal{B}' \times \mathbb{T}$, the map $\mathbf{p} : \mathcal{B}' \times \mathbb{T} \rightarrow \Phi$, $\mathbf{p} : \langle f, t \rangle \mapsto \langle f|_t, t \rangle$ is a state map with state space Φ .*

Proof. This follows because $\Phi(f, t_1) = \Phi(g, t_2)$ if and only if $t_1 = t_2$ and $f|_{t_1} = g|_{t_2} = g|_{t_1}$. Thus, any trajectory which extends f must also extend g , because f and g have exactly the same values for all times up to $t_1 = t_2$. \square

Definition 32 (Canonical Partial-Order State Map). *We call the state map \mathbf{p} of [Proposition 2](#) the **canonical partial-order state map**. For a given dynamical system Σ , we will denote this state map by \mathbf{p}_Σ and its state space by Φ_Σ .*

The state map \mathbf{p} is clearly past-induced and Markovian; this is an easy consequence of [Definition 7](#). These properties will be important subsequently in the context of bisimulation. For now, though, the fact that states of \mathbf{p}_Σ are partial trajectories means that they can easily be ordered by prefixing; that is two states will be ordered if the function in one is a prefix of function in another. This captures

the unrolling idea mentioned before, and it further implies a *tree* partial order over the states – the precursor to a GST. This is made precise in the following definition.

Definition 33 (GST Representation of a Behavioral Dynamical System). *Let $\Sigma = (\mathbb{T}, \mathbb{V} \times \mathbb{D}, \mathcal{B})$ be a behavioral dynamical system with external variable space \mathbb{V} . Then the **GST representation of Σ** is given in terms of \mathfrak{p}_Σ as follows:*

- The tree partial order is defined over the set P

$$\begin{aligned} P &= \{\langle f|_t, t \rangle : f \in \mathcal{B}, t \in \mathbb{T}\} \cup \{p_0\} \\ &= \{\mathfrak{p}_\Sigma(f, t) : f \in \mathcal{B}, t \in \mathbb{T}\} \cup \{p_0\}, \end{aligned}$$

where p_0 will be the root of the tree, and is chosen to make the union disjoint.

- The set P is partially ordered by \preceq where

$$\begin{aligned} \mathfrak{p}_\Sigma(f_1, t_1) \preceq \mathfrak{p}_\Sigma(f_2, t_2) &= \langle f_1|^{t_1}, t_1 \rangle \preceq \langle f_2|^{t_2}, t_2 \rangle \\ &\text{iff } t_1 \leq t_2 \bigwedge f_1|^{t_1} = (f_2|^{t_2})|_{t_1} \end{aligned}$$

and

$$p_0 \preceq \mathfrak{p}_\Sigma(f_1, t_1) \text{ for all } \langle f_1, t_1 \rangle \in \mathcal{B} \times \mathbb{T}.$$

- The labeling function is defined for the set of labels $L = \mathbb{V}$ by

$$\mathcal{L} : \mathfrak{p}_\Sigma(f_1, t_1) = \langle f_1|^{t_1}, t_1 \rangle \mapsto \pi_{\mathbb{V}}(f_1|^{t_1}(t_1)).$$

Proposition 3. *The partial order defined in [Definition 33](#) is a tree partial order, and hence, $(P, p_0, \preceq, \mathcal{L})$ as defined therein is a GST.*

While [Definition 33](#) in some sense captures the idea of “unrolling” a LTS, it does so without considering any state information about the dynamical system. No such extra consideration is need in the LTS case because trajectories consist of the states themselves (through the sequences of valid transitions). However, for behavioral systems, the trajectories under consideration do not necessarily consist of states. Nevertheless, the canonical partial order state map \mathfrak{p}_Σ is by its nature enough to represent – up to bisimulation – any other past-induced, Markovian state map over the same system. The following proposition makes this clear.

Proposition 4 (Bisimilarity between (Σ, \mathfrak{r}) and $(\Sigma, \mathfrak{p}_\Sigma)$). *Let Σ be a behavioral dynamical system with a past induced, Markovian state map \mathfrak{r} . Then the state systems (Σ, \mathfrak{r}) and $(\Sigma, \mathfrak{p}_\Sigma)$ are bisimilar according to [Definition 11](#).*

Proof. We seek a bisimulation relation $\mathfrak{B}_{\mathfrak{p}_\Sigma} \subseteq X \times \Phi_\Sigma$ where $\mathfrak{r} : \mathcal{B} \times \mathbb{T} \rightarrow X$ and \mathfrak{p}_Σ is the canonical partial order state map for Σ (see [Definition 32](#)). We propose the following $\mathfrak{B}_{\mathfrak{p}_\Sigma}$ as such a bisimulation relation: let $\langle \chi, \langle f^{|t}, t \rangle \rangle \in \mathfrak{B}_{\mathfrak{p}_\Sigma}$ if and only if $\chi = \mathfrak{r}(g, t)$ for some $\langle g, t \rangle \in \mathcal{B} \times \mathbb{T}$ such that $g|_t = f^{|t}$. We note that this definition is well-posed, since the state map \mathfrak{r} is *past-induced*: in other words, **any** behaviors $g, h \in \mathcal{B}$ for which $g|_t = h|_t = f^{|t}$ will have the property that $\mathfrak{r}(g, t) = \mathfrak{r}(h, t)$.

We start by showing $\mathfrak{B}_{\mathfrak{p}_\Sigma}$ is a simulation relation indicating that $(\Sigma, \mathfrak{p}_\Sigma)$ simulates (Σ, \mathfrak{r}) . Thus, let $\langle \chi, \phi \rangle \in \mathfrak{B}_{\mathfrak{p}_\Sigma}$ where $\chi \in X$ and $\phi := \langle f^{|t}, t \rangle \in \Phi_\Sigma$. Then take any $w := \langle v_1, d_1 \rangle \in \mathcal{B}$ and $t_1 \in \mathbb{T}$ such that $\mathfrak{r}(w, t_1) = \chi$; also, take any $w_2 := \langle v_2, d_2 \rangle \in \mathcal{B}$ and $t_2 \in \mathbb{T}$ such that $\mathfrak{p}_\Sigma(w_2, t_2) = \langle w_2^{|t_2}, t_2 \rangle = \phi$. Of course $\mathfrak{p}_\Sigma(w_2, t_2) = \phi$ implies that $t = t_2$ and $w_2|_t = f^{|t}$. Thus, we have to show that

there exists a $d'_2 \in \pi_{\mathbb{D}}(\mathcal{B})$ such that properties 1 through 4 of [Definition 11](#) hold. However, since we are simply looking at two state maps on the *same* system, the natural choice for d'_2 is simply $d'_2 = d_2 \wedge_{t_1}^{t_2} d_1$, and hence property 3 is immediately satisfied. This makes $w'_2 = (v_2 \wedge_{t_1}^{t_2} v_1, d'_2) = w_2 \wedge_{t_1}^{t_2} w_1$; of course we know that w'_2 is in \mathcal{B} because $w_2|_{t_2} = f|^{t_2}$, so $\mathfrak{r}(w_1, t_1) = \mathfrak{r}(f|^{t_2}, t_2) = \mathfrak{r}(w_2, t_2)$ by the asserted membership in $\mathfrak{B}_{\mathfrak{p}_{\Sigma}}$. The extension as w_1 then follows from the state map property (see [Definition 7](#)). The preceding also trivially shows property 2 of [Definition 11](#) is satisfied. This, taken with the fact that the state map \mathfrak{r} is *Markovian*, directly implies that w'_2 also satisfies property 4 [Definition 11](#). This establishes that $\mathfrak{B}_{\mathfrak{p}_{\Sigma}}$ is a suitable simulation relation; that it is also a *simulation* follows from the fact that the state map \mathfrak{r} is defined for all $\mathcal{B} \times \mathbb{T}$.

Using almost exactly the same proof, it can be shown that $\mathfrak{B}_{\mathfrak{p}_{\Sigma}}$ establishes that (Σ, \mathfrak{r}) simulates $(\Sigma, \mathfrak{p}_{\Sigma})$. In particular, membership of a pair of states (χ, ϕ) in $\mathfrak{B}_{\mathfrak{p}_{\Sigma}}$ asserts that the trajectories emanating from both χ and ϕ are the same; the resulting states that those trajectories pass through will also be the same because the state map \mathfrak{r} is Markovian. \square

[Proposition 4](#) is of obvious relevance to the task at hand: recall that the state space of the state map \mathfrak{p}_{Σ} is exactly the set of nodes in the GST representation of Σ ! Thus, because bisimulation is an *equivalence* over behavioral systems, [Proposition 4](#) will allow us to chain a bisimulation relation between two GST representations of a behavioral system with any other bisimulation relation on one of the underlying behavioral systems. This is essentially the outline of the next section.

4.3 Comparison of Bisimulation Relations for Behavioral Dynamical Systems

In this section, we demonstrate that when behavioral systems are converted to GSTs according to [Definition 33](#), the notion of bisimulation between the trees matches the notion of bisimulation for the original behavioral systems. The central tools of this section will be [Proposition 4](#) and the fact that bisimulation over behavioral systems is an equivalence.

The comparison of bisimulation between GSTs and behavioral systems is mostly direct, but we will need the following definition, which establishes an additional property for bisimulation relations between GSTs generated according to [Definition 33](#).

Definition 34 (Time Shifting Property). *Let $\Sigma_1 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_1, \mathcal{B}_2)$ and $\Sigma_2 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_2, \mathcal{B}_2)$ be two dynamical systems which share the same external signal space \mathbb{V} . Further suppose that each dynamical system has an associated state map $\mathfrak{x}_i : \mathbb{V} \times \mathbb{D}_i \rightarrow X_i$. Furthermore, let \mathcal{G}_1 and \mathcal{G}_2 be the associated GSTs constructed according to [Definition 33](#). A bisimulation relation \sim between \mathcal{G}_1 and \mathcal{G}_2 is said to have the **time-shift property** if the following condition and its symmetric version hold:*

- *For each $\langle p_1, p_2 \rangle \in \sim$ and $q_1 \succeq_1 p_1$ there exists a $q_2 \in P_2$ and an order-preserving bijection $\lambda : (p_1, q_1] \rightarrow (p_2, q_2]$ such that both $\langle p', \lambda(p') \rangle \in \sim$ and $\pi_{\mathbb{T}}(p'_1) - \pi_{\mathbb{T}}(p_1) = \pi_{\mathbb{T}}(\lambda(p'_1)) - \pi_{\mathbb{T}}(p_2)$ hold for all $p'_1 \in (p_1, q_1]$.*

Lemma 1 (Behavioral systems with bisimilar state spaces have bisimilar GSTs).

Let $\Sigma_1 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_1, \mathcal{B}_2)$ and $\Sigma_2 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_2, \mathcal{B}_2)$ be two dynamical systems which share the same external signal space \mathbb{V} . Further suppose that each dynamical system has an associated state map $\mathfrak{r}_i : \mathbb{V} \times \mathbb{D}_i \rightarrow X_i$. If $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ have bisimilar state spaces according to [Definition 11](#), then the GSTs constructed from Σ_1 and Σ_2 (according to [Definition 33](#)) are bisimilar according to [Definition 30](#). Moreover, it is possible to choose a bisimulation relation with the time-shift property.

Proof. The proof consists mostly of applying [Proposition 4](#) and using the fact that bisimulation is an equivalence between behavioral systems [\[25\]](#). That is $(\Sigma_1, \mathfrak{p}_{\Sigma_1})$ is bisimilar to $(\Sigma_1, \mathfrak{r}_1)$ by [Proposition 4](#), and likewise, $(\Sigma_2, \mathfrak{r}_2)$ is bisimilar to $(\Sigma_2, \mathfrak{p}_{\Sigma_2})$. Since $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ are bisimilar by assumption, we conclude that $(\Sigma_1, \mathfrak{p}_{\Sigma_1})$ is bisimilar to $(\Sigma_2, \mathfrak{p}_{\Sigma_2})$. In the notation from [Section 2.2.2](#), we can express this chain of bisimulation equivalence as:

$$\Sigma_1 \underset{\mathfrak{p}_{\Sigma_1}}{\sim} \Sigma_1 \underset{\mathfrak{r}_1}{\sim} \Sigma_2 \underset{\mathfrak{r}_2}{\sim} \Sigma_2 \underset{\mathfrak{p}_{\Sigma_2}}{\sim} \Sigma_2. \quad (4.1)$$

Thus, the transitivity property of bisimulation as an equivalence relation [\[25\]](#) implies that there exists a relation $\underset{\mathfrak{p}_{\Sigma_1}}{\sim} \underset{\mathfrak{p}_{\Sigma_2}}{\sim} \subseteq \Phi_{\Sigma_1} \times \Phi_{\Sigma_2}$ for which $\Sigma_1 \underset{\mathfrak{p}_{\Sigma_1}}{\sim} \underset{\mathfrak{p}_{\Sigma_2}}{\sim} \Sigma_2$.

However, if $\mathcal{G}_1 = (P_1, \lesssim_1, p_{01}, \mathcal{L}_1)$ and $\mathcal{G}_2 = (P_2, \lesssim_2, p_{02}, \mathcal{L}_2)$ are the GSTs constructed from Σ_1 and Σ_2 according to [Definition 33](#), then $P_1 = \Phi_{\Sigma_1} \cup \{p_{01}\}$ and $P_2 = \Phi_{\Sigma_2} \cup \{p_{02}\}$. Thus, we claim that $\sim := \underset{\mathfrak{p}_{\Sigma_1}}{\sim} \underset{\mathfrak{p}_{\Sigma_2}}{\sim} \cup \{\langle p_{01}, p_{02} \rangle\}$ is a strong bisimulation between \mathcal{G}_1 and \mathcal{G}_2 (with the time shifting property).

It is immediately clear that this definition of \sim matches each node in P_1 to at least one node in P_2 and conversely: $\underset{\mathfrak{p}_{\Sigma_1}}{\sim} \underset{\mathfrak{p}_{\Sigma_2}}{\sim}$ is a bisimulation between $(\Sigma_1, \mathfrak{p}_{\Sigma_1})$ and $(\Sigma_2, \mathfrak{p}_{\Sigma_2})$, and so matches each state in Φ_{Σ_1} to at least one state in Φ_{Σ_2} and

conversely. Thus, it is only left to show that the trajectories emanating from related states satisfy the properties of [Definition 30](#).

We will show only that \mathcal{G}_2 strongly simulates \mathcal{G}_1 using \sim ; the proof in the other direction will be similar. To begin, suppose that $\langle\langle w_1^{t_1}, t_1 \rangle, \langle w_2^{t_2}, t_2 \rangle\rangle \in \mathfrak{p}_{\Sigma_1} \sim \mathfrak{p}_{\Sigma_2} \subset \sim$ and let w'_1 be any behavior such that $w_1^{t_1} = w'_1|_{t_1}$ to establish an maximal trajectory from $\langle w_1^{t_1}, t_1 \rangle$. We can obtain an appropriate maximal trajectory from \mathcal{G}_2 directly from [\(4.1\)](#). In particular, the middle bisimulation of [\(4.1\)](#) and the fact that w'_1 extends $w_1^{t_1}$ from t_1 together imply that $\mathfrak{x}_1(w'_1, t_1) \mathfrak{r}_1 \sim \mathfrak{r}_2 \mathfrak{x}_2(w'_2, t_2)$ for any w'_2 that agrees with $w_2^{t_2}$ up to time t_2 . If we let w'_1 be such a trajectory in Σ_1 , then the definition of bisimulation for behavioral systems ([Definition 11](#)) implies that there exists such a w'_2 in Σ_2 for which the states that w'_1 traverses after t_1 can be matched bijectively – through the time axis – to the states that w'_2 traverses after t_2 ; hence, if we call this time-based bijection λ , then λ in fact bijectively matches each $\langle w'_1, t \rangle \succeq \langle w_1^{t_1}, t_1 \rangle$ to $\langle w'_2, \lambda(t) \rangle \succeq \langle w_2^{t_2}, t_2 \rangle$ where $\mathfrak{x}_1(w'_1, t) \mathfrak{r}_1 \sim \mathfrak{r}_2 \mathfrak{x}_2(w'_2, \lambda(t))$. But employing [\(4.1\)](#) again, we see that the latter implies $\langle w'_1|_t, t \rangle \mathfrak{p}_{\Sigma_1} \sim \mathfrak{p}_{\Sigma_2} \langle w_2'^{\lambda(t)}, \lambda(t) \rangle$, and hence, $\langle w'_1|_t, t \rangle \sim \langle w_2'^{\lambda(t)}, \lambda(t) \rangle$. Of course the labels along these trajectories match by construction of the GSTs and the preservation of external variables in the behavior asserted by bisimulation between behavioral systems.

Consideration of trajectories emanating from $\langle p_{01}, p_{02} \rangle$ follows directly from the above, since all the immediate successors of p_{01} and p_{02} have time index $t = 0$. \square

Lemma 2 (Behavioral systems with bisimilar GSTs have bisimilar state spaces).

Let $\Sigma_1, \Sigma_2, \mathfrak{x}_1$ and \mathfrak{x}_2 be as in [Lemma 1](#). Suppose the GSTs constructed from Σ_1 and

Σ_2 (according to [Definition 33](#)) are bisimilar according to [Definition 30](#). Moreover, suppose that the bisimulation relation satisfies the time-shift property. Then (Σ_1, \mathbf{r}_1) and (Σ_2, \mathbf{r}_2) have bisimilar state spaces according to [Definition 11](#).

Proof. As before, let $\mathcal{G}_1 = (P_1, \lesssim_1, p_{01}, \mathcal{L}_1)$ and $\mathcal{G}_2 = (P_2, \lesssim_2, p_{02}, \mathcal{L}_2)$ be the GSTs constructed from Σ_1 and Σ_2 according to [Definition 33](#); furthermore, let $\mathcal{G}_1 \sim \mathcal{G}_2$ be the claimed bisimulation relation between them. The proof then follows from the observation that \sim effectively defines a bisimulation relation between the behavioral systems $(\Sigma_1, \mathbf{p}_{\Sigma_1})$ and $(\Sigma_2, \mathbf{p}_{\Sigma_2})$; the existence of this bisimulation follows from an argument that is similar to the one in the proof of [Lemma 1](#), only in the other direction. The time-shifting property of the bisimulation is necessary here to ensure that the GST bisimulation relation translates to a behavioral-system bisimulation.

Thus, the assumptions of the Lemma – combined with [Proposition 4](#) – effectively create the following bisimulation chain (compare to [\(4.1\)](#)):

$$\Sigma_1 \underset{\mathbf{r}_1}{\sim} \underset{\mathbf{p}_{\Sigma_1}}{\Sigma_1} \underset{\mathbf{p}_{\Sigma_1}}{\sim} \underset{\mathbf{p}_{\Sigma_2}}{\Sigma_2} \underset{\mathbf{r}_2}{\sim} \Sigma_2. \quad (4.2)$$

The equivalence property of bisimulation for behavioral systems completes the proof by asserting the existence of a bisimulation $\underset{\mathbf{r}_1}{\sim} \underset{\mathbf{r}_2}$. \square

Theorem 7. *Let $\Sigma_1 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_1, \mathcal{B}_2)$ and $\Sigma_2 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_2, \mathcal{B}_2)$ be two dynamical systems which share the same external signal space \mathbb{V} . Further suppose that each dynamical system has an associated state map $\mathbf{r}_i : \mathbb{V} \times \mathbb{D}_i \rightarrow X_i$, which is both past induced and Markovian. Then (Σ_1, \mathbf{r}_1) and (Σ_2, \mathbf{r}_2) are bisimilar in the sense of [Definition 11](#) if and only if the GSTs constructed according to [Definition 33](#) from*

the state systems are strongly bisimilar and are related by a bisimulation relation with the time-shift property.

Proof. [Lemma 1](#) and [Lemma 2](#) establish the claimed necessary and sufficient conditions for the conclusion. \square

It is important to note that the time-shifting property in [Definition 34](#) really is necessary to prove [Lemma 2](#). The following example indicates how it is possible to have bisimilarity between GSTs constructed from behavioral systems, but not between the original behavioral systems.

Example 1. Let $\Sigma_1 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_1, \mathcal{B}_1)$ and $\Sigma_2 = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_2, \mathcal{B}_2)$ be two behavioral systems such that $\mathbb{T} = \mathbb{R}^{\geq 0}$, $\mathbb{V} = \mathbb{R}$ and $\mathbb{D}_1 = \mathbb{D}_2 = \{0\}$. Then let $\mathcal{B}_1 = \{w_1\}$ where $w_1 = \langle v_1, d_1 \rangle$ with $v_1 : t \mapsto t^2$ and $d_1 : t \mapsto 0$; similarly, let $\mathcal{B}_2 = \{w_2\}$ where $w_2 = \langle v_2, d_2 \rangle$ with $v_2 : t \mapsto (t/2)^2$ and $d_2 : t \mapsto 0$. Finally, let $\mathfrak{x}_i : \mathcal{B}_i \times \mathbb{T} \rightarrow X_i$, $i = 1, 2$ be state maps such that $\mathfrak{x}_i : \langle w_i, t \rangle \mapsto 0$. According to the procedure for constructing GSTs from behavioral systems, $\mathcal{G}_1 = (P_1, p_{01}, \lesssim_1, \mathcal{L}_1)$ and $\mathcal{G}_2 = (P_2, p_{02}, \lesssim_2, \mathcal{L}_2)$ are trees with a only a single branch each. That is $P_i \setminus \{p_{0i}\} = \{w_i|_\tau : \tau \in \mathbb{R}^{\geq 0}\}$, $i = 1, 2$.

We claim that

$$\{\langle p_{01}, p_{02} \rangle\} \cup \{\langle \langle w_1|_\tau, \tau \rangle, \langle w_2|_{2\tau}, 2\tau \rangle \rangle : \tau \in \mathbb{T}\} = \sim \quad (4.3)$$

is a bisimulation relation between \mathcal{G}_1 and \mathcal{G}_2 according to [Definition 30](#). This is easy to establish because the trees have only one branch and we have used the time axis to suggest which order-preserving bijections to use. Moreover, this is the largest bisimulation relation between \mathcal{G}_1 and \mathcal{G}_2 ; this is because $\mathcal{L}_1(\langle w_1|_\tau, \tau \rangle) = \tau^2$ and

$\mathcal{L}_2(\langle w_2|_\tau, \tau \rangle) = (\tau/2)^2$, so (4.3) is the only way to pair nodes so that they satisfy the label matching property of Definition 30.

Importantly, \sim from Example 1 does not have the time-shifting property. Indeed, let $\langle p_1, p_2 \rangle := \langle \langle w_1|_{t_1}, t_1 \rangle, \langle w_2|_{2t_1}, 2t_1 \rangle \rangle \in \sim$ and let $q_1 := \langle w_1|_{t_1+1}, t_1 + 1 \rangle \lesssim_1 \langle w_1|_{t_1}, t_1 \rangle$. Then it is obvious that any appropriate order-preserving bijection must have $\lambda(p_1) = \langle w_2|_{2(t_1+1)}, 2(t_1 + 1) \rangle =: q_2$, since $\langle q_1, q_2 \rangle$ is the only pair in \sim such that $\pi_1 \langle q_1, q_2 \rangle = q_1$. However, this implies that $\pi_{\mathbb{T}}(q_1) - \pi_{\mathbb{T}}(p_1) = 1$ and $\pi_{\mathbb{T}}(\lambda(q_1)) - \pi_{\mathbb{T}}(p_2) = 2(t_1 + 1) - 2t_1 = 2$. As \sim is the largest bisimulation relation between \mathcal{G}_1 and \mathcal{G}_2 , there is no bisimulation relation between those trees with the time-shifting property.

Finally, we note that $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ are not bisimilar in the sense of Definition 11. Let $\phi_1 = \mathfrak{p}_{\Sigma_1}(w_1, t_1) = \langle w_1|_{t_1}, t_1 \rangle$ be any element in the state space of $(\Sigma_1, \mathfrak{p}_{\Sigma_1})$ and let $\phi_2 = \mathfrak{f}_{\Sigma_2}(w_2, t_2) = \langle w_2|_{t_2}, t_2 \rangle$ be any element in the state space of $(\Sigma_1, \mathfrak{p}_{\Sigma_2})$. We claim that $\langle \phi_1, \phi_2 \rangle$ cannot be in a simulation relation. In particular, observe that $v_2 \wedge_{t_1}^{t_2} v_1 = (t/2)^2 \wedge_{t_1}^{t_2} t^2$ cannot be matched to a valid behavior in Σ_2 , hence property 1 of Definition 11 cannot possibly be satisfied. Thus, there is no simulation relation which contains $\langle \phi_1, \phi_2 \rangle$. Since we chose ϕ_1 and ϕ_2 arbitrarily, we conclude that $(\Sigma_1, \mathfrak{p}_{\Sigma_1})$ and $(\Sigma_2, \mathfrak{p}_{\Sigma_2})$ cannot possibly be bisimilar, and hence, neither can $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ by Lemma 1.

Chapter 5: GSTs and Modal Logic

5.1 Introduction

One important and well known tool in the formal description and verification of *discrete* processes is the simple modal logic known as Hennessy-Milner Logic (HML) [18] (HML, modal logic and associated notions are summarized in [Section 2.3](#)). This is in large part because Hennessy and Milner noticed a relationship between HML and bisimulation in the context of image-finite processes: that is two image-finite processes are bisimilar if and only if they satisfy the same HML formulas [18]. However, it's possible to study broader classes of processes for which modal equivalence – that is satisfying the same modal formulas – implies bisimulation; such classes of processes are usually called Hennessy-Milner (HM) classes, although there are different variations of the main idea. In this chapter, we will concern ourselves with a notion that we call Visser-Hollenberg Hennessy-Milner (VHHM) classes after the work in [27]. Specifically, a VHHM class of Kripke Structures (KSs) is a class of KSs with the following property: whenever two worlds (or states) from any of the constituent KSs satisfy all the same modal formulas, they are necessarily bisimilar to each other [27]; *this specifically includes the case when both nodes come from the same tree*. Importantly, however, [27] proceeds to define and characterize *max-*

imal VHHM classes of KSs. Maximal VHHM classes are particularly interesting from the point of view of system composition, because that maximality *circumscribes the operations one can perform* and still be able to distinguish bisimulation non-equivalence with modal formulas. In other words, composition operations that preserve membership in a VHHM class will produce KSs that remain distinguishable by modal formulas if they become non-bisimilar; by contrast, composition operations that “spill out” of a *maximal* VHHM class obviously lose that property. [27] considered some basic process-algebraic-like operations with respect to the VHHM class of *m-saturated* KSs.

A result like this for *continuous and hybrid systems models* would have obvious implications for the design and diagnosability of CPSs, particularly in the context of systems whose behavior is difficult to reason about. GSTs are a natural substrate from which to begin such a generalization: in [Section 5.2](#), we begin this project by first proposing a Generalized Hennessy-Milner logic (GHML) – that is a modal logic like Hennessy-Milner Logic (HML) but with semantics specific to GSTs – and consider its relationship to maximal VHHM classes of GSTs with respect to weak bisimulation [19]. As in the discrete case, then, whenever CPS systems can be modeled using composition operators that are consistent with a maximal VHHM class *of GSTs*, the design-test-redesign iteration benefits from valuable additional information: a design or implementation that fails to meet a specification (relative to bisimulation) will have a *GHML witness* to that deviation. As GHML, like HML, is a quite simple logic, a GHML witness offers an easy-to-interpret description of how the specification is violated. Consider for example the value of such a witness in

the context of a modern networked control system that combines software, network effects and real-world dynamics in a very complicated way: in such a circumstance, a GHML formula could explain a specification violation in terms of network delays that affect a feedback loop around physical dynamics, ultimately leading to instability. Likewise, modern machine learning algorithms are increasingly implemented in real-world-connected CPSs such as autonomous vehicles, yet those algorithms produce classifiers whose decisions inherently lack “explainability” on their own, let alone as part of a larger, more complicated system. In such a situation, a GHML formula could clarify how undesirable decisions from a road-sign classifier, say, lead to unexpected behavior from the physical dynamics of the car as whole.

However, to reach this end goal for GSTs, there is an important additional consideration relative to the discrete case: GSTs have the flexibility to capture all sorts of continuous and hybrid behavior, so it is not immediately clear what sorts of practical “dynamical” behaviors are associated with any particular VHHM class, especially as this involves translating “timed” LTI systems into the domain of GHML and weak bisimulation, which *lack a comparable, inherent notion of “clock” time*. That is: describing a VHHM class of surrogate KSs – and hence a VHHM class of GSTs – says nothing about the sorts of practical dynamical-system models that will end up in that class; this is in contrast to the discrete case where the characterization of VHHM classes is in terms of the very models of interest, KSs. Put another way, there are likely to be some subclasses of GSTs that are of significantly more practical importance than others, and their relationship to surrogate KSs and maximal VHHM classes is an important consideration for practical reasons at least.

Specifically, this immediately begs the basic question of whether *any* reasonable classes of dynamical system models fit inside a *single* maximal VHHM class of GSTs to begin with. And if a particular class of dynamical systems is not encompassed by a single VHHM class, then we can simply ask what further specifications are needed to select a sub-class that *is* so encompassed. These (and other similar) questions get at the heart of *which* continuous-time dynamical behaviors can be captured by the expressiveness of the clock-time-unaware weak bisimulation and GHML. Of course, these questions necessarily address as a special case the continuous behaviors that are captured by *m-saturated* surrogate KSs composition directed questions, and as such, must form the basis for compositional results of the sort alluded to above.

The remainder of this chapter, then, is devoted to specifying a *class* of finite-dimensional, Linear Time-Invariant (LTI) control systems, along with a suitably constrained class of inputs, that have this property. Unfortunately, we cannot consider either arbitrary LTI systems or arbitrary inputs: this is because even the relative specificity of LTI dynamics is not enough to result in surrogate KSs that have some manageable structure (viz. nondeterminism).

Indeed, this project first involves considering the structure of the surrogate KSs obtained from the GST encodings of such systems. Importantly, however, weak bisimulation essentially ignores real (clock) time information, so the surrogate KSs obtained from *deterministic* LTI systems will have interesting and non-trivial *nondeterminism*, **despite the above mentioned restrictions**. The conversion of LTI systems to GSTs is the topic of [Section 5.3](#); the structure of the resultant nondeterminism is characterized in [Section 5.4](#). As mentioned before, we will introduce

restrictions on the LTI conversion process so as to end up with specific and manageable nondeterminism in the resultant GSTs and their surrogate KSs. Of course the nature of this nondeterminism has implications for the VHHM classes to which such LTI systems may belong; this is addressed in [Section 5.5](#), which connects the prequel to GHML and modal logic specifically.

5.2 Generalized Hennessy-Milner Logic (GHML)

5.2.1 GHML for GSTs: Syntax and Semantics

Since KSs and STs have well-defined “outgoing edges” from their states, the label inside the HML diamond modality can be naturally matched to those edges (or transitions). As noted before, though, nodes in GSTs generally lack immediate successors. Thus, the project of generalizing HML is essentially one of generalizing the label in the diamond modality, so that it can be appropriately matched to the semantics of GST trajectories (see [Definition 27](#)). The proposed mechanism for this is to define the logic in terms of a linear, total order, called a *domain of modalities*, that can then be compared to GST trajectories via order equivalences [\[19\]](#).

Definition 35 (Domain of modalities [\[19\]](#)). *A domain of modalities is a totally ordered set, $(\mathcal{I}, \preceq_{\mathcal{I}})$, together with a set of labels L .*

Of course the bounded GST trajectories used in the definition of weak bisimulation have both a “starting” point and an “ending” point, so we should confine ourselves to such subsets of a domain of modalities.

Definition 36 (Left-open subset [19]). *A subset I of a totally ordered set \mathcal{I} is **left open (and right-closed)** if it satisfies the following two conditions:*

- *it has both a least upper bound (LUB) and a greatest lower bound (GLB) in \mathcal{I} ; and*
- *it contains its LUB but doesn't contain its GLB.*

Importantly, the preceding definition allows left-open subsets to have “gaps” in them. This is meant to facilitate the treatment of hybrid systems with more complicated time constructs; see for example [33].

Left-open subsets capture the essence of GST trajectories, but they don't capture the external interactions (labels) associated with those trajectories. Hence, the following definition, which forms the basis for our generalized diamond modality.

Definition 37 (Modal execution [19]). *Let (\mathcal{I}, L) be a domain of modalities. A **modal execution** is a map from a left-open subset of \mathcal{I} to the set of labels, L . The set of modal executions over (\mathcal{I}, L) will be denoted $\mathcal{M}(\mathcal{I}, L)$.*

Of course different modal executions may represent essentially the same “sequence” of external interactions in much the same way order-equivalent trajectories do. Hence we extend the idea of order equivalence to modal executions in the sequel.

Definition 38 (Order Equivalent Modal Executions [19]). *Let $E_1 : I_1 \rightarrow L$ and $E_2 : I_2 \rightarrow L$ be two modal executions from a domain of modalities (\mathcal{I}, L) . We say that E_1 is **order equivalent** to E_2 or $E_1 \overset{o.e.}{\sim} E_2$ if there exists an order preserving bijection $\lambda : I_1 \rightarrow I_2$ such that $E_1(i) = E_2(\lambda(i))$ for all $i \in I_1$. The collection of equivalence class of $\mathcal{M}(\mathcal{I}, L)$ under $\overset{o.e.}{\sim}$ will be denoted $|\mathcal{M}(\mathcal{I}, L)|$.*

To maintain consistency with the semantics of GSTs, we will have to impose the following additional condition on domains of modalities; *we will assume it applies to all domains of modalities considered henceforth.*

Definition 39 (Closed under left-open concatenation [19]). *We say that a totally ordered set \mathcal{I} is **closed under left-open concatenation** if for any two left-open subsets $I_1, I_2 \subseteq \mathcal{I}$, there exists another left-open set $I_3 \subseteq \mathcal{I}$ such that $I_3 \stackrel{o.e}{\simeq} I_1; I_2$ where $I_1; I_2 \triangleq (\{1\} \times I_1) \cup (\{2\} \times I_2)$ is totally ordered under the lexicographic ordering. A totally ordered set \mathcal{I} that is closed under left-open concatenation will be denoted $\bar{\mathcal{I}}$.*

We are now in a position to write down the syntax of GHML in terms of $|\mathcal{M}(\bar{\mathcal{I}}, L)|$; the use of such equivalence classes will be important in the subsequent development of VHHM classes of GSTs.

Definition 40 (Set of Generalized HML (GHML) formulas [19]). *Given a domain of modalities $(\bar{\mathcal{I}}, L)$ that is closed under left-open concatenation, the set of **Generalized HML (GHML)** formulas is the set of formulas, $\Phi_{GHML}(\bar{\mathcal{I}}, L)$, inductively defined according to the following rules:*

$$\varphi := \top \quad | \quad \neg\varphi \quad | \quad \varphi_1 \wedge \varphi_2 \quad | \quad \langle\langle |E| \rangle\rangle\varphi \quad (5.1)$$

where $|E| \in |\mathcal{M}(\bar{\mathcal{I}}, L)|$ is an equivalence class of modal executions over the domain of modalities $(\bar{\mathcal{I}}, L)$.

The use of double angle brackets in [Definition 40](#) may seem redundant, but it will be an important distinction in the future, since we will eventually consider *ordinary HML modalities* with the same sorts of labels; see the sequel, [Section 5.2.2](#).

Finally, the semantics of GHML (over GSTs) is defined in more or less the expected way: that is in terms of order equivalences between trajectories in the GST and modal executions over a given domain of modalities. The details are available in [19].

5.2.2 VHHM Classes of GSTs

The important realization in [19] is that, as far as weak bisimulation and GHML are concerned, we can work with certain discrete structures *instead of* GSTs. These structures were dubbed *surrogate Kripke structures*, and they are the basis for the main results in [19].

Definition 41 (Captured by a Domain of modalities [19]). *Let \mathcal{U} be a set of GSTs. We say that \mathcal{U} is **captured** by a domain of modalities $(\bar{\mathcal{L}}, L)$ if every trajectory from every GST in \mathcal{U} is order equivalent to some modal execution over $(\bar{\mathcal{L}}, L)$.*

Definition 42 (Surrogate Kripke Structure [19]). *Let \mathcal{U} be a set of GSTs that is captured by a domain of modalities $(\bar{\mathcal{L}}, L)$. For any GST $G = (P, \preceq_P, p_0, \mathcal{L})$ in \mathcal{U} , we define a **surrogate Kripke structure**, $\mathbf{G} = (P, \{R_{|E|}^G \subseteq P \times P : |E| \in \mathcal{M}(\bar{\mathcal{L}}, L)\}, V)$, as follows:*

- *the set of states is P ; and*
- *$p_1 \xrightarrow{|E|} p_2$ – i.e. $p_1 R_{|E|}^G p_2$ – if and only if $p_1 \preceq_P p_2$ and $(p_1, p_2]$ is order equivalent to an element of $|E|$; and*
- *$V : \Theta \rightarrow \{P\}$ indicates all propositional variables are true in all states in P .*

The importance of surrogate KSs is revealed by the following two theorems, both with rather straightforward proofs [19].

Theorem 8 (Relating GHML formulas on G to HML formulas on \mathbf{G} [19]). *Let $(\bar{\mathcal{I}}, L)$ be a domain of modalities, and let \mathcal{U} be a set of GSTs captured by $(\bar{\mathcal{I}}, L)$. Furthermore, consider HML over the set of labels given by $|\mathcal{M}(\bar{\mathcal{I}}, L)|$. Then for every $G = (P, \preceq_P, p_0, \mathcal{L}) \in \mathcal{U}$,*

1. *for all $\varphi \in \Phi_{GHML}(\bar{\mathcal{I}}, L)$, $G \models \varphi \Rightarrow p_0 \models \varphi_{\langle \rangle}$ and*
2. *for all $\phi \in \Phi_{HML}(|\mathcal{M}(\bar{\mathcal{I}}, L)|)$, $p_0 \models \phi \Rightarrow G \models \phi_{\langle \langle \rangle \rangle}$.*

The notation $\varphi_{\langle \rangle}$ indicates that the GHML formula φ is converted to an HML formula by replacing each $\langle \langle |E| \rangle \rangle$ modality with the corresponding HML modality $\langle |E| \rangle$. $\phi_{\langle \langle \rangle \rangle}$ indicates an analogous conversion from an HML formula to a GHML formula.

Theorem 9 (Weak bisimulation between GSTs and bisimulation between surrogates [19]). *Let \mathcal{U} and $(\bar{\mathcal{I}}, L)$ be as in [Theorem 8](#). Furthermore, let $G_1 = (P, \preceq_P, p_0, \mathcal{L}_P)$ and $G_2 = (Q, \preceq_Q, q_0, \mathcal{L}_Q)$ be two GSTs in \mathcal{U} . Then*

$$G_1 \simeq_w G_2 \iff p_0 \simeq q_0. \quad (5.2)$$

where the bisimulation $p_0 \simeq q_0$ is taken in the context of the surrogate Kripke structures \mathbf{G}_1 and \mathbf{G}_2 .

Thus, as far as GHML and weak bisimulation are concerned, we may as well work with surrogate KSs instead of GSTs!

This connection to KSs, though, leads to another important consequence: it allows us to define and constrain maximal VHHM classes of GSTs in terms maximal

VHHM classes of KSs. We begin with the following predictable definition of VHHM classes for GSTs.

Definition 43 (VHHM class for GSTs [19]). *Let \mathcal{U} be a set of GSTs, and let $(\bar{\mathcal{I}}, L)$ be a domain of modalities that captures \mathcal{U} , so that \approx_{GHML} is interpreted with respect to $\Phi_{GHML-\Theta}(\bar{\mathcal{I}}, L)$. Then we say that a subset $\mathfrak{h} \subseteq \mathcal{U}$ **satisfies the VHHM property for GSTs** if for any two sub-GSTs $G_1|_p$ and $G_2|_q$ from the set \mathfrak{h} (possibly with $G_1 = G_2$),*

$$G_1|_p \overset{\text{tr}}{\simeq}_w G_2|_q \iff G_1|_p \approx_{GHML} G_2|_q. \quad (5.3)$$

Note, however, that we can go from modal equivalence for two nodes in a GST to bisimulation for those two nodes by *going through the surrogate KSs for the respective GSTs*. Since the GST semantics imposes certain characteristics on the transition structure of the surrogate KSs, we may then further constrain maximal VHHM classes of GSTs in terms of maximal VHHM classes of KSs that contain as theorems the appropriate schemata.

Theorem 10 (Maximal VHHM classes for GSTs and refined Henkin-like models).

Let \mathcal{U} and $(\bar{\mathcal{I}}, L)$ be as in [Definition 43](#), and let $\mathfrak{h} \subseteq \mathcal{U}$ be a VHHM class of GSTs.

Furthermore, let Δ be the smallest normal logic that contains all of the following:

- *the propositional variables Θ ;*
- *$\forall |E_1|, |E_2| \in |\mathcal{M}(\bar{\mathcal{I}}, L)|$, the schema $\langle |E_1| \rangle \langle |E_2| \rangle \varphi \rightarrow \langle |E_{1;2}| \rangle \varphi$; and*
- *$\forall |E|, |E_1|, |E_2| \in |\mathcal{M}(\bar{\mathcal{I}}, L)|$ such that there is an order equivalence $\lambda : I_1; I_2 \rightarrow \text{dom}(E)$ with $E \circ \lambda(1, \cdot) \in |E_1|$ and $E \circ \lambda(2, \cdot) \in |E_2|$, the schema $\langle |E| \rangle \varphi \rightarrow$*

$$\langle |E_1| \rangle \langle |E_2| \rangle \varphi.$$

Then $\{\mathbf{G} : G \in \mathfrak{h}\} \subseteq \mathbf{BS}(\mathbf{HC}^\Delta)$ for some Henkin-like model \mathbf{HC}^Δ that preserves the first-order transition-relation properties imposed on \mathbf{C}^Δ by the schemata above. Furthermore, if there is a set $\mathfrak{h}' \subseteq \mathcal{U}$ such that $\mathfrak{h} \subseteq \mathfrak{h}'$ and $\{\mathbf{G} : G \in \mathfrak{h}'\} \subseteq \mathbf{BS}(\mathbf{HC}^\Delta)$, then \mathfrak{h}' is a VHHM class of GSTs.

Given the relationship between surrogate KSs and both weak bisimulation and GHML (Theorems [Theorem 9](#) and [Theorem 8](#), respectively), [Theorem 10](#) seems to be a rather straightforward bound on the “size” of VHHM classes of GSTs. However, this superficial simplicity belies two important subtleties that ultimately have to do with the fact that VHHM classes are characterized by Henkin-like models, which in turn are obtained by *removing* transitions from the Henkin model.

First, the stipulation that \mathbf{HC}^Δ preserve the appropriate first-order transition-relation properties is essential. \mathbf{C}^Δ necessarily satisfies the transitivity and weak-density properties imposed by the schemata in [Theorem 10](#), essentially because the canonical model contains enough transitions to ensure that this is the case; see [\[26\]](#). However, after *removing* transitions from the canonical model to get an arbitrary Henkin-like model, these first-order properties need not hold! Of course we know that the surrogate KSs generated from GSTs *must* satisfy these first-order properties by the nature of their semantics, so we should confine ourselves to such Henkin-like models in order to ensure the bisimulation inclusion (operator \mathbf{B}) captures the surrogate KSs.

Second, using *equivalence classes* of modal executions to label GHML modal-

ities is also important relative to the characterization of VHHM classes in terms of Henkin-like models. Had we not chosen to use such equivalence classes, then we could generate Henkin-like models by removing transitions reflecting one modal execution but not some other *order-equivalent* modal executions! Such a situation would reflect a deviation from the semantics of GHML with respect to GSTs, for it would be possible to assert the existence of a trajectory through one diamond modality and simultaneously the *lack* of such a trajectory through another, simply by choosing modal executions with different domains!

5.3 Linear Time-Invariant Systems as GSTs

In this section, the objective is to represent the ubiquitous finite-dimensional LTI control system models as GSTs. This section will be a middle ground between background material and novel content. On the one hand, we will follow a well-established approach to obtain GST representations from LTI control systems – see [15, 17] and [Chapter 4](#); on the other hand, we introduce here some novel but non-trivial constraints on the LTI systems and their admissible input functions that will prove relevant in subsequent sections. **The universal aim of these restrictions will be to alter the nondeterminism that appears in the resultant surrogate KSs, so that the resulting surrogate KSs have a manageable (but still nontrivial) structure.** The consequences of these assumptions will extend through the remainder of this work.

5.3.1 LTI Systems

In this chapter, we will use the terminology “LTI system” as a short hand for *finite-dimensional* LTI systems that admit a state-space representation; we will further confine ourselves to LTI systems that evolve over a real-valued time axis.

LTI systems of this form can be described in terms of three quantities: the *state*, the *input* and the *output*. The *state* of such a system at any particular time will be given by a vector in \mathbb{R}^n , and that state will *evolve* (in time) according to a system of first-order ordinary differential equations (ODEs) that are written in terms of the component-wise derivatives of the state evolution function. The *inputs* to the system at a particular time will be given by a vector in \mathbb{R}^m , and those inputs as a function of time affect the evolution of the state through the aforementioned ODEs. The *outputs* of the system at a particular time are given by a vector in \mathbb{R}^ℓ , and they will be specified as an appropriate function of only the *current* value of the state. Thus, for our purposes an LTI system will be described by a set of equations like this:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{5.4}$$

where A , B and C are constant, real-valued matrices with dimensions $n \times n$, $n \times m$ and $\ell \times n$, respectively. In keeping with the above, $x(\cdot)$ is called the *state trajectory* (with codomain \mathbb{R}^n), $u(\cdot)$ is called the *input trajectory* (with codomain \mathbb{R}^m), and $y(\cdot)$ is called the *output trajectory* (with codomain \mathbb{R}^ℓ).

As a matter of future convenience, we will impose the following assumption on the structure of the LTI systems that we will consider:

(A1): The matrix CB is full column rank.

The reason for this assumption will become clear later, in [Section 5.4](#); however, like all of the assumptions we make, it is directed at eliminating a particular source of non-determinism in the surrogate KS.

5.3.2 LTI Systems as GSTs

The most natural way of converting an LTI system like (5.4) into a GST involves regarding such a system as a set of functions (or trajectories): one vector-valued function each for the state, the input and the output. This trajectory view of dynamical systems was an insight that was pioneered by [13] – though it was closely related to a similar ideas for discrete systems – and it formed the basis of the GST constructions in both [15] and [17]. We will essentially replicate that type of construction below.

5.3.2.1 Admissible Input Functions

As we are considering LTI systems in terms of their trajectories, it is first necessary to specify exactly what *input* trajectories we will permit. We will not, however, consider the same broad class of input functions described in the above references. The restrictions that we will impose on the admissible input functions directly service the goal of minimizing the non-determinism in the GST conversion; this will become clear in [Section 5.4](#).

First and foremost, we will confine ourselves to inputs u that are piecewise

analytic; for real-valued functions, we mean by this that about each point in its domain, u is (component-wise) representable by a convergent power series (with a non-trivial region of convergence); see also [34]. This forms the first *rather strong* assumption that we will make on the input functions:

(A2): $u : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^m$ is piece-wise analytic and continuous from the right (also with well-defined limits from the left).

Note the assertion that an input signal starts at time $t = 0$; however, it is the analyticity assumption that is most important. Indeed, the latter leads to the immensely useful consequence in the following proposition.

Proposition 5. *For any (piece-wise) analytic input, u (with an appropriate region of convergence), both the resultant state trajectory and output trajectory of the LTI system (5.4) are also (piece-wise) analytic.*

Proof. Analytic functions may be added, multiplied, integrated and differentiated on their regions of convergence (or the intersections thereof) to obtain further analytic functions. Since

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \quad (5.5)$$

where e^{At} is the usual (matrix) exponential power series, both x and y are specified by such operations on (piece-wise) analytic functions; hence, they are also (piece-wise) analytic. □

This assumption will help eliminate non-determinism in the surrogate KS through a very special property of non-constant analytic functions: such functions

are known to have *isolated zeros* – that is each zero of such a function is contained in an open interval that contains no *other* zeros. The usage of this fact will be made clear in [Section 5.4](#).

We will now introduce a second and final restriction on the allowable input functions, u , that concerns how such inputs *interact* with their associated LTI systems response. This interaction will be defined in terms of the following property of system states.

Definition 44 (Output Invariant). *We say that two states x_1 and x_2 (from a common LTI system) are output invariant if*

$$\forall t \geq 0 . Ce^{At}(x_1 - x_2) = 0. \quad (5.6)$$

Equivalently, $x_1 - x_2 \in \mathcal{N}(\mathcal{O})$, where $\mathcal{N}(\mathcal{O})$ is the null space of the usual $n \cdot \ell \times n$ observability matrix matrix, \mathcal{O} . The matrix \mathcal{O} is defined as usual according to:

$$\mathcal{O} = [C; CA; CA^2; CA^{(n-1)}]. \quad (5.7)$$

Remark 1. *At **no point** will we assume that \mathcal{O} is full rank: i.e. we consider even systems that are **not observable!** Note: the system may still be unobservable, even though **(A2)** asserts that CB is full rank!*

Note that the above definition of output invariance defines an equivalence relation on states in \mathbb{R}^n : moreover, the set of output invariant states with respect to any particular $x \in \mathbb{R}^n$ is given by $x + \mathcal{N}(\mathcal{O})$. In particular, any such equivalence class of output invariant states can be identified with a linear space, and such equivalence classes modulo a subspace are easily given a vector space structure themselves; this

is a common construction that is usually described as a *quotient* vector space. For convenience, we denote the quotient space $\mathbb{R}^n/\mathcal{N}(\mathcal{O})$ by $\mathcal{N}(\mathcal{O})^\perp$ and the equivalence class associated with a particular vector x by $[x]$. The assumption we make on the inputs is then described in terms of an a priori specified function that will ultimately constrain the (analytic) input segments allowed for any given value of state and input to the LTI system.

Definition 45 (Branching Choice Function). *A branching choice function for an LTI system (A, B, C) is a (pair of) (total) functions $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_{\neq 0})$, defined over a partition of $\mathbf{D} = \{([x], u) \in \mathcal{N}(\mathcal{O})^\perp \times \mathbb{R}^m : CA[x] + CBu = 0\}$:*

$$\mathcal{F}_0 : \{([x], u) \in \mathbf{D} : CA^2[x] + CABu = 0\} \rightarrow \mathbb{R}^m \setminus \{0\} \quad (5.8)$$

$$\mathcal{F}_{\neq 0} : \{([x], u) \in \mathbf{D} : CA^2[x] + CABu \neq 0\} \rightarrow (\mathbb{R}^m)^\mathbb{N}. \quad (5.9)$$

Now, we may specify the relevant assumption on the inputs in terms of an *a priori fixed* branching choice function, \mathcal{F} :

(A3): Let \mathcal{F} be our given, fixed branching choice function, and let x and y be the state and output trajectories for some (admissible) input trajectory, u . For any $t \geq 0$, if $([x(t)], u(t)) \in \mathbf{D}$ and y and u are *not constant* on some interval $[t, t + \epsilon)$, then the following conditions must be satisfied.

- If $([x(t)], u(t))$ is in the domain of \mathcal{F}_0 , then

$$* \dot{u}(t) = \mathcal{F}_0([x(t)], u(t)).$$

- If $([x(t)], u(t))$ is in the domain of $\mathcal{F}_{\neq 0}$, then

- * if $\dot{u}(t) \neq 0$, then for some component of the output, $1 \leq k \leq \ell$

$$\begin{aligned} \text{sign}([CA^2[x(t)] + CABu(t) + CB\dot{u}(t)]_k) \\ = \text{sign}([CA^2[x(t)] + CABu(t)]_k) \end{aligned} \quad (5.10)$$

and for all integers $j \geq 3$,

$$j \cdot [[CA^2[x(t)] + CABu(t) + CB\dot{u}(t)]_k - [CB\dot{u}(t)]_k] \neq 0; \quad (5.11)$$

- * if $\dot{u}(t) = 0$, then let $\{U_k\} = \mathcal{F}_{\neq 0}([x(t)], u(t))$ and require

$$u(\tau) = u(t) + \sum_{j=2}^{\infty} U_{j-2}(\tau - t)^j \text{ for } \tau \in [t, t + \epsilon']; \quad (5.12)$$

that is u has the particular form of the power series given above in some neighborhood of t .

It is because of this interpretation that we call \mathcal{F} a branching choice function: it

constrains the allowable input functions u (through their first derivatives or the power series above) at any particular state.

These particular constraints are rather unintuitive, but it will become clear subsequently how compliance with these constraints will reduce the nondeterminism in LTI-derived GSTs. For now, though, it is important to recognize that, through the use of a branching choice function, these constraints are imposed identically throughout the evolution of the LTI system, no matter when or how particular states/inputs are reached. On the other hand, **(A3)** does *not* require the same inputs to be enabled in all such $([x], u)$ pairs! Note also that \mathcal{F}_0 merely constrains $\dot{u}(t)$ to be a fixed, common value any time the system passes through a particular $([x], u)$ in the domain of \mathcal{F}_0 ; the *higher* derivatives of u are *unconstrained* by this requirement. Likewise, the constraints on $([x], u)$ pairs in the domain of $\mathcal{F}_{\neq 0}$ generally leave the higher derivatives of u free, except in the final case where $\dot{u}(t) = 0$; there *all of the derivatives* of u are essentially fixed through the power series expression as described above.

There is an important consideration relative to assumption **(A3)** that must be addressed, though, since **(A3)** involves the interaction between inputs, states and outputs. In particular, one must take care to be ensure that there are is a reasonable – and *consistent* – collection of system trajectories that satisfies **(A3)**! We will address this issue in the next subsection, where we take up the question of specifying the set of behaviors for an LTI system.

5.3.2.2 Construction of the GSTs

The assumptions in the previous subsection specify the set of input functions we consider admissible, and hence the trajectories generated by a particular LTI system. As a matter of convenience, we will generate one GST per LTI system *per initial condition*, so we describe the set of behaviors of a particular LTI system as follows:

Definition 46 (Behaviors of an LTI System). *An LTI system (as in (5.4)) that satisfies (A1) will have a set of behaviors (from initial condition x_0) specified by:*

$$\mathcal{B}_{x_0}(A, B, C) = \left\{ (u, y, x) \in (\mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^m) \times (\mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^\ell) \times (\mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^n) : x(0) = x_0 \right. \\ \left. \bigwedge \dot{x}(t) = Ax(t) + Bu(t) \bigwedge y(t) = Cx(t) \bigwedge u, x \text{ and } y \text{ satisfy (A2)} \right\}. \quad (5.13)$$

Following terminology in [13] and [17], the trajectories u and y will be associated with external variables, and the trajectory x will be associated with an internal variable.

A set of behaviors like \mathcal{B}_{x_0} is adequate to specify a GST using the subsequent methodology; see also [15, 17] as noted above. However, in the spirit of reducing nondeterminism in the resultant surrogate KSs (for the purposes elucidated in the introduction), we also need to incorporate the restrictions of (A3) in a *consistent* way. We can accomplish this by indexing \mathcal{B}_{x_0} further by a particular branching choice function, $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_{\neq 0})$, and then *truncating* behaviors in \mathcal{B}_{x_0} that fail to comply with (A3) (as it is specified by \mathcal{F}). Thus, we proceed inductively along the *sequence* of isolated zeros of \dot{y} in a behavior $(u, y, x) \in \mathcal{B}_{x_0}$ in order to evaluate that behavior's candidacy in a new set (of behaviors) that complies with (A3), $\mathcal{B}_{x_0}^{\mathcal{F}}$.

Definition 47 (Choice-Restricted Behaviors of an LTI System). *Let \mathcal{B}_{x_0} be as in Definition 46, and let $\mathcal{F} = (\mathcal{F}_0, \mathcal{F}_{\neq 0})$ be a given branching choice function as described in Definition 45 and (A3). Furthermore, for a given behavior $b = (u, y, x) \in \mathcal{B}_{x_0}$, define $\{T_k^{(b)}\}$ to be the (possibly empty) sequence of times corresponding to the isolated zeros of the function $\dot{y}(t) = Cx(t) = CAx(t) + CBu(t)$. Then the set of choice-restricted behaviors is given by*

$$\mathcal{B}_{x_0}^{\mathcal{F}}(A, B, C) = \left\{ b|_T : b = (u, y, x) \in \mathcal{B}_{x_0} \wedge \right. \\ \left. T = \min \left\{ T_k^{(b)} : ([x(T_k^{(b)})], u(T_k^{(b)})) \text{ fails to satisfy (A3) under } \mathcal{F} \right\} \right\} \quad (5.14)$$

where the notation $|_T$ indicates that the truncation operator excludes T itself (and so restricts b to a right-open interval of the time line, \mathbb{R}). In the case that $T = \infty$, the restriction operation is ignored, and the behavior is included unmodified; if $T = 0$, then the entire behavior is considered to be excluded.

Given such a set of choice-restricted behaviors, we can follow the straightforward procedure in [15, 17] to create a GST; the only material difference is in the restrictions we have placed on the input functions (encapsulated as it is in the specification of the the choice-restricted set of behaviors, $\mathcal{B}_{x_0}^{\mathcal{F}}(A, B, C)$).

Definition 48 (GST representation of an LTI system). *Given a set of behaviors derived from an LTI system, $\mathcal{B}_{x_0}(A, B, C)$, we construct a GST $G_{x_0}(A, B, C) = (P, \preceq, p_0, \mathcal{L})$ according to the following.*

- The set P is given by

$$P = \left\{ (t', f|_{[0,t']}) \in \mathbb{R}^{\geq 0} \times ([0, t'] \rightarrow \mathbb{R}^{m+\ell+n}) : \right. \\ \left. \exists f = (u, y, x) \in \mathcal{B}_{x_0}^{\mathcal{F}}(A, B, C) \forall t \in [0, t'] \cdot f(t) = f|_{[0,t']}(t) \right\} \cup \{p_0\}$$

where p_0 is the root of the tree, and it is chosen to make the union disjoint.

- The set P is partially ordered by \preceq using trajectory prefixing as follows:

$$\forall (t'_1, f_1|_{[0,t'_1]}), (t'_2, f_2|_{[0,t'_2]}) \in P \setminus \{p_0\} : \\ (t'_1, f_1|_{[0,t'_1]}) \preceq (t'_2, f_2|_{[0,t'_2]}) \Leftrightarrow t'_1 \leq t'_2 \bigwedge \forall t \in [0, t'_1] \cdot f_1|_{[0,t'_1]}(t) = f_2|_{[0,t'_2]}(t)$$

and

$$\forall (t'_1, f_1|_{[0,t'_1]}) \in P \setminus \{p_0\} : p_0 \preceq (t'_1, f_1|_{[0,t'_1]}).$$

- The labeling function $\mathcal{L} : P \setminus \{p_0\} \rightarrow \mathbb{R}^{\ell+m}$ is given by

$$\mathcal{L} : (t'_1, (u, y, x)|_{[0,t'_1]}) \in P \setminus \{p_0\} \mapsto (u(t'), y(t')).$$

It is important to note that the possibility of choosing among piece-wise analytic input functions will generate “asymptotic” branching behavior: in particular, such an input function can be “interrupted” with a new piece-wise analytic input function at any time to obtain a *new* piece-wise analytic input function! This means that *at any given time* there are executions of the system with multiple different control inputs allowed, and this possibility is reflected in the GST by the existence of different nodes with the same (partial) state and output trajectories.

Another important observation about this construction is that it reflects the nature of LTI systems as past-induced and Markovian state-based systems. In particular, the elements of P themselves can be regarded as having the property of *states* for the underlying LTI system; this is explained more precisely in [17].

5.4 LTI Systems and Order-Equivalence Semantics

The goal of this section is to describe the fundamental nature of order equivalences between LTI-derived trajectories; this is an important way point on the path to characterizing surrogate KSs of LTI-derived GSTs. The restrictions described in [Section 5.3](#) play a special role here, as they have the effect of *restricting* the possible order-equivalences between LTI-derived trajectories in some manageable – even minimal – way. Understanding the nature of these order equivalences is of course a necessary prerequisite to understand *VHMM classes* of such systems in the context of GHML and weak bisimulation, since those notions are defined in terms of order equivalent trajectories. Moreover, this task will ultimately be easier because we have created context where there is some minimal, manageable characterization of order equivalences.

Importantly, the weak bisimulation/GHML semantics – essentially the semantics of *order equivalences* (see [Definition 28](#)) – can be seen to turn essentially *deterministic* GSTs into *non-deterministic* surrogate Kripke Structures; this fundamental fact has significant consequences as far as surrogate KSs are concerned. To get a sense of this phenomenon, recall the following example of a GST from [19] and

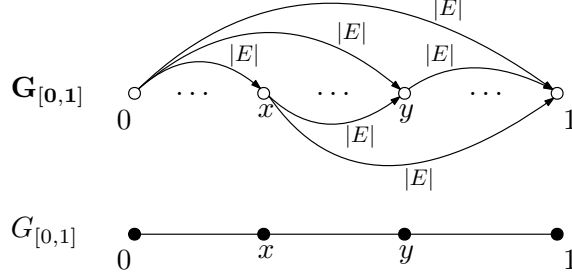


Figure 5.1: GST and surrogate KS from [Example 2](#); define $E : (0, 1] \rightarrow \{\alpha\}$.

depicted in the lower portion of [Fig. 5.1](#).

Example 2 ([\[19\]](#)). Consider the following GST defined over the unit interval $[0, 1] \subset \mathbb{R}$:

$$G_{[0,1]} := ([0, 1], \leq_{\mathbb{R}}, 0, (0, 1] \rightarrow \{\alpha\}). \quad (5.15)$$

Note that the GST $G_{[0,1]}$ is about as deterministic as one could hope for, since it has essentially no branching behavior at all: *indeed, the tree partial order is itself also a linear order!* However, when we consider $G_{[0,1]}$ in the context of a natural domain of modalities, $(\bar{\mathcal{J}}, \leq_{\mathbb{R}}) = (\mathbb{R}, \leq_{\mathbb{R}})$, the surrogate KS exhibits explicit non-determinism: see the surrogate KS in the upper portion of [Fig. 5.1](#)!

Of course this feature will inevitably manifest itself even in *deterministic* LTI systems when they are converted into GSTs using the methodology of the previous section; thus, the results of this section may be regarded as ultimately describing the *nondeterminism* of the those GSTs under this order-equivalence semantics. Indeed, the restrictions we imposed on our LTI systems and their inputs will eventually result in surrogate KSs that have some *minimal* amount of non-determinism; this effort makes the nondeterminism more tractable going forward, but even so, it leaves behind the essential character of nondeterminism described above, which seems prac-

tically unavoidable. In the subsequent section, then, we will examine VHHM classes associated with such GSTs by studying the special and specific *modal* properties of this nondeterminism (as these nondeterministic structures appear *throughout* such GSTs).

5.4.1 Preliminaries

Before we begin the primary enterprise, we use this opportunity to introduce some important facts and notation about order equivalences in LTI-derived GSTs. First, we note that trajectories in an LTI-derived GST $G_{x_0}(A, B, C)$ have a natural order-preserving bijection between half-open subintervals of the real-line; this is simply by the nature of the tree partial order in terms of partial time trajectories, ordered by prefixing. Hence, we have the following remark.

Remark 2. *Let $G_{x_0}(A, B, C)$ be a GST created according to [Section 5.3](#), and consider a pair of trajectories in $G_{x_0}(A, B, C)$ that are specified by left endpoints $p'_i = (t'_i, (u_i, y_i, x_i)|_{t'_i})$ and right endpoints $p''_i = (t''_i, (u_i, y_i, x_i)|_{t''_i})$ for $i = 1, 2$. Without loss of generality, we will generally describe an order equivalence between $(p'_1, p''_1] \rightarrow (p'_2, p''_2]$ using a state-trajectory-specific function that is an order-preserving (monotonic) bijection from one half-open subinterval of the real line to another:*

$$\lambda_{x_1, x_2} : (t'_1, t''_1] \rightarrow (t'_2, t''_2]. \quad (5.16)$$

Such a λ is thus “localized” to a particular pair of state/input trajectories, so that times can be uniquely associated with GST nodes; however, we will often omit the state subscripts for readability. Of course this λ should preserve GST labels as

expected:

$$(u_1(t), y_1(t)) = (u_2(\lambda(t)), y_2(\lambda(t))) \text{ for } t \in (t'_1, t''_1]. \quad (5.17)$$

Given this remark, we can state two further useful facts based on what we know about monotonic functions defined over the reals and the assumptions we have made about our GSTs.

Proposition 6. *Let $\lambda : (t'_1, t''_1] \rightarrow (t'_2, t''_2]$ be an order equivalence between two trajectories (u_1, y_1, x_1) and (u_2, y_2, x_2) in a GST $G_{x_0}(A, B, C)$. Then we can uniquely extend the domain and codomain of λ so that $\lambda : [t'_1, t''_1] \rightarrow [t'_2, t''_2]$ is an order-preserving bijection. In particular, $u_1(t'_1) = u_2(t'_2)$ and $y_1(t'_1) = y_2(t'_2)$.*

Proof. This follows from the assumed right-continuity of the inputs **(A2)** and the continuity of the outputs; the latter is a consequence (5.5). \square

Proposition 7. *An order-preserving (monotonic) bijection $\lambda : [t'_1, t''_1] \subset \mathbb{R} \rightarrow [t'_2, t''_2] \subset \mathbb{R}$ is continuous at every point in its domain and differentiable at almost every point in its domain.*

Proof. First, we know that any monotonic function $\lambda : [t'_1, t''_1] \rightarrow [t'_2, t''_2]$ is continuous at all but countably many points in its domain [35]. But if such a function is, in addition, *bijective*, then it is necessarily continuous at every point in its domain; this follows from the special nature of the discontinuities for monotonic functions on \mathbb{R} [35]. Thus, such an order equivalence function is in fact continuous at every point in its domain and differentiable almost everywhere, the latter being an explicit claim in [35]. \square

5.4.2 LTI Systems and Order-Equivalence Non-determinism: Purely Analytic Inputs and “Local” Non-determinism Structure

Now consider an arbitrary LTI system started from the initial condition $x_0 = 0$. Irrespective of the choice of A , B and C , the output at time $t = 0$ is necessarily $y(0) = 0$; moreover, the point $x = 0$ makes $Ax = 0$ for *any* such LTI system, *so the output will continue to remain at zero for as long as the input is set to zero!* Of course these trajectories also satisfy assumptions **(A1)**-**(A3)** (**(A3)** in particular doesn't apply since we've posited constant input/output trajectories). Thus, if we retain the natural domain of modalities, $(\bar{\mathcal{J}}, \leq_{\mathbb{R}}) = (\mathbb{R}, \leq_{\mathbb{R}})$, then the surrogate KS for $G_0(A, B, C)$ will share a structure like that in [Example 2](#) for all of the GST nodes specified by the state/input trajectories $(u, x) : [0, t'] \times [0, t'] \rightarrow \{0\} \times \{0\}$ – all of which have (GST) labels $(u(t'), y(t')) = (0, 0)$. In other words, for any $t' < t''$, the surrogate KS will have a transition $|Z : (0, 1] \rightarrow \{0\} \times \{0\}|$ from the node specified by $(u, x_1) : [0, t'] \times [0, t'] \rightarrow \{0\} \times \{0\}$ to the node specified by $(u, x_2) : [0, t''] \times [0, t''] \rightarrow \{0\} \times \{0\}$! In particular, the surrogate KS exhibits nondeterminism in this $|Z|$ transition at *each* such GST node, including the one for the associated singleton state/input trajectory that ends at $t = 0$ (the “beginning” of this constant input/output trajectory in the GST/LTI system).

So it is quite clear that even for our deterministic LTI systems, constant inputs that yield constant outputs will generate some kind of nondeterminism in the corresponding surrogate KS: this is through the (above described) possibility that different trajectories may be order equivalent even though they end up in different states.

Moreover, constant inputs are included in most practical classes of input functions, so this type of constant-input/constant-output nondeterminism would seem to be unavoidable for all intents and purposes. Hence, the goal of this section can be recast as answering the question of whether or not there are *other* possible sources of nondeterminism in such surrogate KSs – manifest as they must be in more diverse order equivalences between LTI-system trajectories. In particular, the possible alternatives are order equivalences between *non-constant* input/output trajectories or order equivalences between trajectories that don't start at $t = 0$. However, as we have alluded to earlier, the assumptions **(A1)**-**(A2)** were chosen quite deliberately to avoid nondeterminism in these situations. And it is in this sense that we have specified the *minimum* amount of nondeterminism that can emerge in an LTI system's surrogate KS.

Thus, given a GST $G_{x_0}(A, B, C)$, we outline a proof that this constant-input/constant-output nondeterminism is essentially the *only* nondeterminism in the transitions emanating from a *given node* in the surrogate KS; this will be a specific consequence of the assumptions we have made in [Section 5.3](#). Hence, we suppose that we have a fixed node – or more generally, a fixed initial condition – for the aforementioned LTI system, and we attempt to characterize all of the order equivalent trajectories emanating from that point. Overall, the strategy is to build order equivalences by using induction on the time intervals associated with the analytic segments of a particular input u . Initially, though, we will consider only order equivalences that begin at time $t = 0$ and span a *single analytic segment of input*. This will prove to be without loss of generality, since the time invariance of the underlying dynamics

will allow us to consider nodes elsewhere in $G_{x_0}(A, B, C)$, or equivalently, analytic input segments that start at later times. However, there is a further important prerequisite for an inductive argument that incorporates a *sequence* of analytic segments of input: unfortunately, an order equivalence really only supplies first-hand knowledge about inputs and outputs – rather than *states* – so we must first specify a *state-based* invariant that will be preserved under order equivalence, at least for any order equivalence between trajectories whose initial conditions satisfy the invariant. This is indeed essential, because even order equivalent trajectories emanating from the same initial condition can leave the system in *different* states. This consideration means that as we characterize order equivalences, we will also have to show that any candidate order equivalence preserves the aforementioned invariant (as a prelude: we will consider order equivalences for all of the permutations of constant/non-constant inputs with constant/non-constant outputs). As claimed, this will allow us to build order equivalences between arbitrary trajectories by induction. Of course the invariant we’re going to use was already introduced (with some prescience) in [Section 5.3](#): we’re going to work in terms of *output invariance* between states.

As indicated in the program above, we will initially consider trajectories starting at time $t = 0$ and emanating from states that are *output invariant*, since this will prove to be an invariant with respect to all of the possible order equivalences. Now, to characterize order equivalent trajectories under these assumptions, we will further divide the problem into consideration of the following permutations of constant/non-constant inputs and outputs: the constant-input/constant-output order equiva-

lences from the example above; the non-constant-output/non-constant-input order equivalences; the constant-output/non-constant-input order equivalences; and the non-constant-output/constant-input order equivalences. For all but the constant-input/constant-output case, the task will be to show that whenever the initial *states* satisfy the claimed invariant, then the *only* possible order equivalence is the identity. We begin this sequence with the order equivalences we already know to exist from the discussion above, viz. the constant-input/constant-output case: in this case, though, we need only confirm that output invariance is indeed preserved by valid order equivalences.

Throughout the rest of this section, we will use the following notation for the derivative of a function; this will eliminate any confusion between orders of differentiation and exponents.

$\dot{y}^{\kappa}(t) \triangleq \frac{\partial^{\kappa}}{\partial t^{\kappa}} y(t)$

Table 5.1: Special notation for derivatives.

Lemma 3 (Constant Input/Constant Output). *Let $x_{0,1}$ and $x_{0,2}$ be output invariant states from an LTI system; also let y_i be the constant output trajectory and x_i be the state trajectory obtained from initial condition $x_{i,0}$ and constant input u (where $i = 1, 2$). Then for any order equivalence $\lambda : [0, t_1] \rightarrow [0, t_2]$, it is the case that $x_1(t)$ and $x_2(\lambda(t))$ are output invariant for all $t \in [0, t_1]$.*

Proof. We first note that since the outputs y_1 and y_2 are constant on $[0, t_1]$ and $[0, t_2]$, respectively, all of their derivatives are zero on those intervals.

Using this fact on the first derivative of the outputs yields:

$$\dot{y}_1(t) = C\dot{x}_1(t) = 0 = C\dot{x}_2(\tau)|_{\tau=\lambda(t)} = \dot{y}_2(\tau)|_{\tau=\lambda(t)}. \quad (5.18)$$

But filling in the values of the derivatives according to the state equation (5.4), we get:

$$CAx_1(t) + CBu = 0 = CAx_2(\lambda(t)) + CBu \quad (5.19)$$

which by linearity, implies that:

$$CA(x_1(t) - x_2(\lambda(t))) = 0. \quad (5.20)$$

A similar argument for the k^{th} derivatives leads to an equation like the preceding, only with the k^{th} power of A . Hence,

$$CA^k(x_1(t) - x_2(\lambda(t))) = 0 \quad (5.21)$$

for all $k \geq 1$.

On the other hand, we have assumed that λ is an order equivalence, so by necessity $y_1(t) = y_2(\lambda(t))$ and $C(x_1(t) - x_2(\lambda(t))) = 0$. This shows the required property of the difference $x_1(t) - x_2(\lambda(t))$. \square

This certainly suggests that we have chosen a reasonable invariant! But of course that prescience will be even more clear after we consider the case when we allow the outputs to be non-constant (and the inputs to be constant or not): in such a case, the output invariance will lead to the claimed *unique* order equivalence. Before we take up the proof of our claim for this case, we make one important comment about order equivalences. In particular, given the variation of constants

formula, (5.5), two order equivalent LTI trajectories (u_1, y_1, x_1) and (u_2, y_2, x_2) can also be specified in terms of (u_2, y_2, x_2) , the initial condition $x_{0,1}$ and an order equivalence function, λ . That is given a trajectory (u_2, y_2, x_2) , an initial condition $x_{0,1}$ and an order equivalence, λ , we have the following alternate specification of the LTI trajectory (u_1, y_1, x_1) :

$$x_1(t) = e^{At}x_{0,1} + \int_0^t e^{A(t-\tau)}Bu_2(\lambda(\tau))d\tau. \quad (5.22)$$

Of course an x_1 specified in this way need not be analytic, since we have not constrained the choice of λ in any way that ensures this; indeed the function $u_2 \circ \lambda$ might not even be piece-wise analytic and hence inadmissible. However, in the trivial portions of the subsequent proof, we will be able to easily conclude that λ – and hence $u_2 \circ \lambda$ and y_2 are analytic; in the rest, analyticity of λ will be one of the crucial implications of **(A3)**. At any rate, if there *is* an order equivalence between two admissible (analytic) trajectories (u_1, y_1, x_1) and (u_2, y_2, x_2) , then (5.22) specifies x_1 just as well.

Given this fact, we proceed with the statement of our claimed result for order equivalences between non-constant-output trajectories, though we first introduce the following small lemma to aid this endeavor.

Proposition 8. *Let $u : [a, b] \rightarrow \mathbb{R}$ be analytic and non-constant. Then the functional equation $u(t) = u(\lambda(t))$ has one and only one continuous solution, $\lambda = \lambda_{id} : t \mapsto t$.*

Proof. We consider first the case when \dot{u} is non-constant on $[a, b]$. In this case, we observe that \dot{u} is also analytic, and hence, has isolated zeros in $[a, b]$. Thus,

we can divide $[a, b]$ into finitely many adjacent, open intervals on which \dot{u} is non-zero. But by a generalization of the inverse function theorem, u is invertible in the neighborhood of each point t' within these intervals, and the inverse function is analytic. Hence, $\lambda(t) = u^{-1}(u(t)) = t$ in the neighborhood of each t' for which $\dot{u}(t') \neq 0$. But since the solution to this equation is the same for each such t' , and there are only finitely points at which we cannot apply this reasoning (those where $\dot{u} = 0$), we conclude by assumed continuity of λ that $\lambda = \lambda_{\text{id}} : t \mapsto t$ on all of $[a, b]$.

The case where \dot{u} is constant and zero is excluded by the assumption that u is non-constant, so the only remaining case is when \dot{u} is constant and non-zero. But in this case the aforementioned inverse function theorem may be applied in the neighborhood of every point in the interval $[a, b]$, and we obtain the same result as above. □

Lemma 4 (Non-constant Output/Non-constant Input). *Let y_i , x_i and $x_{i,0}$ be as in [Lemma 3](#), only with respective inputs $u_i : [0, t_i] \rightarrow \mathbb{R}^m$ analytic on their domains and with the y_i and u_i **non-constant**. If the state trajectories x_i and input trajectories u_i satisfy **(A3)** with respect to the outputs y_i (in particular at time $t = 0$), then $\lambda : [0, t_1] \rightarrow [0, t_2]$ is an order equivalence between the aforementioned trajectories if and only if it is the identity order equivalence, $\lambda_{\text{id}} : t \mapsto t$.*

Proof. The essence of this proof will be to “assemble” the identity order equivalence by induction on the time-intervals for which a particular output trajectory’s derivative, \dot{y}_i , is *non-zero* (note: this induction is separate from the one claimed above, which assembles analytic segments of a piece-wise analytic function). We

can accomplish this by first showing that the only valid order equivalence between the (u_i, y_i, x_i) in some neighborhood of $t = 0$ is the identity order equivalence, $\lambda_{\text{id}} : t \mapsto t$. If we do this for both the case when the $\dot{y}_i(0) = 0$ and the case when the $\dot{y}_i(0) \neq 0$, then we are essentially done: since the identity order equivalence preserves output-invariance of the states, we can “propagate” the claimed identity order equivalence from $t = 0$ (for as long as it’s valid) to get two new *output-invariant states* at a later time. But time invariance allows us to recycle our claims for time $t = 0$ at this new time, so we can again claim necessity and uniqueness of the identity order equivalence for some *extended* interval of time. If the $\dot{y}_i(0)$ were zero, then because non-constant analytic functions have isolated-zeros¹, this “propagation” can be used to transfer the system state to one where the output derivatives are *non-zero*; of course, if the original states had non-zero output derivatives, then continuity allows us to assert that this “propagation” must be extendable at least until the first zero of the output derivative(s). Together, these two observations allow us to claim by induction that these entire analytic segments of the trajectories have between them a unique order equivalence, the identity order equivalence.

We proceed with the proof as outlined above, and first consider the case where $y_1(0) = y_2(\lambda(0)) \neq 0$. From this, note that any order equivalence λ is necessarily analytic in some neighborhood of $t = 0$: an order equivalence by definition satisfies $y_1(t) = y_2(\lambda(t))$, which can be solved for $\lambda(t)$ as $\lambda(t) = g(y_1(t))$ for some *analytic* function g by a generalization of the inverse function theorem for analytic functions

¹The case where the derivative is constant (but still non-zero by stipulation) is an obvious special case. We omit it from this top-level discussion for clarity.

y_1 and y_2 [34] (this of course requires the non-zero derivative $\dot{y}_2(0)$). Thus, we need only search for *analytic* order equivalences in the neighborhood of $t = 0$, so our strategy will be to treat the *analytic* order equivalence function λ as a “variable” in an equation $y_1(t) = y_2(\lambda(t))$ obtained by assuming $u_1 = u_2 \circ \lambda$ – that is λ satisfies the requirement of an order equivalence between the inputs by construction (and with the initial conditions as described in the statement of the lemma); see also the prefatory comment preceding this lemma. If $\lambda = \lambda_{\text{id}}$ is the *only* possible solution to the aforementioned equation, then we conclude that when (u_1, y_1, x_1) and (u_2, y_2, x_2) are order equivalent, the order equivalence between them is necessarily λ_{id} . This description is thus one of establishing $\lambda = \lambda_{\text{id}}$ as a *necessary* condition for order equivalence under the conditions of the lemma.

Since λ is analytic near $t = 0$ in this case, we can establish it uniquely by characterizing all of its derivatives at time $t = 0$, near which it necessarily agrees with its power series expansion about the same point, $t = 0$. Moreover, analyticity of λ implies $y_2 \circ \lambda$ is also analytic, so the equality $y_1 = y_2 \circ \lambda$ implies equality of coefficients in the power series expansions of y_1 and $y_2 \circ \lambda$ (about $t = 0$ is the natural choice). Of course this means we are considering

$$\dot{y}_1^{\cdot\kappa}(0) = \frac{d^\kappa}{dt^\kappa} y_2(\lambda(0)) \quad (5.23)$$

for all $\kappa \in \mathbb{N}$ (the normalization $1/\kappa!$ is common to both, and hence may be omitted). Since we’re working with linear dynamics, (5.4), and we have assumed that $u_1 =$

$u_2 \circ \lambda$, we can further note that

$$\begin{aligned} \dot{y}_1^\kappa(t) &= CA^\kappa x_1(t) + \sum_{j=0}^{\kappa-1} CA^{\kappa-1-j} B \dot{u}_1^j(t) \\ &= CA^\kappa x_1(t) + \sum_{j=0}^{\kappa-1} CA^{\kappa-1-j} B \frac{d^j}{dt^j} (u_2 \circ \lambda)(t). \end{aligned} \quad (5.24)$$

On the other hand, by Faà di Bruno's formula [34], we note that

$$\begin{aligned} &\frac{d^\kappa}{dt^\kappa} y_2(\lambda(t)) \\ &= \sum_{k_1+2k_2+\dots+\kappa k_\kappa=\kappa} \frac{\kappa!}{k_1!k_2!\dots k_\kappa!} \dot{y}_2^\sigma(\lambda(t)) \left(\frac{\dot{\lambda}(t)}{1!}\right)^{k_1} \left(\frac{\ddot{\lambda}(t)}{2!}\right)^{k_2} \dots \left(\frac{\dot{\lambda}^\kappa(t)}{\kappa!}\right)^{k_\kappa} \end{aligned} \quad (5.25)$$

where the variable $\sigma = k_1 + k_2 + \dots + k_\kappa$, and the sum is over the variables k_1, \dots, k_κ subject to the indicated constraint (we will use this formula several times in the sequel). We thus create a necessary conditions for an analytic order equivalence λ by simply substituting (5.24) and (5.25) into (5.23) for $\kappa = 1, 2, \dots$ (in that order), and setting $t = 0$ to get the relevant power series coefficients; naturally, we will apply Faà di Bruno's formula as needed to the derivatives of $u_2 \circ \lambda$ in (5.24).

Considering (5.23) first for $\kappa = 1$ we obtain simply

$$CAx_{1,0} + CBu_2(\lambda(0)) = (CAx_{2,0} + CBu_2(\lambda(0))) \cdot \dot{\lambda}(0). \quad (5.26)$$

But as we have assumed that $x_{1,0}$ and $x_{2,0}$ are *output invariant*, we note that $x_{1,0} = x_{2,0} + \xi$ for some $\xi \in \mathcal{N}(\mathcal{O})$; considering this fact in the above, we obtain

$$\begin{aligned} &(CAx_{2,0} + CBu_2(\lambda(0))) \cdot (1 - \dot{\lambda}(0)) + (CA\xi) \cdot \dot{\lambda}(0) = 0 \\ \implies &(CAx_{2,0} + CBu_2(\lambda(0))) \cdot (1 - \dot{\lambda}(0)) = 0 \end{aligned} \quad (5.27)$$

since $\xi \in \mathcal{N}(\mathcal{O})$ and hence $CA\xi = 0$. Of course, we have assumed that $y_2(\lambda(0)) = CAx_{2,0} + CBu_2(\lambda(0)) \neq 0$, so already we see that $\dot{\lambda}(0) = 1$ is the only possible choice

for $\dot{\lambda}(0)$ under the assumptions that we have made. Now we proceed inductively through the remaining values of κ . We take $\kappa = 2$ as the base case, and claim that given (5.27), the only possible solution for $\ddot{\lambda}(0)$ is $\ddot{\lambda}(0) = 0$; then as an induction step, we claim that $\forall j \leq \kappa - 1. \dot{\lambda}^j(0) = 0$ implies that $\dot{\lambda}^\kappa(0) = 0$. This will identify λ as $\lambda_{\text{id}} : t \mapsto t$ in some neighborhood of $t = 0$.

The base case can be established by considering the relevant forms of (5.24) and (5.25) and taking into account the unique solution $\dot{\lambda}(0) = 1$:

$$\begin{aligned}
& CA^2x_{1,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0))\dot{\lambda}(0) = \ddot{y}_2(\lambda(0))\dot{\lambda}^2(0) + \dot{y}_2(\lambda(0))\ddot{\lambda}(0) \\
\implies & CA^2x_{1,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0)) = \ddot{y}_2(\lambda(0)) + \dot{y}_2(\lambda(0))\ddot{\lambda}(0) \\
\implies & CA^2x_{1,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0)) \\
& = CA^2x_{2,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0)) + \dot{y}_2(\lambda(0))\ddot{\lambda}(0) \quad (5.28)
\end{aligned}$$

Noting again that $x_{1,0} = x_{2,0} + \xi$ for some $\xi \in \mathcal{N}(\mathcal{O})$ we see immediately that the last equation above simplifies to

$$\dot{y}_2(\lambda(0))\ddot{\lambda}(0) = 0. \quad (5.29)$$

But of course $\dot{y}_2(\lambda(0)) \neq 0$ by assumption, so the only solution for $\ddot{\lambda}(0)$ is $\ddot{\lambda}(0) = 0$, and this establishes the base case.

The induction step can be established by careful book-keeping, and using Faà di Bruno's formula to expand $\frac{d^j}{dt^j}(u_2 \circ \lambda)$ in (5.24); of course the assumption that higher derivatives of λ are zero makes this considerably easier to do. To start, we take κ to be fixed and the induction hypothesis to be true. By Faà di Bruno's

formula, we expand the final term in (5.24) as promised by noting that

$$\begin{aligned}
& \frac{d^j}{dt^j}(u_2 \circ \lambda)(t) \\
&= \sum_{k_1+2k_2+\dots+jk_j=j} \frac{j!}{k_1!k_2!\dots k_j!} \overset{\sigma}{u}_2(\lambda(t)) \left(\frac{\dot{\lambda}(t)}{1!}\right)^{k_1} \left(\frac{\ddot{\lambda}(t)}{2!}\right)^{k_2} \dots \left(\frac{\overset{j}{\lambda}(t)}{j!}\right)^{k_j} \\
&= \overset{j}{u}_2(\lambda(t)) \overset{j}{\lambda}^j(t) = \overset{j}{u}_2(\lambda(t))
\end{aligned} \tag{5.30}$$

where $\sigma = k_1 + k_2 + \dots + k_j$ as before; this clearly follows from the induction assumption that $\overset{j}{\lambda}(0) = \overset{j}{\ddot{\lambda}}(0) = \dots = \overset{j}{\lambda}(0) = 0$ for $j \leq \kappa - 1$ and $\dot{\lambda}(0) = 1$. On the other hand, applying the induction assumption to (5.25), we see that the only (possibly) nonzero terms occur when the indexes $k_2 = k_3 = \dots = k_{\kappa-1} = 0$. In this case there are only two solutions to the constraint equation, which becomes $k_1 + \kappa k_\kappa = \kappa$: namely, $k_1 = \kappa, k_\kappa = 0$ and $k_1 = 0, k_\kappa = 1$. Thus, we can simplify (5.25) to be simply

$$\begin{aligned}
\frac{d^\kappa}{dt^\kappa} y_2(\lambda(t)) &= \overset{\kappa}{y}_2(\lambda(0)) \overset{\kappa}{\lambda}^\kappa(0) + \overset{\kappa}{y}_2(\lambda(0)) \overset{\kappa}{\lambda}(0) \\
&= \overset{\kappa}{y}_2(\lambda(0)) + \overset{\kappa}{y}_2(\lambda(0)) \overset{\kappa}{\lambda}(0).
\end{aligned} \tag{5.31}$$

But when we consider (5.23) under the conclusions derived above, we get

$$CA^\kappa x_{1,0} + \sum_{j=0}^{\kappa-1} CA^{\kappa-1-j} B \overset{j}{u}_2(\lambda(0)) = \overset{\kappa}{y}_2(\lambda(0)) + \overset{\kappa}{y}_2(\lambda(0)) \overset{\kappa}{\lambda}(0). \tag{5.32}$$

However, as before $x_{1,0} = x_{2,0} + \xi$ for $\xi \in \mathcal{N}(\mathcal{O})$, so that we observe in the preceding that the left-hand side cancels exactly the first term on the right-hand side, and

that leaves the following simple equation:

$$\dot{y}_2(\lambda(0))\dot{\lambda}(0) = 0. \quad (5.33)$$

Of course this equation has as a unique solution $\dot{\lambda}(0) = 0$, since we have assumed that $\dot{y}_2(\lambda(0)) \neq 0$. This completes the induction and this portion of the proof, so we conclude that when $\dot{y}_1(0) = \dot{y}_2(\lambda(0)) \neq 0$, the only possible order equivalence between the specified trajectories is $\lambda = \lambda_{\text{id}}$.

Now we proceed with the case when $\dot{y}_1(0) = \dot{y}_2(\lambda(0)) = 0$. First of all, we can no longer apply the (more general) inverse function theorem to the equation $y_1(t) = y_2(\lambda(t))$ in order to confine our attention to only analytic functions λ . However, we have designed **(A3)** in such a way as to ensure that the only valid *inputs* allow us to draw a similar conclusion. We have two cases now: either the $([x_{1,0}], u_1(0)) = ([x_{2,0}], u_2(\lambda(0)))$ are in the domain of \mathcal{F}_0 , or else they are in the domain of $\mathcal{F}_{\neq 0}$. In the former case, by **(A3)**, we see that $\dot{u}_1 = \dot{u}_2(\lambda(0)) \neq 0$, so we may directly apply the aforementioned inverse function theorem to conclude we should restrict our attention to analytic order equivalences λ . The latter case is more complicated, since **(A3)** allows a single, albeit unique input u for which $\dot{u}(0) = 0$. Clearly, then, when we consider any possible order equivalence between allowable inputs we have a number of different scenarios. In the most obvious case, $\dot{u}_1(0)$ and $\dot{u}_2(\lambda(0))$ are both nonzero, and earlier arguments show that λ must be analytic in a neighborhood of $t = 0$. Another scenario has $\dot{u}_1(0) = 0$ and $\dot{u}_2(\lambda(0)) \neq 0$ or visa versa. But in this case, we can clearly choose either λ or λ^{-1} to be analytic in a neighborhood of $t = 0$ according to the inverse function theorem, and we can

simply swap the order of the initial conditions in the sequel accordingly. In the final case, both $\dot{u}_1(0) = 0$ and $\dot{u}_2(\lambda(0)) = 0$. But by **(A3)** this means that $u_1 = u_2$, and [Proposition 8](#) ensures that the only possible order equivalence between them is λ_{id} , hence analytic.

Having established that we may confine ourselves to analytic order equivalence functions we proceed largely as above: we will use exactly versions of [\(5.23\)](#) to solve for derivatives of the order equivalence function, λ , only the equations will (unsurprisingly) look somewhat different. Unfortunately, the assumed situation that

$$\dot{y}_1(0) = CA[x_{1,0}] + CBu_1(0) = \dot{y}_2(\lambda(0)) = CA[x_{2,0}] + CBu_2(\lambda(0)) = 0 \quad (5.34)$$

means that we cannot begin an inductive proof in the same way as above. In particular, adapted to the current setting, [\(5.27\)](#) does not have a unique solution! In fact, $0 \cdot (1 - \dot{\lambda}(0)) = 0$ is no constraint at all on $\dot{\lambda}(0)$! So we have to use some of the other consequences of **(A3)** to first establish that $\dot{\lambda}(0) = 1$ is the unique choice for our analytic order equivalence. In the case that $([x_{1,0}], u_1(0)) = ([x_{2,0}], u_2(\lambda(0))) \in \text{dom}(\mathcal{F}_0)$, the stipulation of **(A3)** that $u_1(0) = u_2(\lambda(0))$ ensures (by the chain rule) that $\dot{\lambda}(0) = 1$. On the other hand, if $([x_{1,0}], u_1(0)) = ([x_{2,0}], u_2(\lambda(0))) \in \text{dom}(\mathcal{F}_{\neq 0})$, then either $u_1(0) = u_2(\lambda(0)) = 0$ and $u_1 = u_2$, in which case we have already argued that $\dot{\lambda}(0) = 1$, or else one of the u_i has non-zero derivative at $t = 0$, say $\dot{u}_2(\lambda(0)) \neq 0$ without loss of generality. But in this case, we can consider using [\(5.23\)](#) with $\kappa = 2$ in an attempt to create an equation that has $\dot{\lambda}(0) = 1$ as a unique solution; of course **(A3)** has just the right restrictions to make this possible (although those restrictions are clearly not the *only* ones to achieve this objective!). To wit, after

expanding (5.23) with $\kappa = 2$ using (5.24), the first line of (5.30) and (5.25), we get

$$\begin{aligned}
& CA^2x_{1,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0))\dot{\lambda}(0) \\
&= (CA^2x_{2,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0)))\dot{\lambda}^2(0) + (CAx_{2,0} + CBu_2(\lambda(0)))\ddot{\lambda}(0) \\
&= (CA^2x_{2,0} + CABu_2(\lambda(0)) + CB\dot{u}_2(\lambda(0)))\dot{\lambda}^2(0) \tag{5.35}
\end{aligned}$$

since by assumption, $CAx_{2,0} + CBu_2(\lambda(0)) = 0$. If we let $v = CA^2x_{2,0} + CABu_2(\lambda(0))$ and $w = CB\dot{u}_2(\lambda(0))$, then this gives us the following quadratic equation in $\dot{\lambda}(0)$:

$$(v + w)\dot{\lambda}^2(0) - w\dot{\lambda}(0) - v = 0. \tag{5.36}$$

Note that since we are working with $([x_{2,0}], u_2(\lambda(0))) \in \text{dom}(\mathcal{F}_{\neq 0})$, we can be assured that $v \neq 0$. Thus, if $v + w = 0$, then the above reduces to a linear equation in $\dot{\lambda}(0)$ that has a unique solution $\dot{\lambda}(0) = 1$; otherwise, the above has as solutions the following:

$$\dot{\lambda}(0) = 1 \quad \text{or} \quad \dot{\lambda}(0) = -\frac{[[v]]_k}{[[v]]_k + [[w]]_k}. \tag{5.37}$$

If $\dot{u}_2(\lambda(0)) = 0$, then clearly the only solution that yields an *increasing* order equivalence – i.e. $\dot{\lambda}(0) > 0$ – is $\dot{\lambda}(0) = 1$; otherwise, **(A3)** ensures that $-\frac{[[v]]_k}{[[v]]_k + [[w]]_k}$ is negative for some k , hence $\dot{\lambda}(0) = 1$ is the only solution to yield an increasing order equivalence. Thus, under the conditions of the lemma and the assumptions we have made, we conclude that $\dot{\lambda}(0) = 1$.

Since we used equation (5.23) with $\kappa = 2$ to establish that $\dot{\lambda}(0) = 1$, it makes sense to start an induction to establish $\overset{j}{\dot{\lambda}}(0) = 0$, $j = 2, 3, \dots$ with $\kappa = 3$. This difference aside, we proceed largely as before: the induction step will be to assume that $\overset{j}{\dot{\lambda}}(0) = 0$ for all $j < \kappa - 1$ and show that this implies $\overset{j}{\dot{\lambda}}(0) = 0$ for all $j < \kappa$, or

equivalently, that it implies $\overset{\cdot\kappa-1}{\lambda}(0) = 0$. Starting with the base case of $\kappa = 3$, we see that (5.23) and $\dot{\lambda}(0) = 1$ together with the usual substitutions gives us

$$(3\ddot{y}_2(\lambda(0)) - CB\dot{u}_2(\lambda(0)))\ddot{\lambda}(0) = 0. \quad (5.38)$$

Of course **(A3)** indicates that the above will have a unique, non-zero coefficient of $\ddot{\lambda}(0)$ both when $\dot{u}_2(\lambda(0)) = 0$ and otherwise: we simply consider all of the cases described in **(A3)**, starting with the case when $([x_{2,0}], u_2(\lambda(0))) \in \text{dom}(\mathcal{F}_0)$ and then both proscribed sub-cases when $([x_{2,0}], u_2(\lambda(0))) \in \text{dom}(\mathcal{F}_{\neq 0})$. In the domain of \mathcal{F}_0 , $\dot{u}_2(\lambda(0)) \neq 0$, so of course there will necessarily be a component of (5.38) with a non-zero coefficient for $\ddot{\lambda}(0)$. In the domain of $\mathcal{F}_{\neq 0}$, the situation where $\dot{u}_2(\lambda(0)) = 0$ poses no problem at all since the coefficient for $\ddot{\lambda}(0)$ reduces to simply v , which is non-zero by assumption. In the case that $\dot{u}_2(\lambda(0)) = 0$, **(A3)** contains the explicit requirement that there is a component k for which $3[[\ddot{y}_2(\lambda(0))]]_k - [[CB\dot{u}_2(\lambda(0))]]_k \neq 0$, so again we conclude that the only possible solution for $\ddot{\lambda}(0)$ is 0. This completes the proof of the base case.

The induction step now proceeds as suggested before: assume that $\overset{\cdot j}{\lambda}(0) = 0$ for all $j < \kappa - 1$ and show that this implies $\overset{\cdot j}{\lambda}(0) = 0$ for all $j < \kappa$, or equivalently, that it implies $\overset{\cdot\kappa-1}{\lambda}(0) = 0$. The crucial difference here relative the prior induction proof is the offset of the indexes: whereas before we were specifying the κ^{th} derivative of λ at index κ , here we will be specifying the $\kappa - 1^{\text{th}}$ derivative of λ at index κ . At index κ , then, we have to adjust (5.30) at $j = \kappa - 1$ to account for the weaker induction hypothesis, which only asserts up to $\overset{\cdot\kappa-2}{\lambda}(0) = 0$. In this situation, we have two solutions to the constraint equation: $k_1 = \kappa - 1$ and $k_{\kappa-1} = 0$; and $k_1 = 0$

and $k_{\kappa-1} = 1$, so that

$$\frac{d^{\kappa-1}}{dt^{\kappa-1}}(u_2 \circ \lambda)(0) = \overset{\cdot \kappa-1}{u_2}(\lambda(0))\dot{\lambda}^{\kappa-1}(0) + \dot{u}_2(\lambda(0)) \overset{\cdot \kappa-1}{\lambda}(0) \quad (5.39)$$

Recall that we only have to consider a maximum index of $\kappa - 1$ because the highest derivative of $u_2 \circ \lambda$ lags behind the derivative of highest derivative of $y_2 \circ \lambda$ by one; of course $\frac{d^j}{dt^j}(u_2 \circ \lambda)(0) = \overset{\cdot j}{u_2}(\lambda(t))$ for all $j < \kappa - 1$ by the same arguments above. In expanding $\frac{d^\kappa}{dt^\kappa} y_2(\lambda(t))$ as in (5.25), we also see fewer terms disappear due to our induction assumption. Indeed, we see that since only $\ddot{\lambda}(0) = \overset{\cdot \cdot}{\lambda}(0) = \dots = \overset{\cdot \kappa-2}{\lambda}(0) = 0$, we have to consider different solutions of the constraint equation: i.e. we should allow $k_1, k_{\kappa-1}$ and k_κ to be non-zero. This leads to exactly three solutions to the constraint equation: $k_1 = 0, k_{\kappa-1} = 0$ and $k_\kappa = 0$; $k_1 = 1, k_{\kappa-1} = 1$ and $k_\kappa = 0$; and $k_1 = 0, k_{\kappa-1} = 0$ and $k_\kappa = \kappa$. Thus, we simplify (5.23) to be

$$\begin{aligned} CA^\kappa x_{1,0} + \sum_{j=0}^{\kappa-1} CA^{\kappa-1-j} B \overset{\cdot j}{u_2}(\lambda(0)) \dot{\lambda}^j(t) + CB \dot{u}_2(\lambda(0)) \overset{\cdot \kappa-1}{\lambda}(0) \\ = \dot{y}_2(\lambda(0)) \overset{\cdot \kappa}{\lambda}(0) + \kappa \ddot{y}_2(\lambda(0)) \overset{\cdot \kappa-1}{\lambda}(0) + \overset{\cdot \kappa}{y_2}(\lambda(0)) \dot{\lambda}^\kappa(0). \end{aligned} \quad (5.40)$$

But using $\dot{\lambda}(0) = 1$ again, noting that $\dot{y}_2(\lambda(0)) = 0$, and using the equality $x_{1,0} = x_{2,0} + \xi$, we can reduce the above to simply:

$$\left(\kappa \ddot{y}_2(\lambda(0)) - CB \dot{u}_2(\lambda(0)) \right) \overset{\cdot \kappa-1}{\lambda}(0) = 0. \quad (5.41)$$

Finally, then, **(A3)** tells us that the preceding has a unique solution whenever $\kappa \geq 3$, so this completes our proof of the induction step. Thus, we conclude that $\lambda = \lambda_{\text{id}}$ in a neighborhood of $t = 0$ when $\dot{y}_1(0) = \dot{y}_2(\lambda(0)) = 0$, at least when **(A3)** is satisfied.

The rest of the proof follows straightforwardly from the arguments laid out in the introductory paragraphs. □

Now that we have completed proofs for both the constant-input/constant-output case and the non-constant-output case, we can consider the remaining cases: that is when the output is constant but the input is non-constant and when the input is constant but the output is non-constant.

Lemma 5 (Constant Output/Non-constant Input). *Let $x_{0,1}$ and $x_{0,2}$ be output invariant states from an LTI system; also let y_1 be a constant output associated with state trajectory x_1 , initial condition $x_{0,1}$ and non-constant input u_1 . Under assumption **(A2)**, u_1 is the unique input that generates the output y_1 from any initial condition that is output invariant with respect to $x_{0,1}$; hence, any order equivalence $\lambda : [0, t_1] \rightarrow [0, t_2]$ to a trajectory (u_2, y_2, x_2) (from initial condition $x_{0,2}$) is necessarily the identity order equivalence.*

Proof. We begin by considering the uniqueness assertion of the lemma. First, note that by constancy of the output y_1 , we can conclude that $\dot{y}_1(t) = C\dot{x}_1(t) = 0$ and

$$\begin{aligned} \ddot{y}_1(t) &= C\ddot{x}_1(t) = C \frac{d}{dt} [Ax_1(t) + Bu_1(t)] \\ &= CA^2x_1(t) + CABu_1(t) + CB\dot{u}_1(t) \\ &= 0. \end{aligned} \tag{5.42}$$

Of course the latter allows us to write down a linear ODE in u_1 :

$$CB\dot{u}_1(t) = -CA^2x_1(t) - CABu_1(t). \tag{5.43}$$

Here we see the reason for assumption **(A2)**: under that assumption, this equation can effectively be solved *uniquely* for \dot{u}_1 using a pseudo-inverse; for simplicity, we simply call this operator $(CB)^{-1}$.

Now, since the trajectory x_1 is obtained from the input u_1 driving our LTI system (from initial condition $x_{0,1}$), we can augment the dynamics (as before) to obtain a self-contained ODE for u_1 and x_1 :

$$\begin{aligned} \begin{bmatrix} \dot{u}_1 \\ \dot{x}_1 \end{bmatrix} &= \begin{bmatrix} -(CB)^{-1}CA^2x_1(t) - (CB)^{-1}CABu_1(t) \\ Ax_1(t) + Bu_1(t) \end{bmatrix} \\ &= \begin{bmatrix} -(CB)^{-1}CAB & -(CB)^{-1}CA^2 \\ B & A \end{bmatrix} \begin{bmatrix} u_1(t) \\ x_1(t) \end{bmatrix} \triangleq \tilde{A} \begin{bmatrix} u_1(t) \\ x_1(t) \end{bmatrix}. \end{aligned} \quad (5.44)$$

Because it's a linear ODE, we know that (5.44) has a unique solution for each particular initial condition. However, to get the claimed uniqueness result, we have to show that it produces the same solution – at least for u_1 – for any two *output invariant* initial conditions (that the initial condition for the variable u_1 must be the same is the only relevant situation because of the conditions of an order equivalence).

Using the variation of constants formula (5.5), we can write the solution to the (autonomous) linear ODE (5.44) in terms of a matrix exponential in the matrix \tilde{A} :

$$\begin{bmatrix} u_1(t) \\ x_1(t) \end{bmatrix} = e^{\tilde{A}t} \begin{bmatrix} u_1(0) \\ x_{0,1} \end{bmatrix} = \sum_{i=0}^{\infty} \left(\frac{t^i}{i!} \cdot \tilde{A}^i \begin{bmatrix} u_1(0) \\ x_{0,1} \end{bmatrix} \right). \quad (5.45)$$

So we can effectively verify that $x_{0,1}$ and $x_{0,2}$ lead to the same solution u_1 by checking that multiplying by successive powers of \tilde{A} preserves equality of the u_1 components and *output invariance* of the x_1 components. That is for

$$\begin{bmatrix} u_1^{(k,i)} \\ x_1^{(k,i)} \end{bmatrix} \triangleq \tilde{A}^k \begin{bmatrix} u_1(0) \\ x_{0,i} \end{bmatrix} = \tilde{A} \begin{bmatrix} u_1^{(k-1,i)} \\ x_1^{(k-1,i)} \end{bmatrix} \quad (5.46)$$

we should verify that for all $k \in \mathbb{N}$, $u_1^{(k,1)} = u_1^{(k,2)}$ and the “states” $x_1^{(k,1)}$ and $x_1^{(k,2)}$

are output invariant. This is, of course, best accomplished by induction on the exponent, k , with the base case given by the assumed initial conditions. Thus, suppose that $[u_1^{(k-1,1)}; x_1^{(k-1,1)}]$ and $[u_1^{(k-1,2)}; x_1^{(k-1,2)}]$ satisfy the claimed properties, and observe that

$$\begin{aligned} u_1^{(k,1)} - u_1^{(k,2)} &= \\ &- (CB)^{-1}CAB\left(u_1^{(k-1,1)} - u_1^{(k-1,2)}\right) - (CB)^{-1}CA^2\left(x_1^{(k-1,1)} - x_1^{(k-1,2)}\right) \\ &= 0 \end{aligned} \quad (5.47)$$

and

$$\begin{aligned} x_1^{(k,1)} - x_1^{(k,2)} &= B\left(u_1^{(k-1,1)} - u_1^{(k-1,2)}\right) + A\left(x_1^{(k-1,1)} - x_1^{(k-1,2)}\right) \\ &= A\left(x_1^{(k-1,1)} - x_1^{(k-1,2)}\right). \end{aligned} \quad (5.48)$$

But in the latter case, we note that the null space of \mathcal{O} is A -invariant, so (5.48) implies that in fact, $x_1^{(k,1)}$ and $x_1^{(k,2)}$ are output invariant as well. Consequently, (5.44) results in a *unique* solution for u_1 , given any two x_1 initial conditions that are output invariant.

However, the preceding gives us directly the claim for the order equivalence λ . For if there were a non-trivial order equivalence, then there would in fact be *two distinct* input functions that yield the same constant output (from output invariant initial conditions $x_{0,1}$ and $x_{0,2}$): namely $u_1 \neq u_2$ (with $u_1 = u_2 \circ \lambda$ for $\lambda \neq \lambda_{\text{id}}$)! \square

Lemma 6 (Non-constant Output/Constant Input). *Let $x_{0,1}$ and $x_{0,2}$ be output invariant states from an LTI system; also let y_1 be the non-constant output associated with state trajectory x_1 , initial condition $x_{0,1}$ and constant input u_1 . Any input that*

is order equivalent to u_1 generates the output y_1 from any initial condition that is output invariant with respect to $x_{0,1}$; hence, any order equivalence $\lambda : [0, t_1] \rightarrow [0, t_2]$ to a trajectory (u_2, y_2, x_2) (from initial condition $x_{0,2}$) is necessarily the identity order equivalence.

Proof. Since we’re considering inputs that have a singleton codomain, it is clear that the system outputs will necessarily be the same for all order-equivalent inputs, given output-invariant initial conditions. The conclusion about uniqueness of the identity order equivalence then follows from the non-constancy of the output and [Proposition 8](#). □

5.4.3 LTI Systems and Order-equivalence Non-determinism: Piecewise Analytic Inputs and “Global” Non-determinism Structure

In the previous subsection, we characterized order equivalences within the four different classes of input/output pairs that are possible with a purely analytic input. In this subsection, we will concatenate those “local” order equivalences into “global” order equivalences: that is we will consider which order equivalences are possible when we string together purely analytic inputs in order to make *piecewise analytic* inputs. This allows us characterize order equivalences between trajectories that can reach all of the nodes in a GST constructed according to [Section 5.3](#).

Importantly, the assumptions of [Section 5.3](#) allowed us to show that in all such cases but one – the constant-input/constant-output case – the only possible order equivalence is the identity order equivalence, $\lambda_{\text{id}} : t \mapsto t$. Of course the identity order

equivalence preserves output invariance of trajectory states, so given [Lemma 3](#), we conclude that all order equivalent trajectories started from output invariant states lead to output invariant states; hence, output invariance is an *invariant* with respect to order equivalence along analytic segments, at least under the assumptions we have made (i.e. **(A1)** - **(A3)**). Thus, given that the lemmas in the previous subsection are stated in terms of output invariant initial conditions, we have done essentially all of the hard work in characterizing such order equivalences: we need only observe that by construction, all trajectories emanating from a single GST node start in a single, specified initial condition, which is of course order equivalent to itself. Then we may proceed inductively along analytic segments, using the fact that output invariance is preserved in the terminal states of each. This concept is formalized in the following theorem; see also [Fig. 5.2](#).

Theorem 11 (The Character of Order Equivalences in LTI Systems). *Let $p = (t, (u, y, x)|_t) \in G_{x_0}(A, B, C)$ be a fixed node, and suppose that $q_1 = (t_1, (u_1, y_1, x_1)|_{t_1})$ and $q_2 = (t_2, (u_2, y_2, x_2)|_{t_2})$ are nodes for which $(p, q_1]$ and $(p, q_2]$ are order equivalent trajectories (from p). Then there exists an order equivalence $\lambda : [t, t_1] \rightarrow [t, t_2]$ such that for all $t' \in [t, t_1]$:*

- *if there exists an $\epsilon > 0$ and an interval $[t', t' + \epsilon)$ over which **both** u_1 is analytic **and** either u_1 or y_1 is non-constant, then $\lambda(\tau') - \lambda(\tau'') = \tau' - \tau''$ for all $\tau', \tau'' \in [t', t' + \epsilon)$.*

Proof. The proof proceeds along the lines suggested in the preceding paragraph. The input u_1 is piecewise analytic, so divide it into analytic segments over a sequence of

adjacent time intervals $J_i = [\mu_i, \mu_{i+1}]$ with $\mu_0 = t$. Then proceed by induction along the J_i .

The first analytic “segment” of input is either constant or non-constant, and leads to either a constant or non-constant output. In the case where either $u_1|_{J_0}$ or $y_1|_{J_0}$ are non-constant, then we can appeal to Lemmas [Lemma 4](#), [Lemma 5](#) or [Lemma 6](#) as necessary to show that the only possible order equivalence between $(u_1|_{J_0}, y_1|_{J_0})$ and (u_2, y_2) is the identity order equivalence; no matter what, then, at the end of the segment J_0 , the order equivalence pairs output invariant states from $x_1(\mu_1)$ and $x_2(\lambda(\mu_1))$. In the case where $u_1|_{J_0}$ or $y_1|_{J_0}$ are both constant, then the order equivalence may be selected arbitrarily as a monotonic bijection that spans the segments; by [Lemma 3](#), though, the resultant states are still output invariant.

Since $x_1(\mu_1)$ and $x_2(\lambda(\mu_1))$ are necessarily output invariant, we may use time-invariance of the underlying dynamics to repeat the above analysis on the next interval, J_1 , using $x_1(\mu_1)$ and $x_2(\lambda(\mu_1))$ as “initial conditions” at “time 0”. Given this fact, we may clearly proceed inductively through the rest of the intervals J_i using the same type of argument, and hence, we can assert the claim of the theorem for the entire trajectories (u_1, y_1, x_1) and (u_2, y_2, x_2) started from the specified time t . □

The claim of [Theorem 11](#) is illustrated in [Fig. 5.2](#) using cartoon trajectories: during non-constant trajectories, we see that all order equivalences preserve time differences through the (appropriately shifted) identity order equivalence, whereas during constant-input/constant-output trajectories, the choice of order equivalence

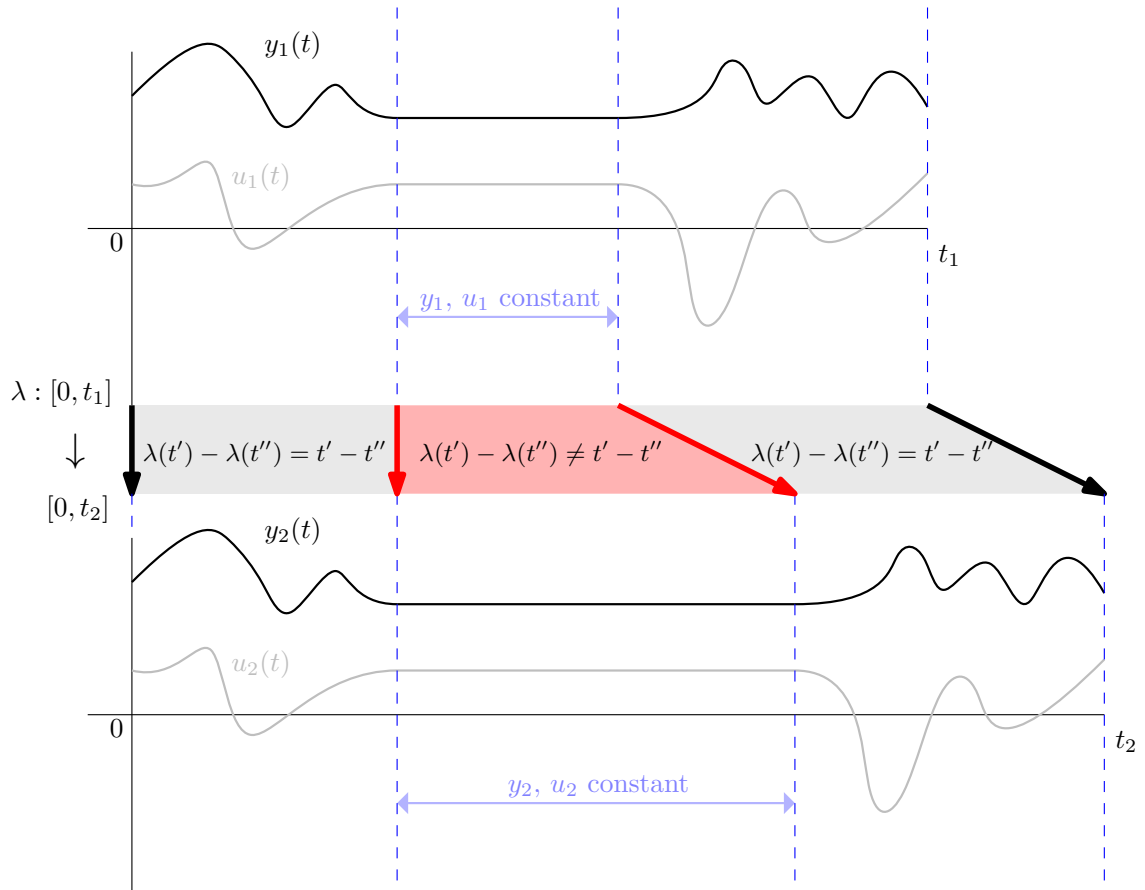


Figure 5.2: Cartoon of order equivalent LTI trajectories (single input/single output).

is essentially unconstrained. The fact that such constant-input/constant-output trajectories may appear in the middle of the response to an arbitrary piecewise analytic input means that *even trajectories that have known starting and ending states may have different order equivalences between them*. However, the character of these order equivalences is entirely determined by (the arbitrariness of) order equivalences along the constant-input/constant-output sub-segments of the trajectory. Indeed, given two order-equivalent trajectories with at least one such sub-segment, there are infinitely many ways to create order equivalences between them, each one of which is a different function from one trajectory to the other; on the other hand, one can achieve any particular one of these order equivalences by merely composing it with another order equivalence. This foreshadows in part the relevance of considering equivalence classes of modal executions that we have used as a foundation for GHML: recall [Definition 38](#) and [Definition 40](#).

Despite seriously constraining the potential order equivalences between LTI-system trajectories, [Theorem 11](#) still seems to permit a hopelessly intractable mess of trajectories that are comparable by order equivalence. There is, however, one feature of LTI systems/behaviors that we have given scant attention to as yet: these systems are, in their natural, timed context, entirely *deterministic*. Thus, when we construct a GST from an LTI system with a single, unambiguous initial condition, the problem is essentially reduced to the consideration of order equivalences between a trajectory and itself; given the above comments, *this* problem is suddenly a lot more tractable! In fact, this observation forms a useful starting point from which to begin describing the surrogate KSs that result from the LTI-derived GSTs: that

is the subject of the next section.

5.5 Modal Logic and Order-equivalence Induced Nondeterminism

In [Section 5.3](#), we considered an LTI system with carefully specified restrictions on its admissible inputs, and those restrictions allowed us to precisely characterize the nature of order equivalences between LTI system trajectories in [Section 5.4](#); see [Theorem 11](#) in particular. Given the nature of weak bisimulation and the related definition of GHML in terms of equivalence classes of functions over a domain of modalities (modulo order-equivalence, essentially), we now have the necessary prerequisites to describe the structure of a surrogate KS, $\mathbf{G}_{x_0}^{\mathcal{F}}(A, B, C)$, that results from an LTI-derived GST, $G_{x_0}^{\mathcal{F}}(A, B, C)$. Hence, in this section, we can provide a treatment of VHHM classes and their relationship LTI-system models as GSTs.

Throughout this section, we will work with the natural domain of modalities for LTI systems: for a total order, we will use the real number line with the usual ordering, $(\mathcal{I}, \preceq_{\mathcal{I}}) := (\mathbb{R}, \leq_{\mathbb{R}})$, and for a set of labels, we will use pairs of (instantaneous) inputs and outputs, $L := \mathbb{R}^m \times \mathbb{R}^{\ell}$. It is clear then that this choice captures any set of GSTs derived from a common LTI system; the use of $(\mathbb{R}, \leq_{\mathbb{R}})$ also means that modal executions share an obviously special relationship with trajectories of the underlying LTI systems. In general, then, we will consider a set of GSTs, \mathcal{U} , each element of which is comprised of a *subset* of behaviors from some $G_{x_0}^{\mathcal{F}}(A, B, C)$; that is we may consider any set of different control “programs” executed on (A, B, C) starting from *some* x_0 , so long as they all comply with **(A1)**-**(A3)** (under a common

\mathcal{F}). It is in this context that the following results will hold.

5.5.1 The Structure of Nondeterminism in LTI-derived Surrogate KSs

We begin by revisiting [Example 2](#) (see also [Fig. 5.1](#)), which was meant to serve in part as motivation for our current undertaking. Another way of framing the important lesson of that example is as follows: when a GST contains a trajectory that is order equivalent to a prefix of itself, that GST will have *nondeterminism* in its surrogate KS; for [Example 2](#), this nondeterminism is illustrated in [Fig. 5.1](#). In the succeeding paragraphs of [Section 5.4](#), we argued that constant-input/constant-output trajectories from an LTI system exhibit exactly the same kind of property: if both input and output are constant for some interval of time, then in essence, *trajectories over sub-intervals* are indistinguishable by order equivalence! This is straightforwardly reflected in terms of modal executions: after all, a modal execution is really nothing more than a “trajectory” over a domain of modalities, which can itself be seen as analogous to a *linear* GST (albeit without any notion of root); the semantics of GHML then are about matching such modal executions with GST trajectories by *order equivalence*. What is perhaps not quite as straightforward is how [Theorem 11](#) (using **(A1)** - **(A3)**) will affect the structure of an LTI-derived GST’s surrogate KS, $\mathbf{G}_{x_0}^{\mathcal{F}}(A, B, C)$. This structure is described in the following theorem, the main result of this subsection.

Theorem 12 (The Nondeterminism Structure of LTI-derived Surrogate KSs). *Let*

$G_{x_0}^{\mathcal{F}}(A, B, C)$ be as in [Definition 48](#), and let $\mathbf{G}_{x_0}^{\mathcal{F}}(A, B, C)$ be the associated surrogate KS over the domain of modalities $(\mathcal{I}, \preceq_{\mathcal{I}}) := (\mathbb{R}, \leq_{\mathbb{R}})$ and set of labels $L := \mathbb{R}^m \times \mathbb{R}^{\ell}$. Furthermore, let $p_1 = (t_1, (u_1, y_1, x_1)|_{t_1})$ and $p_2 = (t_2, (u_2, y_2, x_2)|_{t_2})$ be two distinct, non-root nodes in $G_{x_0}^{\mathcal{F}}$ (and hence, also states in the surrogate KS). Then, if there is a modal execution, $|E|$ such that $(p_0, p_1]$ and $(p_0, p_2]$ are order equivalent to $|E|$, then:

- there exists a common subinterval $[a, b] \subset [0, t_1]$ and $[a, b] \subset [0, t_2]$ on which both (u_1, y_1) and (u_2, y_2) are constant and equal; and
- there exists a time $\tau \in [a, b]$ such that $(u_1, y_1, x_1)|_{\tau} = (u_2, y_2, x_2)|_{\tau}$, and in particular, $x_1(\tau) = x_2(\tau)$ so that p_1 and p_2 share a common non-root ancestor.

Proof. This is more or less a consequence of [Theorem 11](#) and the fact that we are constructing GSTs from *deterministic* executions of an LTI system.

We begin by noting that two trajectories in $G_{x_0}^{\mathcal{F}}$ that are order equivalent to a common $|E|$ must necessarily be order equivalent to each other – simply compose the (invertible) order equivalences as needed. Hence, [Theorem 11](#) applies, and we know the nature of this order equivalence – and hence, how these trajectories “satisfy” $|E|$ – simply by the type of analytic segments comprising (u_1, y_1) and (u_2, y_2) . If the first analytic segment of (u_1, y_1) is not constant-input/constant-output, then we know the *only* possible order equivalence to another trajectory in the tree is the identity: this is because both trajectories start from the same, common initial condition and we are considering *deterministic* executions by construction. Thus, the first analytic segments of (u_1, y_1) and (u_2, y_2) are literally identical: that is

they go through the same *state* trajectory as well. Of course by [Theorem 11](#), this argument applies up to the first non-constant-input/constant-output segment of the trajectory, of which there must be at least one, lest the entire trajectories be identical (this would be impermissible by the assumed distinctness of the trajectory endpoints). This proves both the first claim and the second claim of the theorem. \square

Corollary 1. *Let $G_{x_0}^{\mathcal{F}}(A, B, C)$ be as in [Definition 48](#), only created with some subset of the behaviors from $\mathcal{B}_{x_0}^{\mathcal{F}}(A, B, C)$. Then the conclusions of [Theorem 12](#) still hold.*

Proof. [Theorem 12](#) depends on uniqueness of the identity order equivalence in certain situations and determinism of the underlying LTI system; these properties are retained when considering a subset of the specified set of behaviors. \square

With regard to these results, it is first worth emphasizing the crucial role played by *determinism* in the underlying behavioral system model: there is determinism not just in the LTI-system model itself but also in the trajectories (or executions) of said system. In particular, we have a priori ruled out the possibility of modeling uncertainty by means of nondeterministic behaviors, wherein the same input leads to *multiple* behaviors or execution trajectories. That is these models cannot use nondeterministic executions to encode some auxiliary “hidden” state that affects future branching behavior: only the *actual* system state and clock time can be used for that purpose. On the one hand, this may be seen as consistent with the determinism inherent in the LTI-system model, especially since we have as yet considered known, fixed initial conditions; on the other hand, this is perhaps a severe modeling constraint for CPSs, where the supplied inputs may conceivably be created or

influenced by, say, the execution of some complicated interactive computer program or indeed even another CPS. There are two counterpoints to this latter objection. For one, the abstraction of time – through weak bisimulation and GHML via order equivalence – converts a deterministic system into a nondeterministic one that has an inherently interesting structure, especially in terms of modal logic; allowing *other* types of nondeterminism merely obscures this structure, which is an interesting subject of study in itself. Second, though, we have advertised this as a contribution towards a *compositional* paradigm of modeling: in that sense, it seems reasonable to start with deterministic models – especially when those are standard and useful modeling primitives – and use *composition operators* to introduce nondeterminism as necessary. This is a limitation to be sure, but it is not inconsistent with a process algebraic approach. In that sense, it also ensures a certain level of formalism with which nondeterminism is introduced. And remember, GSTs as such have abstracted away the notion of real, “clock” time, so there is actually the hope that composition operators in this domain are easier to describe and far less “brittle”; see [15, 17] for example composition operators.

Now, to communicate the intuition behind the claims of [Theorem 12](#), [Fig. 5.3](#) illustrates the essence of the branching structure that it implies for a surrogate KS. [Fig. 5.3](#) thus depicts some cartoon LTI-system trajectories, along with their associated surrogate KS arcs, which are shown in green. There are three trajectories depicted: the upper and lower trajectories, (u_1, y_1, x_1) and (u_2, y_2, x_2) , are shown only for times after their “branching” point from the central trajectory, (u_0, y_0, x_0) ; according to [Theorem 12](#), this structure for the picture makes sense, as these tra-

jectories must necessarily share their initial non-constant trajectory segments. All three trajectories also share a common segment of non-constant input/output trajectory that starts after $t_0 + \Delta t_{b_0}$, $t_0 + \Delta t_{b_1}$ and $t_0 + \Delta t_{b_2}$, respectively and which lasts for a (common) duration of $t_1 - \Delta t_{b_1} - t_0$; however, these segments of trajectory *do not share state trajectories* because the branching has forced them to begin at different *times* relative to each other. Recall that the nodes in our tree are labeled with times as well as states! Indeed, the actual state values along these trajectories may not even be the same, although by previous arguments, order equivalent nodes will necessarily have output invariant states. Nevertheless, the surrogate KS will have identical arcs from the root of the tree to the nodes given by $(t'_0, (u_0, y_0, x_0)|_{t'_0})$, $(t_1, (u_1, y_1, x_1)|_{t_1})$ and $(t_2, (u_2, y_2, x_2)|_{t_2})$: these arcs are labeled by

$$|E| \triangleq |t \in [0, t'_0] \mapsto (u_2(t), y_2(t))|. \quad (5.49)$$

This is the constant-input/constant-output nondeterminism that we have claimed! To emphasize this point, [Fig. 5.3](#) also depicts surrogate KS arcs to intermediary nodes of interest: of particular note is the arc over constant portions of the respective trajectories,

$$|E_C| \triangleq |t \in [t_0, t_0 + \Delta t_{b_0}] \mapsto (u_0(t), y_0(t)) = (u_0(t_0), y_0(t_0))|. \quad (5.50)$$

In fact, the actual picture of surrogate KS arcs between the nodes connected by $|E_C|$ is exactly like the one depicted in [Fig. 5.1](#): this is an essential structure that we will study going forward! Finally, note the different *extensions* of the system trajectories depicted after times t'_0 , t_1 and t_2 , respectively: since the respective trajectories lead to different nodes in the GST at these times, permitting *subsets* of behaviors from

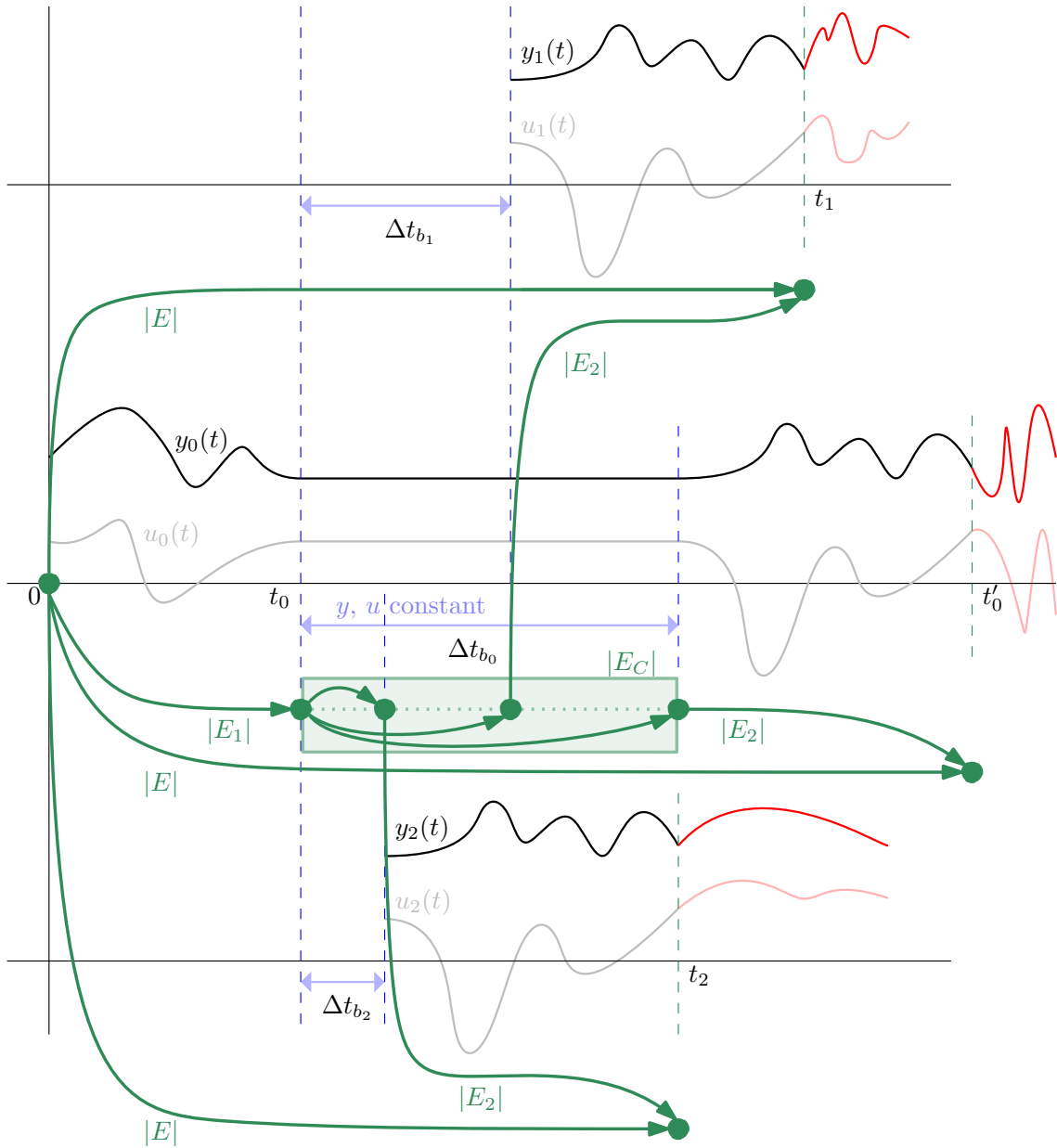


Figure 5.3: Cartoon of nondeterministic order-equivalent trajectories. Surrogate KS arcs (in green) pictorially “span” their associated trajectories; they have labels $|E|$, $|E_1|$, $|E_C|$ and $|E_2|$, which are defined in the text. $|E_C|$ -labeled arcs lie entirely in the shaded box.

$\mathcal{B}_{x_0}^{\mathcal{F}}$ – as in [Corollary 1](#) – means that we are allowing GSTs that have possibly different behavior *after* such a branching has occurred.

Despite its rather compact statement, then, [Theorem 12](#) somewhat narrowly specifies the branching structure of a surrogate KS $\mathbf{G}_{x_0}^{\mathcal{F}}(A, B, C)$. In fact, it can be summarized as saying that nondeterminism in $\mathbf{G}_{x_0}^{\mathcal{F}}(A, B, C)$ arises only *at or because of* nodes within stretches of constant-input/constant-output. This essential fact creates additional structure in the nondeterminism of the surrogate KS beyond the anticipated weak density and transitivity of [Theorem 10](#). In particular, as a (further) consequence of determinism in the underlying behavioral model, the nodes within a particular segment of constant-input/constant-output occur in a *linear order* with respect to their associated constant-function modal execution – a property called *right-linearity*. This is precisely the structure claimed for the $|E_C|$ transitions depicted in [Fig. 5.3](#) and the $|E|$ transitions depicted in [Fig. 5.1](#). The implications of this for modal logic and GHML, particularly with respect to VHHM classes, are significant: not only is this a type of nondeterminism, but it is also a type of “*infinitary*” nondeterminism. Thus, we make this the subject of the next subsection.

5.5.2 Quasi-discreteness and VHHM Classes for Weak Dense, Transitive, Right-linear Structures

Since we claim that *right linearity* [36] is induced by the LTI order-equivalence structure over constant-input/constant-output trajectories, we begin this section

with a definition of right-linearity, or as it is much more commonly called, *weak connectedness* [26].

Definition 49 (Right linearity/Weakly connected). *A transition relation $R \subseteq P \times P$ is said to be right-linear or weakly connected if it satisfies the following first-order property:*

- For all $s, t, u \in P$,

$$sRt \wedge sRu \implies (tRu \vee t=u \vee uRt). \quad (5.51)$$

Furthermore, this property is captured by the schema [37]:

$$\Box((A \wedge \Box A) \rightarrow B) \vee \Box((B \wedge \Box B) \rightarrow A) \quad (5.52)$$

or its equivalent, somewhat more intuitive schema [36]:

$$\Diamond A \wedge \Diamond B \rightarrow \Diamond(A \wedge \Diamond B) \vee \Diamond(A \wedge B) \vee \Diamond(\Diamond A \wedge B). \quad (5.53)$$

(This equivalence holds at least in the case of logics that are also transitive and weakly dense; see the following proposition.)

Proposition 9. *A transitive, weak-dense normal logic satisfies schema (5.52) if and only if it satisfies schema (5.53).*

Proof. The forward direction follows straightforwardly from the normal logic schema K .

For the reverse direction, simply observe:

$$\begin{aligned}
& \diamond A \wedge \diamond B \rightarrow \diamond(A \wedge \diamond B) \vee \diamond(A \wedge B) \vee \diamond(\diamond A \wedge B) \\
\implies & \quad \diamond(A \wedge B) \rightarrow \diamond((A \wedge \diamond B) \vee (A \wedge B) \vee (\diamond A \wedge B)) \\
\implies & \quad \Box(\neg(A \wedge \diamond B) \wedge \neg(A \wedge B) \wedge \neg(\diamond A \wedge B)) \rightarrow \Box\neg(A \wedge B) \\
\implies & \quad \Box\neg(A \wedge \diamond B) \wedge \Box\neg(A \wedge B) \wedge \Box\neg(\diamond A \wedge B) \rightarrow \Box\neg(A \wedge B) \\
\implies & \quad \Box\neg(A \wedge \diamond B) \wedge \Box\neg(\diamond A \wedge B) \rightarrow \Box\neg(A \wedge B) \\
\implies & \quad \Box(\neg(A \wedge \diamond B) \wedge \neg(\diamond A \wedge B)) \rightarrow \Box\neg(A \wedge B)
\end{aligned}$$

Now, expanding the argument of the box on the left using propositional distribution of \wedge over \vee , we get the following.

$$\implies \Box((\neg A \wedge \neg \diamond A) \vee (\neg B \wedge \neg \diamond B) \vee (\neg A \wedge \neg B) \vee (\neg \diamond A \wedge \neg \diamond B)) \rightarrow \Box\neg(A \wedge B).$$

The left-hand side of this expression can be strengthened by distributing the box across the disjunct to get:

$$\begin{aligned}
\implies & \quad \Box(\neg A \wedge \neg \diamond A) \vee \Box(\neg B \wedge \neg \diamond B) \vee \Box(\neg A \wedge \neg B) \vee \Box(\neg \diamond A \wedge \neg \diamond B) \\
& \quad \rightarrow \Box\neg(A \wedge B).
\end{aligned}$$

Now, the $\Box(\neg A \wedge \neg B)$ on the left is obviously redundant, and by transitivity and weak density, the term $\Box(\neg \diamond A \wedge \neg \diamond B)$ is likewise redundant, since:

$$\Box(\neg \diamond A \wedge \neg \diamond B) \leftrightarrow \Box\Box\neg A \wedge \Box\Box\neg B \leftrightarrow \Box\neg A \wedge \Box\neg B.$$

By strengthening the remaining disjunct on the left to a conjunction, we can continue

simplifying as follows.

$$\begin{aligned}
&\implies && \Box(\neg(A \vee \Diamond A) \wedge \neg(\Diamond B \vee B)) \rightarrow \Box\neg(A \wedge B) \\
&\implies && \Box\neg(A \vee \Diamond A) \wedge \Box\neg(\Diamond B \vee B) \rightarrow \Box\neg(A \wedge B) \\
&\implies && \Box\neg(A \vee \Diamond A) \wedge \Box\neg(\Diamond B \vee B) \rightarrow \Box\neg A \vee \Box\neg B \\
&\implies && \neg\Box\neg(A \vee \Diamond A) \vee \neg\Box\neg(\Diamond B \vee B) \vee \Box\neg A \vee \Box\neg B \\
&\implies && \neg\Box\neg(A \vee \Diamond A) \vee \Box\neg B \vee \neg\Box\neg(\Diamond B \vee B) \vee \Box\neg A \\
&\implies && (\Box\neg(A \vee \Diamond A) \rightarrow \Box\neg B) \vee (\Box\neg(\Diamond B \vee B) \rightarrow \Box\neg A)
\end{aligned}$$

The last of which is clearly equivalent to (5.52) when we allow A and B to span over all formulas. \square

In fact, as a consequence of using a *deterministic* behavioral model, LTI-derived GST nodes along constant-input/constant-output trajectories are in *correspondence* with real numbers (i.e. \mathbb{R}): this means that they actually satisfy a stronger property than *weak*-connectedness. This property, called *connectedness* [26], will of course *imply* right linearity/weak connectedness, but it will be of some special importance later when we consider VHHM classes.

Definition 50 (Connected [26]). *A transition relation is connected if it satisfies the following first-order property:*

- For all $t, u \in P$,

$$tRu \vee t=u \vee uRt . \tag{5.54}$$

Proposition 10. *The orderings $<$ and \leq are connected relations over the real number line, \mathbb{R} , and its open sub-intervals $(a, b) \subseteq \mathbb{R}$.*

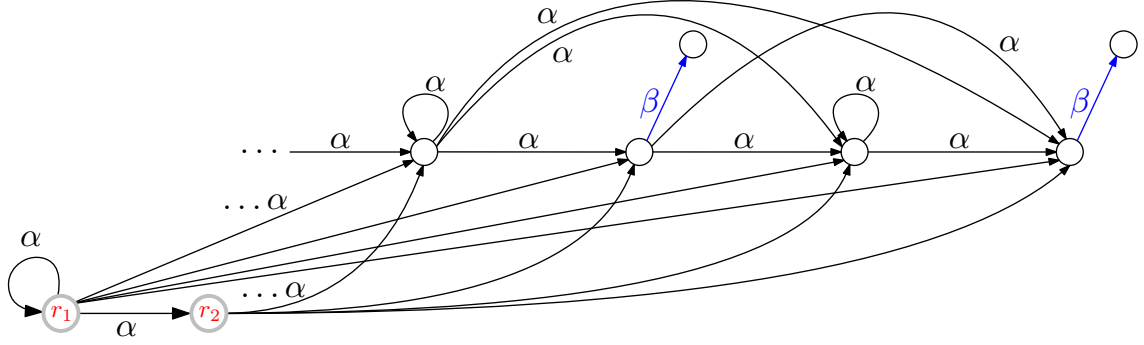


Figure 5.4: A right-linear, weakly dense, transitive example from [19] (note: the shading was altered on the second node from the left). Not shown are “transitive” $\alpha; \beta$ transitions. Two important nodes are labeled as r_1 and r_2 , respectively.

Now, as we did in Section 5.4, we will consider an informative example from [19]. In particular, consider the KS depicted in Fig. 5.4, which is transitive, weakly dense and right-linear with respect to the transition α . In [19], this KS was in fact (bisimilar to) the surrogate KS of a particular GST; see Example 6 in [19]. However, given the discussion in the previous section, we can also think of this KS as a hypothetical surrogate KS obtained from an LTI-derived GST *with a stretch of constant-input/constant-output trajectory*: the α transitions then stand-in for the modal execution of a constant-input/constant-output trajectory. In [19], the important observation about this KS is as follows: both gray nodes r_1 and r_2 satisfy the same formulas, yet they are clearly not bisimilar. Hence, this KS cannot belong to *any* VHHM class of KSs²; moreover, an α -labeled self-transition may be added to

²There is a slight overstatement in [19]: although Fig. 5.4 definitely meets these criteria, it is not exactly difficult to obtain KSs that don’t belong to *any* VHHM class. What is special about the KS in Fig. 5.4, though, is that it is an example that is weakly dense, transitive and right-linear. And those particular structures have special importance for surrogate KSs and hence, GSTs.

r_2 without changing the formulas satisfied by *any* of the nodes. It turns out that the proof of this latter fact is revealing, and it may be generalized. In particular, though, the ability to add a transition without changing formula satisfaction suggests immediately that we are dealing with a node that is *not* m-saturated. Recall [Theorem 2](#): VHHM classes are obtained by removing transitions from the canonical model (aka the Henkin model) to obtain Henkin-*like* models, and in particular, then the Henkin model itself identifies the *unique* m-saturated VHHM class.

The observations in the previous paragraph were present in [\[19\]](#) as claimed, but it is natural to ask the question: what is it about the example KS in [Fig. 5.4](#) that makes its node r_2 modally *un*-saturated? On the one hand, consider the infinite set of formulas:

$$\Theta = \left\{ \begin{array}{l} \theta_1 = \langle \alpha \rangle \langle \beta \rangle \top, \\ \theta_2 = \langle \alpha \rangle (\langle \beta \rangle \top \wedge \langle \alpha \rangle \langle \beta \rangle \top), \quad \left[= \langle \alpha \rangle (\langle \beta \rangle \top \wedge \theta_1) \right] \\ \theta_3 = \langle \alpha \rangle (\langle \beta \rangle \top \wedge \langle \alpha \rangle (\langle \beta \rangle \top \wedge \langle \alpha \rangle \langle \beta \rangle \top)), \quad \left[= \langle \alpha \rangle (\langle \beta \rangle \top \wedge \theta_2) \right] \\ \dots \\ \theta_k = \langle \alpha \rangle (\langle \beta \rangle \top \wedge \theta_{k-1}), \\ \dots \end{array} \right\}. \quad (5.55)$$

Each formula $\theta_k \in \Theta$ essentially describes the ability to do an α transition followed by the possibility of doing k different β transitions. Clearly, this set of formulas is finitely satisfied by node r_2 in [Fig. 5.4](#), but there is no α -*reachable* node that satisfies *all* of these formulas simultaneously. Hence, this KS is most certainly

not m-saturated. On the other hand, while the set of formulas, Θ , witnesses lack of modal saturation, it doesn't necessarily give us much of a hint about what transitions and nodes would need to be “added” to make it m-saturated. Recall that the set of formulas satisfied by r_2 is a maximally consistent set, and so corresponds to a node in the canonical model; the same is true of the other nodes. However, the set of formulas satisfied by node r_2 – when considered as a state in the canonical model – may have α transitions to *other* nodes in the canonical model, and indeed it must, since it's not m-saturated.

To get a sense of the additional nodes/transitions required to “saturate” [Fig. 5.4](#), we drew inspiration from [\[36\]](#) and a first-order transition-relation property therein called *quasi-discreteness*. The idea of quasi-discreteness appears there in the context of studying *determination* – an essentially “orthogonal” consideration to ours – but in the context of logics that are likewise transitive, weak dense and right-linear (also *serial*, although for our purposes, this will not prove relevant). Nevertheless, this property captures an essential feature of the structure in [Fig. 5.4](#), and so we will show that it has a particular relationship to m-saturation, and hence, VHHM classes; in particular, those VHHM classes related LTI-derived GSTs as they are constructed in the preceding.

Definition 51 (Quasi-Discreteness [\[36\]](#)). *A transition relation $R \subseteq P \times P$ is quasi-discrete if*

$$\forall s. \exists t. [sRt \wedge \forall u. (sRu \implies tRu)]. \quad (5.56)$$

First, note that quasi-discreteness as it appears in [\[36\]](#) implies *seriality* (as

indicated above): that is every node has at least one R -successor. For our purposes, we will relax this constraint by simply referring to particular *nodes* as being quasi-discrete rather than an entire KS (thus eliminating the universal quantifier on s in [Definition 51](#)); hence, we introduce our own restricted definition as follows.

Definition 52 (Quasi-Discrete Node). *A KS node, s , is quasi-discrete if it satisfies the predicate for the universally quantified variable s in (5.56).*

Thus, a node s may be described as quasi-discrete if it has an *immediate* (α -)successor: that is s has an α successor, t (possibly $t = s$) such that every α successor of s is also an α successor of t (if $t = s$ then s should have an α self-loop). This property is more intuitively understood in terms of total orders like $<$ on a subset $I \subseteq \mathbb{R}$: for that relation, a node (or real number), x , is quasi-discrete if the set $\{y \in I : x < y\}$ has a *minimal* element (in this example, only *isolated points* in I are quasi-discrete).

Now, the example in [Fig. 5.4](#) can be considered in more or less the same light as the ordering $<$ on \mathbb{R} if we think of α transitions as “ordering” nodes according to their position in the diagram, with each node coming “after” all of the nodes to its left. Thus, careful consideration of the node r_2 reveals that it is *not* quasi-discrete in the sense of [Definition 52](#): every one of its α successors fails to be an α successor of some other one. In this sense, then, node r_2 and its α successors may be regarded as a more or less fundamental illustration of non-quasi-discreteness, though the KS itself does have one superfluous feature in that regard. The KS in [Fig. 5.4](#) has the further special property that “between” any two successors of r_2 , there is always

a node that has an α self-loop and only α (or α -prefixed) outgoing and incoming transitions otherwise.

In this context, we can revisit the important concept in the proof that r_1 and r_2 satisfy the same HML formulas: in particular, that r_2 *with and without an α self-loop* satisfy the same formulas. This concept has an alternate interpretation in terms of *quasi-discreteness*: the *non-quasi-discrete* node r_2 can be made *quasi-discrete* without changing the formulas that it satisfies! This is the relationship between quasi-discreteness and m-saturation that we will develop here.

Theorem 13. *Let $\mathcal{K} = (S, \{R_\ell : \ell \in L\})$ be a KS and let R_α be a weakly dense, transitive and right-linear transition relation. Suppose a node w (with α successors) is not quasi-discrete with respect to α . Define two variants of \mathcal{K} that differ in the “vicinity” of the node w :*

- $\mathcal{K}_1 = (S_1, \{R_\ell^{(1)} : \ell \in L\})$ is the same as \mathcal{K} except that

$$wR_\alpha^{(1)}w;$$

and

- $\mathcal{K}_2 = (S_2, \{R_\ell^{(2)} : \ell \in L\})$ has states $S_2 = S \uplus \{\bar{w}\}$ (disjoint union) and for all $\ell \in L$, $R_\ell^{(2)} = R_\ell$ except for the following additions

(i) for any $\ell = \alpha$ or $\ell = \alpha'$ and any $t \in S = S_2 \setminus \{\bar{w}\}$,

$$wR_\ell t \implies \bar{w}R_\ell^{(2)}t;$$

(ii) for any $t \in S = S_2 \setminus \{\bar{w}\}$,

$$tR_\alpha w \implies tR_\alpha^{(2)}\bar{w};$$

(iii) for any $\ell \in L$ and $t \in S = S_2 \setminus \{\bar{w}\}$,

$$tR_\ell w \implies tR_{\ell;\alpha}^{(2)} \bar{w};$$

(iv) $wR_\alpha^{(2)} \bar{w}$;

(v) $\bar{w}R_\alpha^{(2)} \bar{w}$.

Then in at least one of \mathcal{K}_1 or \mathcal{K}_2 , the node w satisfies exactly the same formulas as it does in \mathcal{K} , and w is quasi-discrete with respect to α in that KS.

Proof. Whether w satisfies the same formulas in \mathcal{K}_1 or \mathcal{K}_2 depends on whether or not w has non- α -prefixed transitions emanating from it: if it has no such transitions, then \mathcal{K}_1 works; otherwise, \mathcal{K}_2 must be used. The proof in the latter case follows more or less directly from techniques used in the proof of the former, so we consider first the case when w has only α or α -prefixed transitions emanating from it.

In this first case, our task is to show that for any given formula $\phi \in \Phi_{\text{HML}}$, $w \models_{\mathcal{K}} \phi$ if and only if $w \models_{\mathcal{K}_1} \phi$. However, we're working in HML and $[\alpha; \ell]A \leftrightarrow [\alpha][\ell]A$ by transitivity *and* weak density, so this problem is actually equivalent to simpler one. In particular, it is enough to show that $w \models_{\mathcal{K}} [\alpha]\theta$ if and only if $w \models_{\mathcal{K}_1} [\alpha]\theta$, where all of the modalities in θ are of the form $[\alpha]$ or $[\ell]$ for $\ell \neq \alpha; \ell'$. We can further restrict the scope of the claim that needs to be proved by noting that $w \models_{\mathcal{K}_1} [\alpha]\theta$ implies $w \models_{\mathcal{K}} [\alpha]\theta$. This is by bisimilarity of the corresponding α successors to w in the models \mathcal{K} and \mathcal{K}_1 . Hence, we can further restrict ourselves to formulas $[\alpha]\theta$ for which $w \models_{\mathcal{K}} [\alpha]\theta$ and $w \not\models_{\mathcal{K}_1} [\alpha]\theta$, which taken together, further imply that $w \not\models_{\mathcal{K}_1} \theta$ (again by bisimilarity of the α successors to w across both models).

The strategy of the proof, then, will be to construct filtrations of \mathcal{K} and \mathcal{K}_1 using a common set of formulas that is derived from the (sub-formulas of) formula $[\alpha]\theta$, and then to show that both filtrations are bisimilar; this will be enough prove the claim, since bisimilarity preserves HML formulas, and the filtration lemma – see [26] – connects the truth of $[\alpha]\theta$ in the filtrations to its truth in their respective underlying KSs.

To construct useful filtrations, we first note that the (necessarily) *finite* formula θ contains at most finitely many distinct non- $[\alpha]$ modalities, say, $\{[\ell_1], [\ell_2], \dots, [\ell_N]\}$; from this list of modalities, construct the set of formulas $\Gamma_{[\phi_i]} := \{\perp, [\ell_1]\perp, [\ell_2]\perp, \dots, [\ell_N]\perp\}$. Then we will construct filtrations of \mathcal{K} and \mathcal{K}_1 from the set of formulas

$$\Gamma := \text{Sf}([\alpha]\theta) \cup \Gamma_{[\phi_i]} = \text{Sf}([\alpha]\theta) \cup \{\perp, [\ell_1]\perp, [\ell_2]\perp, \dots, [\ell_N]\perp\}, \quad (5.57)$$

where $\text{Sf}([\alpha]\theta)$ is the set of *sub-formulas* of $[\alpha]\theta$ (to include $[\alpha]\theta$ itself); clearly, Γ is closed under sub-formulas, as is required to construct Γ filtrations. Intuitively, the inclusion of the formulas $\Gamma_{[\phi_i]}$ will allow the filtration(s) to distinguish nodes that have different transition structures; we will use this fact in the future to reconcile this proof with others. Note also that $w \models_{\mathcal{K}} [\ell_1]\perp \wedge \dots \wedge [\ell_N]\perp$ and $w \models_{\mathcal{K}_1} [\ell_1]\perp \wedge \dots \wedge [\ell_N]\perp$, since in the case under consideration, w has no non- α (-prefixed) outgoing transitions in \mathcal{K} (and hence none in \mathcal{K}_1 , either).

Now, we will construct Γ filtrations of \mathcal{K} and \mathcal{K}_1 , although without loss of generality, we can restrict ourselves to the submodels thereof generated by w (see the submodel lemma in [26]); we will largely elide this distinction in our notation from here on. Importantly, though, we will consider in both cases the *largest* Γ -filtrations

with respect to R_α and $R_\alpha^{(1)}$, which we denote ${}^\lambda R_\alpha$ and ${}^\lambda R_\alpha^{(1)}$, respectively³; for all of the other transitions, we will use the *smallest* Γ -filtrations, ${}^\sigma R_\ell$ and ${}^\sigma R_\ell^{(1)}$. Recall the definitions of the largest and smallest filtrations from [26]:

$$|s| {}^\lambda R_\alpha |t| \quad \text{iff} \quad \forall B \in \Phi_{\text{HML}}. \left([\alpha]B \in \Gamma \text{ and } s \models_{\mathcal{K}} [\alpha]B \implies t \models_{\mathcal{K}} B \right) \quad (5.58)$$

$$|s| {}^\sigma R_\ell |t| \quad \text{iff} \quad \exists s' \in |s|, t' \in |t|. (s' R_\ell t') \quad (5.59)$$

where we have used the common notation $|s|$ to represent the state in the filtration associated with the (Γ equivalence class of) s . The Γ filtrations of \mathcal{K} and \mathcal{K}_1 so constructed will be denoted by \mathcal{FK} and \mathcal{FK}_1 , respectively. Henceforth, we will refer to the states from \mathcal{FK} using the bare notation $|s|$ and the states from \mathcal{FK}_1 using the notation $|s|_1$.

Now, according to the argument above, we need to establish a bisimulation relation between \mathcal{FK} and \mathcal{FK}_1 . It should be clear that the only problematic situation occurs in trying to match $|w|$ and $|w|_1$ in a bisimulation relation: this is because, by virtue of bisimilarity, any successor to w in \mathcal{K} satisfies the same formulas as its correspondent in \mathcal{K}_1 and vice-versa. Thus, we can easily begin to define a bisimulation relation between \mathcal{FK} and \mathcal{FK}_1 with the following:

$$w R_\alpha s \implies |s| \mathbf{B} |s|_1 \quad (5.60)$$

$$(w R_\alpha s \text{ and } s R_\ell t) \implies |t| \mathbf{B} |t|_1, \quad (5.61)$$

which together cover every node except w in \mathcal{K} (really its model generated by w).

³This notation roughly matches the notation in [26] for the largest/smallest filtration: λ for *largest* and σ for *smallest*. This λ has nothing to do with the order equivalences from before.

Furthermore, we note that, by construction,

$$\forall s \neq w : \quad wR_\alpha s \iff wR_\alpha^{(1)}s \quad (5.62)$$

$$\forall s, t \neq w : \quad (wR_\alpha s \text{ and } sR_\ell t) \iff (wR_\alpha^{(1)}s \text{ and } sR_\ell^{(1)}t), \quad (5.63)$$

so that \mathbf{B} – as it stands now – also covers every node except w in \mathcal{K}_1 (again, really the submodel of \mathcal{K}_1 generated by w). Now, in considering how to match $|w|$ and $|w|_1$, we have to consider two cases: the first when $w \models_{\mathcal{K}} \theta$ and the second when $w \not\models_{\mathcal{K}} \theta$. The latter is the more difficult case, and its consideration will reveal that simply adding $|w|\mathbf{B}|w|_1$ is sufficient to complete a bisimulation relation in the former.

The problem with the case $w \not\models_{\mathcal{K}} \theta$ is evident from the definition of the largest filtration in (5.58): $|w|_1$ will have an α self-loop in \mathcal{FK}_1 , since w has an α self-loop in \mathcal{K} ; but $|w|$ need *not* have an α self-loop in \mathcal{FK} since $[\alpha]\theta \in \Gamma$ and $w \models_{\mathcal{K}} [\alpha]\theta$ (by assumption) yet $w \not\models_{\mathcal{K}} \theta$, as is the defining assumption for this case (compare this observation to the case when $w \models_{\mathcal{K}} \theta$, as discussed above). This discrepancy in α self-loops for w means that we have to do a bit more work in order to have a bisimulation relation that includes $|w|\mathbf{B}|w|_1$. However, there is an important consequence of the assumption that \mathcal{K} is not quasi-discrete at w : since we have chosen a *finite* set of formulas, Γ , for our filtrations, \mathcal{FK} has only finitely many states (equivalence classes of states in \mathcal{K}), and so at least one equivalence class, say $|i|$, must contain *infinitely* many w -successors. However, this means that $|i|$ necessarily has an α self-loop, and indeed, we claim further that including $|i|\mathbf{B}|w|_1$ makes \mathbf{B} a bisimulation between \mathcal{FK} and \mathcal{FK}_1 . To see this, simply note that for any

α successor to w in \mathcal{K} , say $u : wR_\alpha u$, it is the case that u is an α successor to *some* element of $|i|$. This is by the lack of quasi-discreteness at w and the corresponding lack of an immediate successor. Hence, there will be an α transition between the corresponding states in the filtration, \mathcal{FK} : that is $|w|^\lambda R_\alpha |u|$. Of course this pattern also takes care of *all* of the transitions from w because we have assumed that w has only α or α -prefixed outgoing transitions. In particular, then, if we wish to show that the α transition from $|w|_1$ to itself is *simulated* in \mathcal{FK} , then we can use the transition $|w|^\lambda R_\alpha |i|$ as a witness to this simulation. Thus, for $|i|$ as defined above, we conclude that simply adding $|i|\mathbf{B}|w|_1$ to \mathbf{B} makes it a bisimulation between \mathcal{FK} and \mathcal{FK}_1 that includes $|w|\mathbf{B}|w|_1$; recall that we can assert the existence of such an $|i|$ by the pigeonhole argument above.

The preceding paragraph actually completes the proof that when w has only α or α -prefixed outgoing transitions, \mathcal{K}_1 meets the claimed conclusions. Importantly, though, the technique developed above works just as well when w has non- α -prefixed transitions. We can quickly dispense with formulas like $[\ell]\phi$, $\ell \neq \alpha$, since \mathcal{K}_2 simply adds extra α transitions from w : thus, the truth of $[\ell]\phi$ is unchanged between \mathcal{K} and \mathcal{K}_2 . To treat a formula like $[\alpha]\phi$, then, we simply augment $\Gamma_{\{\phi\}}$ with any of the non- α labels that appear in ϕ : as a result, we can continue matching $|w|\mathbf{B}|w|_2$, since the equivalence class of the new node, \bar{w} , will be bisimilar to some $|i|$, where $|i|$ is, as above, an equivalence class that includes infinitely many nodes. Following through on this observation completes the proof in the last remaining case. \square

As we have alluded to with examples, the intended use of [Theorem 13](#) is to

show that non-quasi-discreteness *can* lead to non-m-saturation. This is made precise in the following corollary.

Corollary 2. *Let \mathcal{K} and w be as in [Theorem 13](#). Suppose further that there exists an infinite sequence of formulas $\Theta_\infty = \{\theta_k\}$ such that for each $k \in \mathbb{N}$, there exists a w_k such that $wR_\alpha w_k$ and for all u , $w_k R_\alpha u \implies u \not\vdash_{\mathcal{K}} \theta_k$. Then \mathcal{K} is not modally saturated, and moreover, in the canonical model, $Fma(w)$ has either an α self-loop or else an α transition to $Fma(\bar{w})$, which itself has an α self-loop.*

Proof. [Theorem 13](#) explicitly defines new models in which w satisfies the same formulas, yet has extra transitions. Thus, since the canonical model is m-saturated, those extra transitions must be present as claimed. \square

The assertion that non-quasi-discreteness merely *can* lead to non-modal saturation is a very deliberate distinction: in fact, this is only the consequence when there is, as the statement of [Corollary 2](#) suggests, a “limit” point effect in the formulas satisfied by α accessible nodes. The following example makes this somewhat more explicit in a way that exploits properties of the real line. In particular, it shows that quasi-discreteness is irrelevant to m-saturation whenever there are only finitely many sets of formulas that are accessible from a non-quasi-discrete node; however, this is clearly a much stronger property than merely asserting that there are only finitely many labels, or even that the non- α structures are themselves bisimilar.

Example 3. *Consider the totally ordered interval $[0, 1] \subset \mathbb{R}$, and let $C \subset [0, 1]$ be the Cantor set; see [\[35\]](#). If for $x, y \in [0, 1]$, we relabel $x < y$ as $xR_\alpha y$, then R_α is a transitive, weakly dense and right-linear transition relation as discussed*

before. Now, consider amending this transition structure with a single β transition emanating from each element in C that is the endpoint of a “removed” open interval. Each endpoint in the resulting KS is thus not quasi-discrete (by the density of the real numbers). Nevertheless, the result is m -saturated (with respect to α transitions) because it is bisimilar to an image finite, quasi-discrete KS [27].

Another way to read [Example 3](#) is that non-quasi-discreteness only reflects on m -saturation when the original model is in some sense *irreducible to a quasi-discrete model*.

The full relationship between quasi-discreteness and m -saturation is yet more complicated. We can’t necessarily “saturate” a model by affecting only α transitions, either through the additional of a single α self-loop – as in \mathcal{K}_1 of [Theorem 13](#) – or through the addition of a node that essentially affects only α transitions – as in \bar{w} in \mathcal{K}_2 of [Theorem 13](#). As a consequence, there are models that may not be m -saturated by merely adding α transitions; an example of this can be derived easily from the KS in [Fig. 5.4](#).

Example 4. *Given the submodel generate by r_2 (from the KS of [Fig. 5.4](#)), define a new KS that deletes all α -self-loop nodes and replaces each β transition with k transitions β_1, \dots, β_k where k is the position of the originating, counted from the right of the diagram.*

The preceding example has the property that adding an α self-loop to r_2 actually changes the formulas satisfied by r_2 ! Intuitively, this suggests that to get to m -saturation by affecting *only* the quasi-discreteness of α transitions, some addi-

tional property is needed: in some sense, then, this is a reflection of the fact that m-saturation is a property that cannot be expressed merely in terms of the “topological” considerations of quasi-discreteness, even in weakly dense, transitive and right-linear structures. To modally saturate [Example 4](#), for example, there should be a single node with all of the transitions $\{\beta_k : k \in \mathbb{N}\}$, which is a requirement that clearly extends beyond mere quasi-discreteness and α transitions. This observation, like the notion of irreducibility suggested above, will be an important consideration with regard to the modal saturation of LTI-derived GSTs: it will be relevant particularly as we try to use trajectory specifications to induce m-saturated GSTs of LTI systems.

5.5.3 Implications for LTI Systems

In this section, we finally pull all of the threads together to say something about the relationship between LTI-derived GSTs and VHHM classes of GSTs.

On the one hand, we have shown that stretches of constant-input/constant-output trajectory lead to right-linear transition structures with respect to their associated (constant) modal executions. Then, in [Section 5.5.2](#), we connected the m-saturation of such structures to quasi-discreteness. Thus, the preceding subsections have established restrictions on how branching choices must be made at the beginning of – and within – such constant-input/constant-output segments of trajectory in order to achieve the m-saturation of their constituent nodes. It is thus natural to regard the m-saturation of nodes *within* a constant-input/constant-

output trajectory segment as “local” to that segment (compare with the terminology of [Section 5.4](#)). On the other hand, [Theorem 12](#) states that two distinct nodes that are reachable by order-equivalent trajectories need only have a common ancestor in some portion of constant-input/constant-output trajectory (in the absence of such a segment, there are no order equivalent trajectories that reach them, and their common ancestor is the root). Thus, *from the root node of the tree*, we have potential nondeterminism during and after each segment of constant-input/constant-output trajectory – see [Fig. 5.3](#); this nondeterminism “compounds” further with each segment of constant-input/constant-output trajectory that is included, as each such inclusion allows more and more different ways to achieve order equivalences. This type of nondeterminism is thus a sort of “global” nondeterminism, and it is an unavoidable consequence of the *transitivity* of the order-equivalence semantics: no matter which two trajectories are executed in sequence, there is necessarily a third that spans them.

Intuitively, global nondeterminism is the more complicated of the two to deal with: indeed, trajectories that long ago diverged from a common prefix trajectory may yet be responsible for identical transitions in the surrogate KS, and thereby be relevant for m-saturation and VHHM classes: as noted above, revisit [Fig. 5.3](#) for an illustration. And this fact is just as true of the root node as it is of *any other antecedents* of constant-input/constant-output trajectory segments. However, recall the amount of work that went into proving [Theorem 12](#): though its conclusion ultimately still allows “global” nondeterminism such as we have described, the situation would actually be much more complicated *without* that conclusion of **(A1)**-**(A3)**. A

subtle but invaluable convenience born of those assumptions is that nodes reachable by order-equivalent trajectories may be characterized entirely in terms of *output-invariance* ([Definition 44](#)). This is actually quite a non-trivial achievement, as it allows us to specify all pertinent branching structures in the surrogate KS *purely in terms of (clock) time and output-invariance classes of states!*

In light of this, the most straightforward way to control global nondeterminism and coerce membership in a VHHM class is to simply ensure that each output-invariant equivalence class of states has its own (common) branching behavior: that is the *same* (shifted) behaviors emanate from each and every one of its constituent states, irrespective of when any one of those states is traversed (in the tree). Or, more generally, one might propose a fixed and finite selection of such branching behaviors that are permissible in each such $[x]$, again agnostic of (clock) time. Ultimately, either scheme would result in that particular transition-label structure being bisimilar to an image finite tree, and hence, a member of every VHHM class [\[27\]](#). These ideas can be made precise using the following definitions, which suggest how common sets of behaviors may be combined with enforced non-determinism to obtain image-finite surrogate KS.

Definition 53 (Output-Invariance Branching Determiner). *An Output-Invariance Branching Determiner is a function*

$$\begin{aligned} \mathcal{D} : \mathcal{N}(\mathcal{O})^\perp &\rightarrow 2^{\{\mathcal{B}_x^{\mathcal{F}}(A,B,C) : x \in \mathbb{R}^n\}} \\ \mathcal{D} : [x] &\mapsto 2^{\mathcal{B}_{[x]}^{\mathcal{F}}(A,B,C)}. \end{aligned} \tag{5.64}$$

In particular, the codomain of such a \mathcal{D} is a set of behaviors starting from time 0,

as per our definition of $\mathcal{B}_x^{\mathcal{F}}$; see [Definition 47](#).

Now, we define what it means for an LTI-derived GST to be obtained by “choosing” from among a fixed selection of such branching determiners.

Definition 54 (Determiner-Finite LTI GST). *Let $\mathbb{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_N\}$ be a finite set of Output-Invariance Branching Determiner functions. Then we say that an LTI-derived set of behaviors – and its associated GST – is determiner finite (with respect to \mathbb{D}) if given any behavior (y, u, x) and time τ in said GST, the set of all behaviors that extend $(y, u, x)|_{\tau}$ is exactly one of the sets of behaviors $\mathcal{D}_i([x(\tau)])$ shifted by τ .*

The above definition is somewhat more restrictive than necessary, at least as far as our objective of getting bisimilarity with image-finite structures. Really, we could allow the number of branching determiner functions to vary based on the $[x]$ under consideration instead of using a common maximal integer N as in [Definition 54](#); this would complicate the notation with little conceptual gain, though.

Theorem 14 (Effective Image-finiteness of Non-constant Trajectories in Determiner-Finite LTI GSTs). *Given a GST $G_{x_0}^{\mathcal{F}}(A, B, C)$ that is determiner-finite with respect to \mathbb{D} , any modal execution label will necessarily have transition structure in $\mathbf{G}_{x_0}^{\mathcal{F}}(A, B, C)$ that is bisimilar to an image finite structure in that label.*

Proof. By the accumulated claims of [Lemma 3](#) through [Lemma 6](#), we know that all nodes reachable by order-equivalent trajectories are necessarily output invariant (with respect to each other). Then, since \mathbb{D} specifies only a finite number of different trees that these nodes may serve as the root of, we conclude that we may as well

consider only finitely many different modal executions with the appropriate label in the surrogate KS, at least as far as bisimulation is concerned. \square

Consider, however, the ramifications of [Definition 54](#) for an output-invariance class that lies along a trajectory of constant-input/constant-output. Since that invariance class is preserved along such a trajectory by [Lemma 3](#), the necessity of distributing finitely many “sets” of behaviors along such a trajectory means that: one, said constant-input/constant-output trajectory must be time infinite; and two, those sets of behaviors must be self-similar or “recursive” (in some sense) in order to cover the entire trajectory and still meet the definition. Thus, “choices” such as these are unavoidably *future-aware* in a very fundamental sense. This property is not terribly interesting in the context of LTI systems, which are often driven and controlled by “programs” that are only aware of the immediate state and time. Consider then a set of LTI behaviors created by a very simple “controller”: suppose that such a controller has only two input values available to it, and one – the initial one, say – results in a constant output trajectory. If that controller’s *second* input is enabled only at certain times – say, because of some (hypothetical) event trigger, perhaps one triggered by going through a *particular state* (as opposed to an output-invariance class of states) – then it is easy to imagine creating a structure like the KS in [Fig. 5.4](#). Here we have easily concocted a situation where local, state/time-based decision making leads to a somewhat more interesting structure, especially with regard to VHHM classes: after all, the lesson of the KS in [Fig. 5.4](#) is one on just that subject. Indeed, it’s possible for just such “controllers” to create an

LTI-derived GST that doesn't belong *any* VHHM class.

Though the results so far appear to be exclusionary – as in they specify structures that *can't* m-saturated – their ramifications are amplified by the determinism imposed on our LTI system's behaviors and the properties of the real number line. As argued above, we have to enforce a less restrictive notion on constant-input/constant-output trajectories than satisfying a branch determining function.

Definition 55 (Non-constant Branching Determiner Function). *Let \mathcal{D}^\sim be an output-invariance branching determiner function. We say that \mathcal{D}^\sim is a non-constant output-invariance branching determiner function if it has the further property that none of the behaviors in $\mathcal{D}^\sim([x])$ begin with constant-input/constant-output segments.*

Now we define a way of specifying *when* these non-constant determiner functions lead to branching from constant-input/constant-output segments of trajectory.

Definition 56 (Determiner Distribution). *A determiner distribution over a finite set of non-constant output-invariance branching determiner functions, $\mathbb{D}^\sim = \{\mathcal{D}_1^\sim, \dots, \mathcal{D}_N^\sim\}$, is a function*

$$\mathcal{E}_{y,[x],u} : [0, 1) \subset \mathbb{R} \rightarrow \{0\} \cup \{1, \dots, N\} \quad (5.65)$$

indexed by $[x] \in \mathcal{N}(\mathcal{O})^\perp$, $y \in \mathbb{R}^\ell$ and $u \in \mathbb{R}^m$; moreover, $\mathcal{E}_{y,[x],u}$ must satisfy the property that

- *for any $x, y \in \mathcal{E}_{y,[x],u}^{-1}(\{1, \dots, N\})$, there exists an open interval $(a, b) \subseteq (x, y)$ such that $\mathcal{E}_{y,[x],u}((a, b)) \subseteq \{0\}$; i.e. $\mathcal{E}_{y,[x],u}$ maps each element of (a, b) to 0.*

And we redefine what it means to have a determiner-finite GST in terms of non-constant determiner functions and determiner distributions.

Definition 57 (Determiner-Distribution-Finite LTI GST). *Given an (indexed collection of) determiner distributions $\mathcal{E}_{y,[x],u}$, we say that an LTI-derived set of behaviors – and its associated GST – is determiner-distribution finite if given any behavior (y, u, x) where $[t_1, t_2) \subset \mathbb{R}$ is any maximal interval of constant-input/constant-output trajectory segment therein, then*

- *there is a monotonic bijection $\lambda : [t_1, t_2) \rightarrow [0, 1)$ such that for all $\tau \in [t_1, t_2)$*
 - *if $\mathcal{E}_{y,[x],u}(\lambda(\tau)) \neq 0$, then each and every extension of $(y(\tau), u(\tau), x(\tau))$ that begins with a non-constant-input/constant-output segment is in $\mathcal{E}_{y,[x],u}(\lambda(\tau))$ and the only constant-input/constant-output trajectory emanating from $(y(\tau), u(\tau), x(\tau))$ as constant values $(y(\tau), u(\tau))$*
 - *otherwise the only trajectory emanating from $(y(\tau), u(\tau), x(\tau))$ is $(y', u', x') = (y(\tau), u(\tau), x(\tau)) = \text{const.}$*

For nodes with no constant-input/constant-output trajectory emanating from them, $\mathcal{E}_{y,[x],u}(0)$ specifies the behaviors.

These tedious and somewhat elaborate definitions are actually relatively straightforward generalizations of their earlier counterparts, only now able to represent examples like the hypothetical controller example above. Importantly, we now have a formalism within which to begin treating more interesting issues of VHHM membership.

Theorem 15. *Suppose there is a $k \in \{1, \dots, N\}$ for which $\mathcal{E}_{y,[x],u}^{-1}(k)$ has an accumulation point, \bar{x} . If every interval $(0, b) \ni \bar{x}$ excludes all but finitely many elements of $\mathcal{E}_{y,[x],u}^{-1}(k)$, then **no LTI GST that is determiner-distribution finite by $\mathcal{E}_{y,[x],u}$ is modally saturated.***

Proof. The important observation here is that modal formulas will be able to “count” the number of \mathcal{D}_k^\sim branches that are remaining in the trajectory, so this structure meets the hypotheses of [Corollary 2](#). In particular, the Henkin model has an extra transition from $\text{Fma}(\bar{x})$ to some other node with a self-loop. \square

Much more importantly, though, we note that we can *never* modally saturate such structures within LTI GSTs as we have defined them! Indeed, the addition of a node that has a constant-modal-execution self-loop can only be represented by an (additional) open sub-interval of \mathbb{R} within which no other branching is possible. However, this is impossible while still retaining the assumed accumulation point \bar{x} ! On the other hand, we note that not being m-saturated doesn’t imply exclusion from a VHHM class, so indeed such structures *may* belong to some VHHM class of GSTs. However, [Fig. 5.4](#) and its KS are again relevant: in the context of determiner-distribution finite tree, this aforementioned KS indicates what sort of constant-input/constant-output branching structure eludes *all* VHHM classes. In particular, it suggests that a prototypical example is a constant-input/constant-output trajectory with a *delay* – literally a fixed time delay – during which no branching is possible that then abuts an accumulation point like \bar{x} from the proof of [Theorem 15](#)!

Chapter 6: Congruence Results

In this chapter, we will consider three different *congruence* results. By “congruence result” we mean the following: given the composition of several component sub-systems, any one of the sub-systems may be substituted for another *bisimilar* system, and the new composition will be *bisimilar* to the original one.

6.1 Synchronous Interconnection of GSTs Derived from Behavioral Systems

In [25], Julius et. al. defined a composition operator between behavioral systems, and proved that bisimulation (Definition 11) is a congruence for that particular operator. We repeat these results from [25] below.

Definition 58 (Composition of behavioral systems [25]). *Let Σ_1 and Σ_2 be two behavioral systems with associated state maps \mathfrak{x}_1 and \mathfrak{x}_2 , as in, say Lemma 1. Then the (synchronous) composition of $(\Sigma_1, \mathfrak{x}_1)$ and $(\Sigma_2, \mathfrak{x}_2)$ is denoted by $(\Sigma, \mathfrak{x}) := (\Sigma_1, \mathfrak{x}_1) || (\Sigma_2, \mathfrak{x}_2)$, where*

1. $\Sigma = (\mathbb{T}, \mathbb{V} \times \mathbb{D}_1 \times \mathbb{D}_2, \mathcal{B})$;
2. $\mathcal{B} = \{(v, d_1, d_2) | (v, d_1) \in \mathcal{B}_1 \wedge (v, d_2) \in \mathcal{B}_2\}$; and

$$3. \mathfrak{r}(\langle v, d_1, d_2 \rangle, t) := \langle \mathfrak{r}_1(\langle v, d_1 \rangle, t), \mathfrak{r}_2(\langle v, d_2 \rangle, t) \rangle.$$

As a consequence of this composition operator, Julius et. al. prove the following congruence theorem [25].

Theorem 16 (Composition for behavioral systems is a congruence [25]). *For $i = 1, 2, 3$, let $(\Sigma_i, \mathfrak{r}_i)$ be behavioral state systems with associated state maps as in Definition 58. Then*

$$\Sigma_1 \mathfrak{r}_1 \sim_{\mathfrak{r}_2} \Sigma_2 \implies (\Sigma_1, \mathfrak{r}_1) \parallel (\Sigma_3, \mathfrak{r}_3) \sim_{\mathfrak{r}_1; \mathfrak{r}_3} \sim_{\mathfrak{r}_2; \mathfrak{r}_3} (\Sigma_2, \mathfrak{r}_2) \parallel (\Sigma_3, \mathfrak{r}_3). \quad (6.1)$$

Where $\mathfrak{r}_1; \mathfrak{r}_3$ is notation for the composed state map of Definition 58.

It is natural to ask whether or not there is a similar composition operator for GSTs that likewise exhibits this congruence property. For GSTs constructed from behavioral systems, we define such a composition operator below, and subsequently prove that strong bisimulation (with the time-shift property) is indeed a congruence.

Definition 59 (Synchronous composition of GSTs derived from behavioral systems). *Let GSTs \mathcal{G}_1 and \mathcal{G}_2 be GSTs derived from $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ according to Definition 33. We define the composition $\mathcal{G}_1 \parallel \mathcal{G}_2 = (P_{1;2}, p_{01;02}, \preceq_{1;2}, \mathcal{L}_{1;2})$ as follows:*

- $P_{1;2} := \{ \langle \langle v, d_1, d_2 \rangle^t, t \rangle \mid \langle \langle v, d_1 \rangle^t, t \rangle \in P_1 \wedge \langle \langle v, d_2 \rangle^t, t \rangle \in P_2 \} \cup \{ \langle p_{01}, p_{02} \rangle \};$
- $p_{01;02} = \langle p_{01}, p_{02} \rangle;$
- $\langle \langle v, d_1, d_2 \rangle^t, t \rangle \preceq_{1;2} \langle \langle v', d'_1, d'_2 \rangle^{t'}, t' \rangle$ if and only if $\langle v', d'_1, d'_2 \rangle^{t'}|_t = \langle v, d_1, d_2 \rangle^t;$
 $\langle p_{01}, p_{02} \rangle \preceq_{1;2} \langle \langle v, d_1, d_2 \rangle^t, t \rangle \forall \langle \langle v, d_1, d_2 \rangle^t, t \rangle \in P_{1;2};$ and
- $\mathcal{L}_{1;2}(\langle \langle v, d_1, d_2 \rangle^t, t \rangle) := \pi_{\forall}(\langle \langle v, d_1, d_2 \rangle^t, t \rangle).$

Given this definition of composition, we can establish an easy congruence result if the alternate GST is strongly bisimilar to its replacement via a bisimulation relation with the time-shift property.

Theorem 17. *For $i = 1, 2, 3$, let \mathcal{G}_i be GSTs derived from the respective behavioral state systems $(\Sigma_i, \mathfrak{r}_i)$. Furthermore, let $\mathcal{G}_1 \sim \mathcal{G}_2$ be strongly bisimilar with a bisimulation relation \sim that has the time-shift property. Then $\mathcal{G}_1 || \mathcal{G}_3$ and $\mathcal{G}_2 || \mathcal{G}_3$ are strongly bisimilar (and are related by a bisimulation relation with the time-shift property).*

Proof. The proof is an easy consequence of [Theorem 7](#). In particular, because \mathcal{G}_1 and \mathcal{G}_2 are strongly bisimilar with the time-shift property, [Theorem 7](#) asserts that $(\Sigma_1, \mathfrak{r}_1)$ and $(\Sigma_2, \mathfrak{r}_2)$ are bisimilar. However, by construction, the (non-root) nodes of $\mathcal{G}_1 || \mathcal{G}_3$ are exactly the states of the canonical partial order map $\mathfrak{p}_{\Sigma_1 || \Sigma_3}$, and likewise for $\mathcal{G}_2 || \mathcal{G}_3$ and $\mathfrak{p}_{\Sigma_2 || \Sigma_3}$. That is $\mathcal{G}_1 || \mathcal{G}_3$ is exactly the GST constructed from the composition $(\Sigma_1, \mathfrak{r}_1) || (\Sigma_3, \mathfrak{r}_3)$, and likewise for $\mathcal{G}_2 || \mathcal{G}_3$ and $(\Sigma_2, \mathfrak{r}_2) || (\Sigma_3, \mathfrak{r}_3)$. Since bisimulation is congruence for behavioral systems ([Theorem 16](#)), applying [Theorem 7](#) again gives the claimed result. \square

The proof of [Theorem 17](#) depends on having a strong bisimulation with the time-shift property, but this is not a severe restriction, since we are working exclusively with GSTs that are derived from behavioral systems. However, as [Example 1](#) shows, it is not a trivial or superfluous requirement, either.

Interestingly, the composition operator defined in [Definition 59](#) looks roughly like a special case of the CSP-parallel composition operator defined in [\[15\]](#). That

operator, however, is not exclusive to GSTs derived from behavioral systems, and so is more general at the expense of being time “unaware”. [Example 1](#) suggests that time-unaware notions of composition are unlikely to be comparable to bisimulation for behavioral systems, but the congruence properties of that CSP-parallel operator are nevertheless interesting and the subject of future work.

6.2 CSP-type Parallel Composition

6.2.1 Preliminaries

First, we recall the definition of our CSP-parallel composition operator for GSTs from [\[15\]](#), which is written in terms of *S-synchronized interleavings of trajectories*: see [Definition 27](#).

Definition 60 (S-synchronized Interleaving). *Let $G_1 = (P_1, \preceq_1, p_1, \mathcal{L}_1)$ and $G_2 = (P_2, \preceq_2, p_2, \mathcal{L}_2)$ be GSTs, and WLOG assume that $P_1 \cap P_2 = \emptyset$. Also let T_1 and T_2 be trajectories (cf. [Definition 27](#)) from the roots of G_1 and G_2 , respectively. Also let $S \subseteq L$. Then total order (Q, \preceq_Q) is an S-synchronized interleaving of T_1 and T_2 iff there exists a monotonic bijection $\lambda \in \{p \in T_1 : \mathcal{L}_1(p) \in S\} \rightarrow \{p \in T_2 : \mathcal{L}_2(p) \in S\}$ such that the following hold.*

1. $\mathcal{L}_1(p) = \mathcal{L}_2(\lambda(p))$ for all $p \in T_1$ such that $\mathcal{L}_1(p) \in S$.
2. $Q = \{p \in T_1 : \mathcal{L}_1(p) \notin S\} \cup \{p \in T_2 : \mathcal{L}_2(p) \notin S\} \cup \{(p, \lambda(p)) : \mathcal{L}_1(p) \in S\}$.
3. Define $\pi_1 \in Q \rightarrow (T_1 \cup T_2)$ by $\pi_1(p) = p$ if $p \in T_1 \cup T_2$ and $\pi_1((p'_1, p'_2)) = p'_1$ otherwise, and similarly for π_2 . Then, $\pi_1(q) \preceq_1 \pi_1(q')$ or $\pi_2(q) \preceq_2 \pi_2(q')$

implies $q \preceq_Q q'$.

We write $I_S(T_1, T_2)$ for the set of S -synchronized interleavings of T_1 and T_2 .

We thus have the following definition of a CSP-parallel composition operator for GSTs.

Definition 61. Let $G_1 = (P_1, \preceq_1, p_1, \mathcal{L}_1)$ and $G_2 = (P_2, \preceq_2, p_2, \mathcal{L}_2)$ be GSTs with $P_1 \cap P_2 = \emptyset$. Then the GST $G_1 |S| G_2 = (Q, \preceq_Q, q_0, \mathcal{L}_Q)$ is given by:

1. $Q = \{(p_1, p_2)\} \cup \{T : T \in I_S(T_1, T_2) \text{ for some trajectories } T_1 = (p_1, p'_1] \text{ of } G_1, T_2 = (p_2, p'_2] \text{ of } G_2 \text{ with } T_1 \text{ and } T_2 \text{ not both empty.}\}$.
2. $q \preceq_Q q'$ iff $q = (p_1, p_2)$, or $q = (r, \preceq_r)$, $q' = (r', \preceq_{r'})$, and \preceq_r is a prefix of $\preceq_{r'}$.
3. $q_0 = (p_1, p_2)$.
4. Let $q \in Q$ and let $p' = \sup(q)$. Then define \mathcal{L}_Q according to

$$\mathcal{L}_Q(q) = \begin{cases} \mathcal{L}_1(p') & \text{if } p' \in P_1 \\ \mathcal{L}_2(p') & \text{if } p' \in P_2 \\ \mathcal{L}_2(p'_1) & \text{if } p' = (p'_1, p'_2) \end{cases}$$

6.2.2 Congruence and Weak Bisimulation

We begin our consideration of weak bisimulation with an observation about the relationship between trajectories in the CSP-parallel composition of two GSTs and trajectories in the composed GSTs themselves.

Lemma 7. For $i = 1, 2$, let $G_i = (P_i, \preceq_i, p_i, \mathcal{L}_i)$ be GSTs, and let $C_1 = G_1|S|G_2 = (Q_1, \preceq_{Q_1}, q_{01}, \mathcal{L}_{Q_1})$ be the CSP-parallel composition of G_1 and G_2 on a set of synchronizing labels S . If $q_1 = (r_1, \preceq_{r_1})$ is a non-root node in C_1 where r_1 is obtained from the interleaving of right-closed trajectories $T_{G_1}^{q_1} \subseteq P_1$ and $T_{G_2}^{q_1} \subseteq P_2$, no more than one of which is empty, then the interval $(q_{01}, q_1] \subseteq Q_1$ is order equivalent to the set $r'_1 = \{p \in r_1 : \sup \pi_1((q_{01}, p]) \text{ and } \sup \pi_2((q_{01}, p]) \text{ exist}\}$, when ordered by \preceq_{r_1} .

Proof. The essential observation is that any $q' = (r', \preceq_{r'}) \preceq_{Q_1} q_1$ is itself an S-synchronized interleaving of two right-closed trajectories $T_{G_1}^{q'}$ and $T_{G_2}^{q'}$ that are necessarily prefixes of the trajectories $T_{G_1}^{q_1}$ and $T_{G_2}^{q_1}$, respectively. This follows directly from the definition of the tree partial order \preceq_{Q_1} in terms of *prefixes* (see condition 2 in [Definition 61](#)). Since such an r' necessarily has a maximal element – and indeed $\sup \pi_1(r')$ and $\sup \pi_2(r')$ both exist – it can be injectively matched to a unique node in the interleaving $q_1 = (r_1, \preceq_{r_1})$ of which it is a prefix. Conversely, if any element of the total order (r_1, \preceq_{r_1}) , say p , has the property that $\{p' \in r_1 : p' \preceq_{r_1} p\}$ has a supremum after projection through each of π_1 and π_2 , then it corresponds to one and only one S-synchronized interleaving (of the appropriate right-closed trajectories). \square

We note, however, that such an interval $(q_{01}, q_1]$ need not be order equivalent to the *entire* interleaving $q_1 = (r_1, \preceq_{r_1})$. This is clarified in the subsequent example.

Example 5. Let $G_1 = ([0, 1) \cup (1, 2], \leq, 0, \mathcal{L}_1 : x \mapsto x)$ and $G_2 = ([0, 1], \leq, 0, \mathcal{L}_2 : x \mapsto x)$ be two GSTs, and consider the CSP-parallel composition $C_1 = G_1|[0, 1)|G_2$ (that is the set of synchronizing labels is the right-open interval $[0, 1) \subset \mathbb{R}$). Furthermore, let $q = (r, \preceq_r)$ be the S-synchronized interleaving of the trajectory $[0, 1.5]$ in

G_1 (the interval being defined according to the partial order for G_1) and the trajectory $[0, 1]$ in G_2 that places the node 1 from G_2 in the gap between $[0, 1]$ and $(1, 1.5]$ in G_1 . That is (r, \preceq_r) is isomorphic to the interval $[0, 1.5] \subset \mathbb{R}$. We claim that the set of ancestors of $q = (r, \preceq_r)$ in C_1 , i.e. $[0, q)$, is in fact order equivalent to $(r \setminus \{1\}, \preceq_r)$.

This example seems innocuous, but it indicates an important consequence of [Lemma 7](#) in terms of congruence properties. This will be revisited in [Section 6.2.4](#), but for now, we proceed to prove the first of two congruence properties to be considered.

Theorem 18 (Weak Bisimulation is a congruence for CSP-parallel composition).

For $i = 1, 2, 3$, let $G_i = (P_i, \preceq_i, p_i, \mathcal{L}_i)$ be GSTs such that G_1 and G_2 are weakly bisimilar to each other. Then $C_1 := G_1|S|G_3$ is weakly bisimilar to $C_2 := G_2|S|G_3$.

Proof. The proof consists of establishing a weak bisimulation relation between $C_1 = (Q_1, \preceq_{Q_1}, q_{01}, \mathcal{L}_{Q_1})$ and $C_2 = (Q_2, \preceq_{Q_2}, q_{02}, \mathcal{L}_{Q_2})$. To this end, let $q_1 = (r_1, \preceq_{r_1})$ be a (non-root) node in C_1 , and let $q_2 = (r_2, \preceq_{r_2})$ be a (non-root) node in C_2 . By definition then, q_1 is an S-synchronized interleaving of some trajectories $T_{G_1}^{q_1} \subseteq P_1$ and $T_{G_3}^{q_1} \subseteq P_3$ from the roots of G_1 and G_3 , respectively, and q_2 is an S-synchronized interleaving of some trajectories $T_{G_2}^{q_2} \subseteq P_2$ and $T_{G_3}^{q_2} \subseteq P_3$ from the roots of G_2 and G_3 , respectively. We may assume from part 1 on [Definition 61](#) that at least one of $T_{G_1}^{q_1}$ and $T_{G_3}^{q_1}$ is non-empty, and likewise for $T_{G_2}^{q_2}$ and $T_{G_3}^{q_2}$. Furthermore, let $\sim_{1,2}$ be a weak bisimulation relation between G_1 and G_2 that includes (p_1, p_2) , the pair of root nodes from each.

Now we propose a candidate bisimulation relation, $\sim \subseteq Q_1 \times Q_2$, between the nodes in C_1 and C_2 . Unsurprisingly, we start by insisting that $(q_{01}, q_{02}) \in \sim$. We specify the remaining elements of \sim according to a set of pairs defined by the following condition:

$$\begin{aligned}
q_1 \sim q_2 \text{ iff} \\
& \left((T_{G_1}^{q_1} \neq \emptyset \wedge T_{G_2}^{q_2} \neq \emptyset) \vee (T_{G_3}^{q_1} \neq \emptyset \wedge T_{G_3}^{q_2} \neq \emptyset) \right) \wedge \\
& \left((T_{G_1}^{q_1} \neq \emptyset \wedge T_{G_2}^{q_2} \neq \emptyset) \implies (\pi_1(\text{sup } T_{G_1}^{q_1}) \sim_{1,2} \pi_1(\text{sup } T_{G_2}^{q_2})) \right) \wedge \\
& \left((T_{G_3}^{q_1} \neq \emptyset \wedge T_{G_3}^{q_2} \neq \emptyset) \implies (\pi_2(\text{sup } T_{G_3}^{q_1}) = \pi_2(\text{sup } T_{G_3}^{q_2})) \right). \tag{6.2}
\end{aligned}$$

That is non-root nodes q_1 and q_2 are related according to \sim if and only if the endpoints of their constituent trajectories from G_3 are identical **and** the endpoints of their constituent trajectories from G_1 and G_2 are bisimilar according to $\sim_{1,2}$. The first conjunct in (6.2) ensures that interleavings with empty trajectories from one operand GST do not get matched to interleavings with non-empty trajectories from the corresponding operand.

We claim that \sim as defined above is a weak bisimulation relation. We will prove the bisimulation property only for the case when $T_{G_1}^{q_1} \subseteq P_1$ and $T_{G_2}^{q_2} \subseteq P_2$ are both non-empty, and both $\pi_1(\text{sup } r_1) = \text{sup } T_{G_1}^{q_1} \in P_1$ and $\pi_1(\text{sup } r_2) = \text{sup } T_{G_2}^{q_2} \in P_2$. The other cases will have a similar proof.

To that end, suppose that $q_1 \sim q_2$ (with the aforementioned properties) and $q'_1 \succeq_{Q_1} q_1$, where $q'_1 = (r'_1, \preceq_{r'_1})$ is the interleaving of trajectories $T_{G_1}^{q'_1} \subseteq P_1$ and $T_{G_3}^{q'_1} \subseteq P_3$; we will show that there exists a $q'_2 \succeq_{Q_2} q_2$ such that $q'_1 \sim q'_2$ and $(q_1, q'_1]$ is order equivalent to $(q_2, q'_2]$. We will obtain the requisite q'_2 by replacing elements in

the totally ordered set $(r'_1 \setminus r_1, \preceq_{r'_1})$ with elements from trajectories that extend $T_{G_2}^{q_2}$ and $T_{G_3}^{q_2}$; the result will then extend the total order $q_2 = (r_2, \preceq_{r_2})$ as an interleaving of trajectories from G_2 and G_3 , thus providing a $q'_2 \succeq_{Q_2} q_2$ that has the desired properties.

By definition of \succeq_{Q_1} , we can say that $T_{G_1}^{q_1}$ is a prefix of $T_{G_1}^{q'_1}$ and $T_{G_3}^{q_1}$ is a prefix of $T_{G_3}^{q'_1}$. Moreover, because $\pi_1(\sup T_{G_1}^{q_1}) \sim_{1,2} \pi_1(\sup T_{G_2}^{q_2})$, there exists a trajectory from the root of G_2 – call it $T_{G_2}^{q'_2}$ – that extends $T_{G_2}^{q_2}$ and such that:

- $T_{G_2}^{q'_2} \setminus T_{G_2}^{q_2}$ is order equivalent to the trajectory $T_{G_1}^{q'_1} \setminus T_{G_1}^{q_1}$ in G_1 , which starts from $\pi_1(\sup r_1) = \pi_1(\sup T_{G_1}^{q_1})$; and
- $\pi_1(\sup T_{G_1}^{q'_1}) \sim_{1,2} \pi_1(\sup T_{G_2}^{q'_2})$.

Let $\lambda : T_{G_1}^{q'_1} \setminus T_{G_1}^{q_1} \rightarrow T_{G_2}^{q'_2} \setminus T_{G_2}^{q_2}$ witness the order equivalence claimed above, and replace elements of $(r'_1 \setminus r_1, \preceq_{r'_1})$ as follows: replace the elements of $r'_1 \setminus r_1$ derived from $T_{G_1}^{q'_1} \setminus T_{G_1}^{q_1} \subset T_{G_1}^{q'_1}$ with their appropriate image under λ . Formally, define a total order (s, \preceq_s) as follows:

$$s = \{\lambda \circ \pi_1(\hat{r}) : \hat{r} \in r'_1 \setminus r_1 \text{ and } \hat{r} \in P_1\} \cup \{\pi_2(\hat{r}) : \hat{r} \in r'_1 \setminus r_1 \text{ and } \hat{r} \in P_3\} \cup \{(\lambda \circ \pi_1(\hat{r}), \pi_2(\hat{r})) : \hat{r} \in r'_1 \setminus r_1 \text{ and } \hat{r} \in P_1 \times P_3\} \quad (6.3)$$

$$s_1 \preceq_s s_2 \text{ iff } \pi_1(s_1) \preceq_1 \pi_1(s_2) \text{ or } \pi_2(s_1) \preceq_1 \pi_2(s_2). \quad (6.4)$$

Because λ is a bijection and the total order in (6.4) is a subset of an interleaving total order, we can construct an interleaving between $T_{G_2}^{q'_2}$ and $T_{G_3}^{q'_1}$ by simply joining $s \cup r_2$ and suitably extending $\preceq_s \cup \preceq_{r_2}$. Of course we will call this interleaving q'_2 ,

and we claim that it has the aforementioned properties.

That $q'_2 \succeq_{Q_2} q_2$ is obvious from the construction. Likewise, it is obvious that $q'_1 \sim q'_2$. The only property that remains to be checked is that $(q_1, q'_1]$ is order equivalent to $(q_2, q'_2]$. But this follows from the construction and [Lemma 7](#). In particular, [Lemma 7](#) ensures that all elements of $(q_2, q'_2]$ can be mapped injectively to elements in the interleaving $q'_2 = (r'_2, \preceq_{r'_2})$, and likewise, all of the elements of $(q_1, q'_1]$ can be mapped injectively to elements in the interleaving $q'_1 = (r'_1, \preceq_{r'_1})$. Now, λ – through the mapping suggested in (6.3) – effectively serves as an order equivalence between the elements of the interleaving $r'_1 \setminus r_1$ and the elements of the interleaving $r'_2 \setminus r_2$, so it preserves the joint supremum property in [Lemma 7](#). That is take a $p \in (r'_1, \preceq_{r'_1})$ for which set $I = \{p' \in r_1 : p' \preceq_{r_1} p\}$ meets the condition that both $\sup \pi_1(I)$ and $\sup \pi_2(I)$ exist; then the analogous suprema will exist for the image of p under the order equivalence. Hence, the previous assertion based on [Lemma 7](#) assures that $(q_1, q'_1]$ and $(q_2, q'_2]$ are order equivalent. \square

6.2.3 Congruence and Strong Bisimulation

Recall the definition of strong bisimulation from [\[15\]](#) and given in [Definition 30](#). Strong bisimulation is also a congruence for CSP-parallel composition, as we shall now prove.

Theorem 19 (Strong Bisimulation is a congruence for CSP-parallel composition). *For $i = 1, 2, 3$, let $G_i = (P_i, \preceq_i, p_i, \mathcal{L}_i)$ be GSTs such that G_1 and G_2 are strongly bisimilar to each other. Then $C_1 := G_1|S|G_3$ is strongly bisimilar to $C_2 := G_2|S|G_3$.*

Proof. As in the proof of [Theorem 18](#), we establish a strong bisimulation relation between $C_1 = (Q_1, \preceq_{Q_1}, q_{01}, \mathcal{L}_{Q_1})$ and $C_2 = (Q_2, \preceq_{Q_2}, q_{02}, \mathcal{L}_{Q_2})$. As before, let $q_1 = (r_1, \preceq_{r_1})$ be a (non-root) node in C_1 , and let $q_2 = (r_2, \preceq_{r_2})$ be a (non-root) node in C_2 . Again, q_1 will denote an S-synchronized interleaving of trajectories $T_{G_1}^{q_1} \subseteq P_1$ and $T_{G_3}^{q_1} \subseteq P_3$ from the roots of G_1 and G_3 , respectively, and q_2 will denote an S-synchronized interleaving of trajectories $T_{G_2}^{q_2} \subseteq P_2$ and $T_{G_3}^{q_2} \subseteq P_3$ from the roots of G_2 and G_3 , respectively. We may assume from part 1 on [Definition 61](#) that at least one of $T_{G_1}^{q_1}$ and $T_{G_3}^{q_1}$ is non-empty, and likewise for $T_{G_2}^{q_2}$ and $T_{G_3}^{q_2}$. However, for this proof we will let $\sim_{1,2}$ be a *strong bisimulation* relation between G_1 and G_2 that includes (p_1, p_2) , the pair of root nodes from each.

The proof of [Theorem 18](#) offers a useful starting point: indeed, the proposed bisimulation relation in [\(6.2\)](#) is an adequate choice even for strong bisimulation, since the additional properties of the (now) strong bisimulation $\sim_{1,2}$ can be propagated through the proof without any further modifications. In particular, the order equivalence λ should associate nodes in G_1 with *bisimilar* nodes in G_2 as per the definition of strong bisimulation. Of course, this will translate into pairing nodes in $(q_1, q'_1]$ to bisimilar nodes in $(q_2, q'_2]$ because of the way that the proposed bisimulation relation (between C_1 and C_2) is defined. Thus, the same candidate bisimulation relation, derived as it now is from a *strong* bisimulation relation between G_1 and G_2 , will be a strong bisimulation relation for *right-closed trajectories* that extend one or the other of two nodes that are bisimilar.

However, we need to prove that each *right-open* trajectory from q_1 can be matched to a *right-open* trajectory from q_2 . To wit, let $q_1 \sim q_2$ (according to [\(6.2\)](#)),

and let $I_{q_1} = (q_1, \infty_1)$ be a right open trajectory in C_1 : we have to show that there exists a right-open trajectory (q_2, ∞_2) that is order-equivalent to (q_1, ∞_1) in a way that is consistent with the bisimulation relation \sim .

We begin by noting that $I_{q_1} = (q_1, \infty_1)$ is a set of interleavings of right-closed trajectories that are totally ordered by set containment (the prefix property in [Definition 61](#)). Thus, the constituent trajectories of these interleavings *themselves* form a right-open trajectory in *at least* one of the composed GSTs, G_1 and G_3 . However, the strong bisimulation relation $\sim_{1,2}$ considers right-open trajectories as well, so we can follow the program in [Theorem 18](#) to create a suitable set of right-closed interleavings that will form the desired trajectory (q_2, ∞_2) . Formally, let $R = \{(r_i, \preceq_i) : i \in I_{q_1}\}$ be the set of interleavings (with indexed total orders), and let each $(r_i, \preceq_{r_i}) \in R$ be an S-synchronized interleaving of trajectories $T_{G_1}^{r_i}$ and $T_{G_3}^{r_i}$. Then it is clear that for $i \preceq_{Q_1} j$, $T_{G_1}^{r_i} \subseteq T_{G_1}^{r_j}$ and $T_{G_3}^{r_i} \subseteq T_{G_3}^{r_j}$; furthermore, at least one of the sets $\cup\{T_{G_1}^{r_i} : i \in I_{q_1}\} \subset P_1$ and $\cup\{T_{G_3}^{r_i} : i \in I_{q_1}\} \subset P_3$ fails to contain its own supremum in P_1 or P_3 , respectively (if said supremum even exists). In other words, either $\cup\{T_{G_1}^{r_i} : i \in I_{q_1}\} \subset P_1$ is a *right-open trajectory* from $\sup T_{G_1}^{q_1}$ in G_1 **or** $\cup\{T_{G_3}^{r_i} : i \in I_{q_1}\} \subset P_3$ is a *right-open trajectory* from $\sup T_{G_3}^{q_1}$ in G_3 (both may be such right-open trajectories). Note that we could also have used the sets of suprema – e.g. $\cup\{\sup T_{G_1}^{r_i} : i \in I_{q_1}\} \subset P_1$ – without falling afoul of [Lemma 7](#), since we’re considering the endpoints of *both* constituent trajectories simultaneously: an endpoint from two interleaved trajectories is only “hidden” in the way suggested by [Lemma 7](#) when there fails to be an *interleaving* that has that *precise node* as a supremum – see also [Example 5](#). From this observation, we

continue the program in [Theorem 18](#): we generate a trajectory in C_2 by interleaving the trajectory $T_{G_3}^{q_1, \infty} = \cup\{\text{sup } T_{G_3}^{r_i} : i \in I_{q_1}\} \subset P_3$ with a trajectory that is **bisimilar** (according to $\sim_{1,2}$) to the trajectory $T_{G_1}^{q_1, \infty} = \cup\{\text{sup } T_{G_1}^{r_i} : i \in I_{q_1}\} \subset P_1$. Thus, let $T_{G_2}^{q_2, \infty} \subset P_2$ be a trajectory in G_2 for which there is a witness to *strong* bisimulation in the form of an order-preserving bijection $\lambda : T_{G_1}^{q_1, \infty} \rightarrow T_{G_2}^{q_2, \infty}$; such a λ may be applied to I_{q_1} as in [\(6.3\)](#) and [\(6.4\)](#), thereby constructing a trajectory of interleavings in C_2 . That the generalization of λ from [\(6.3\)](#) and [\(6.4\)](#) preserves bisimulation of the intermediary nodes is a direct consequence of both the definition [\(6.2\)](#) and the fact that all of the constituent trajectories are themselves prefixes of larger, right-open trajectories. □

6.2.4 Anomalies in CSP-Parallel Composition for GSTs

As it turns out, [Lemma 7](#) has rather profound implications for the congruence properties of CSP-parallel composition. As we have seen in [Section 6.2.2](#) and [Section 6.2.3](#), both weak and strong bisimulation are congruences for CSP-parallel composition of GSTs. However, in a literal sense, these congruence results only hold because not all nodes in an interleaving of trajectories are represented as nodes in the composed GST, as described in [Lemma 7](#) and exemplified in [Example 5](#). This effective hiding of nodes (and their labels) is obviously troublesome in the context of further composition, but the examples in this section demonstrate that it is a necessary price to pay for aforementioned compositionality results.

The culmination of this section will be an example that *ought* to contradict the

congruence for both weak and strong bisimulation *in the absence of the omission of nodes described in Lemma 7*. The essence of this example is that asymptotic choices can be made **differently** even in GSTs that strongly bisimilar; we introduce this concept in the following example, [Example 6](#) (contrast this with the example of asymptotic choice that illustrates the semantic difference between weak and strong bisimulation in [\[15\]](#)).

Example 6 (A bisimilar order equivalence between trajectories does not imply equivalent extensibility). Consider two GSTs – G_1 and G_2 – defined from the sets of nodes $P_1 = [0, 1) \cup (1, 2] \subset \mathbb{R}$ and $P_2 = [0, 1) \cup \bigcup_{x \in (0,1)} (x \times (x, 1) \cup x \times (1, 2])$, where 0 is the root of each tree, and the tree partial order is suggested by [Fig. 6.1](#). The labeling functions assign to any node $x \in (0, 1)$ (from either tree) and any $\langle y, x \rangle \in y \times (y, 1]$ the label given by the real number x , and to all other (non-root)

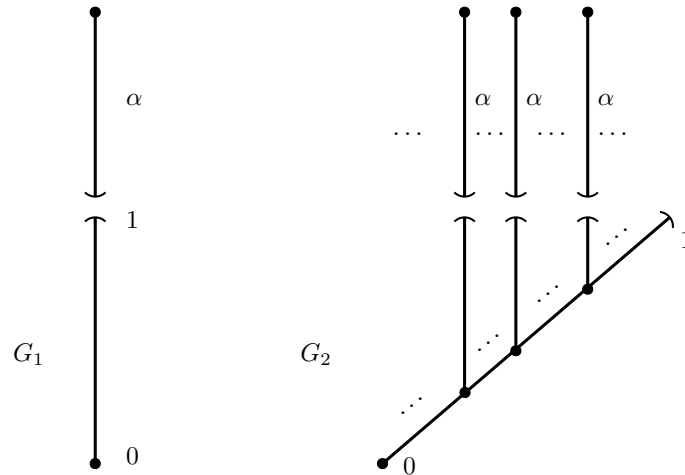


Figure 6.1: GSTs in [Example 6](#). G_1 and G_2 are strongly bisimilar according to [Definition 30](#), yet G_2 contains an un-extendable trajectory that nevertheless has a bisimilar order equivalence to an extendable trajectory from G_1 .

nodes the label α (again, see [Fig. 6.1](#)).

Proposition 11. *GSTs G_1 and G_2 given by [Example 6](#) and [Fig. 6.1](#) are strongly bisimilar according to [Definition 30](#).*

Proof. We claim that the following is a strong bisimulation relation between G_1 and G_2 :

$$\sim := \{ \langle x, x \rangle \subset P_1 \times P_2 : x \in [0, 1) \} \cup \cup_{x \in (0, 1)} \{ \langle y, \langle x, y \rangle \rangle : y \in (x, 1) \cup (1, 2) \}. \quad (6.5)$$

The only difficult case to check is for nodes $x \sim x$ for $x \in (0, 1)$ and trajectories like $(x, 2]$ in G_1 and like $(x, 1)$ in G_2 .

First, we consider the trajectory $(x, 2]$ in G_1 , and show that there is a bisimilar order equivalence to a trajectory in G_2 . Indeed there is, though: such a trajectory can be of the form $(\langle x, x \rangle, \langle x, 2 \rangle]$, or of the form $(x, y]; (\langle y, y \rangle, \langle y, 2 \rangle]$ for any $y > x$. In either case, matching identical real numbers forms a bisimilar order equivalence. On the other hand, this clearly covers all trajectories in G_2 emanating from x except $(x, 1)$, so it remains to check that there is a bisimilar order equivalence between that trajectory and one in G_1 . Again, though, the real numbers match $(x, 1)$ to the trajectory $(x, 1)$ in G_1 in a bisimilar order equivalence: the previous consideration of other trajectories ensures this. \square

The importance of [Proposition 11](#) is thus: G_1 and G_2 as defined in [Example 6](#) are strongly bisimilar, but clearly the trajectory $(0, 1)$ in G_1 is *always* extendable, whereas the trajectory $(0, 1)$ in G_2 is *never* extendable. This feature of G_1 and

G_2 can be used with CSP parallel composition to create *interleavings* that effectively “place” nodes *after* the trajectory $(0, 1)$ in G_2 . This *ought* to break strong bisimilarity because *no α 's are possible* after that particular trajectory in G_2 ; as suggested above, congruence will only be saved because certain nodes in the interleaving will not appear as nodes in the CSP-parallel composition. The subsequent example alters G_1 and G_2 to exploit this idea explicitly in CSP parallel composition.

Example 7 (“Counterexample” to strong bisimulation as a congruence for CSP parallel). Consider two GSTs – G'_1 and G'_2 – defined from the sets of nodes $P'_1 = [0, 1) \cup \{0, 1\} \times (1, 2] \subset \mathbb{R}$ and $P'_2 = [0, 1) \cup (1, 2] \cup \bigcup_{x \in (0, 1)} (x \times (x, 1) \cup x \times \{0, 1\} \times (1, 2])$, where 0 is the root of each tree, and the tree partial order is suggested by [Fig. 6.2](#). Note the addition of separate “branches” after $[0, 1)$ in G_1 and $(0, 1)$ and $(\langle x, x \rangle, \langle x, 1 \rangle)$ in G_2 . These extra branches carry the additional label γ : see [Fig. 6.2](#). The labeling functions for these GSTs are otherwise the same as described in [Example 6](#); again, see [Fig. 6.2](#).

Now consider a third GST, G_3 , defined over the set of nodes $P_3 = [0, 1] \subset \mathbb{R}$, where 0 is again the root node, and the tree partial order is given by \leq over the reals. The labeling function assigns to each node $x \in (0, 1)$ the label given by the real number x , and to the node 1 it assigns the label β . See [Fig. 6.2](#).

Proposition 12. GSTs G'_1 and G'_2 given by [Example 7](#) and [Fig. 6.2](#) are strongly bisimilar according to [Definition 30](#).

Proof. We propose a bisimulation relation between G'_1 and G'_2 that is much the same as the one in [Proposition 11](#), only with the addition that nodes labeled by γ

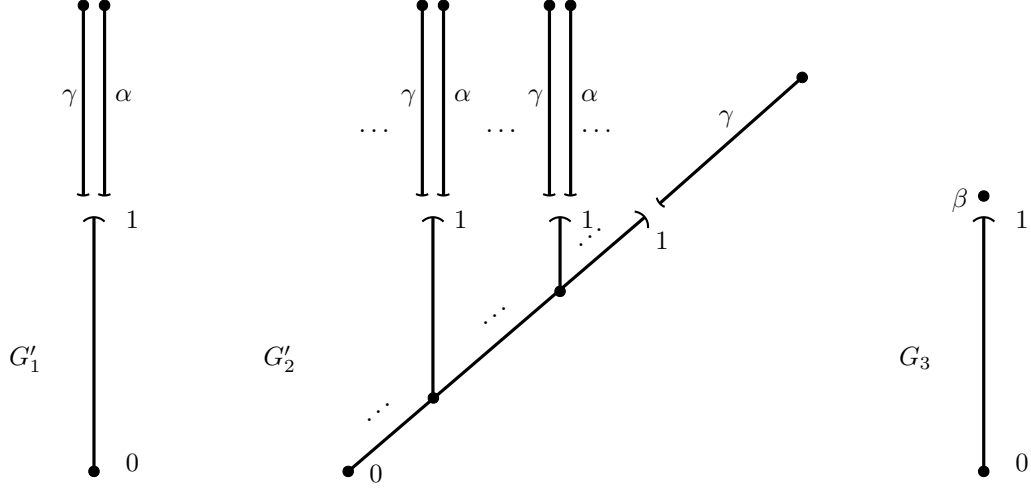


Figure 6.2: G'_1 and G'_2 are strongly bisimilar according to [Definition 30](#).

matched in the obvious way (by pairing real numbers).

The similarity between G_1 and G_2 and G'_1 and G'_2 also forms the basis for a proof of strong bisimulation: only the new trajectories that end in γ -labeled nodes need to be considered. As before, the properties of bisimulation are mostly obvious except for pairs of nodes of the form $x \sim x$ for $x \in (0, 1)$. Indeed the only interesting new case is the trajectory $(x, 2]$ in G'_2 , since all of the other trajectories that involve γ 's are matched covered by the γ branches that parallel α branches. However, in the case of $(x, 2]$ in G'_2 , there is no harm in matching the γ -labeled nodes of that trajectory to those in G'_1 since the nodes $y \in (x, 1) \subset P'_1$ each have the option to do α branches through the extensions $(\langle y, y \rangle, \langle y, 1, 2 \rangle]$. \square

In [Example 7](#), we're using synchronization to forcibly create an interleaving in $G'_2 | \mathbb{R} | G_3$ that places the β -labeled node from G_3 in between the $[0, 1)$ spine and the *solitary* γ branch in G'_2 , effectively completing said trajectory and providing an “anchor” from which to distinguish it from the others. If the node from that

interleaving were represented as a node in the composition, then it would serve as a counterexample to congruence for both weak and strong bisimulation. However, as there is no interleaving of *right-closed trajectories* that places the β node from G_3 in exactly such a position, the CSP-parallel composition effectively ignores this “choice” point.

6.3 A Process Algebra for Hybrid Systems: HyPA

This section shows how discrete GSTs can be constructed from HyPA terms so that bisimulation on GSTs (recall that weak and strong bisimulation coincide for discrete trees) corresponds exactly with a congruence for HyPA terms in [4]. HyPA is summarized in [Section 2.4](#).

In [4], the operational semantics of HyPA are given as a hybrid transition system; the states in the transition were HyPA-term / variable-valuation pairs; this construction is summarized in [Section 2.4](#). Bisimulation may then be defined as usual for such a transition system. The authors of [4] then consider different definitions of bisimulation for terms alone. One obvious candidate defines terms \mathbf{p} and \mathbf{q} to be bisimilar iff for all variable valuations ν , $\langle \mathbf{p}, \nu \rangle$ and $\langle \mathbf{q}, \nu \rangle$ are bisimilar as states in the HyPA hybrid transition system. Unfortunately this relation is not a congruence for HyPA terms; the problem resides in the fact that parallel processes within terms can interfere with the variable valuations produced by another parallel process. To fix this problem, the authors introduce another relation, *robust bisimulation*, show it to be a congruence for HyPA, then establish that it is the same as

another relation, *stateless bisimulation*, given in the same paper.

In the rest of this section we show how to construct GSTs from HyPA terms in such a way that two HyPA terms are statelessly bisimilar iff the corresponding GSTs are bisimilar. We begin by reviewing the definition of stateless bisimulation. $\mathcal{T}(\mathcal{V}_p)$ is the set of HyPA terms that use only the recursive process variables \mathcal{V}_p , Val is the set of valuations for the (continuous) model variables, the transition $\xrightarrow{\ell}$ represents either a discrete action or a continuous flow (depending on ℓ) and \checkmark is a set of “terminating” states.

This discussion suggests a way to obtain congruence with respect to HyPA: one must exhibit a bisimulation relation that includes all of the subprocess/ valuation pairs that are hidden by any initial choice of valuation. This is exactly the idea behind stateless bisimulation, which is indeed a congruence for HyPA [4]¹. See [Definition 20](#), which recalls the following notation from [4] (and [Section 2.4](#)): $\mathcal{T}(\mathcal{V}_p)$ is the set of HyPA terms that use only the recursive process variables \mathcal{V}_p , Val is the set of valuations for the (continuous) model variables, the transition $\xrightarrow{\ell}$ represents either a discrete action or a continuous flow (depending on ℓ) and \checkmark is a set of “terminating” states. Thus, the goal should be to construct GSTs from HyPA processes in such a way that bisimulation between the GSTs corresponds exactly to stateless bisimulation ([Definition 20](#)) between the original processes.

For simplicity we ignore \checkmark in what follows. The construction of GST $G_{\mathbf{p}}$ from

¹It is proved in [4] that stateless bisimulation is equivalent to a more complicated notion of equivalence called robust bisimulation; thus, we may treat only stateless bisimulation without any loss of generality.

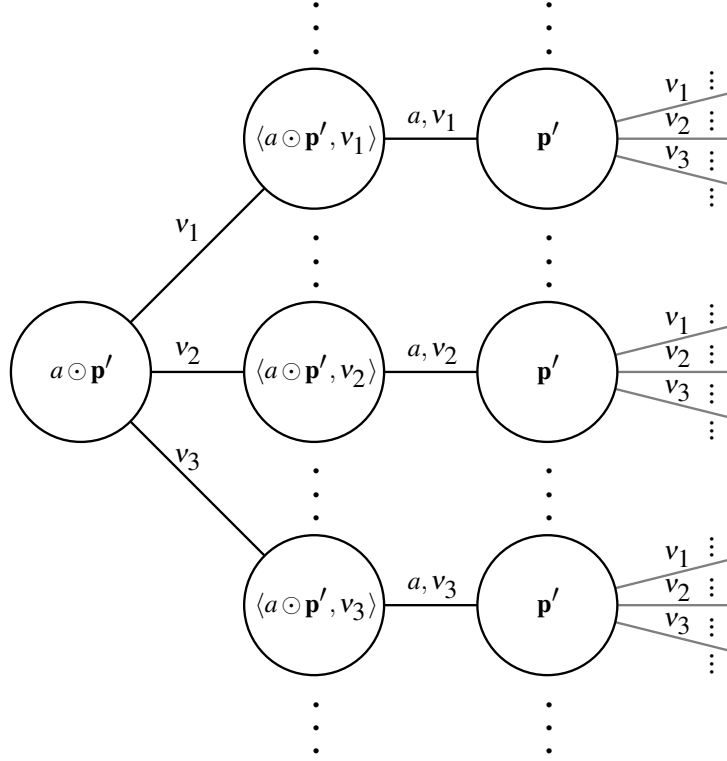


Figure 6.3: Constructing GSTs from HyPA Terms

HyPA term \mathbf{p} is exemplified in Fig. 6.3. First, let the root of the $G_{\mathbf{p}}$ be identified with \mathbf{p} . Then for each valuation ν of the model variables, create one successor node of \mathbf{p} that is identified with $\langle \mathbf{p}, \nu \rangle$ and label these nodes as ν . Since each $\langle \mathbf{p}, \nu \rangle$ is a hybrid transition system state, the node has transitions of form $\langle \mathbf{p}, \nu \rangle \xrightarrow{\ell} \langle \mathbf{p}', \nu' \rangle$ for some ℓ ; for each such $\langle \mathbf{p}', \nu' \rangle$, make a node for \mathbf{p}' labeled by ℓ . Now repeat this procedure (coinductively) from \mathbf{p}' to obtain discrete GST $G_{\mathbf{p}}$. We now have the following (recall that weak and strong bisimilarity coincide for discrete GSTs).

Theorem 20. *Let \mathbf{p} and \mathbf{q} be HyPA terms. Then \mathbf{p} and \mathbf{q} are statelessly bisimilar iff $G_{\mathbf{p}}$ and $G_{\mathbf{q}}$ are bisimilar.*

Proof. Let \mathbf{p} and \mathbf{q} be two HyPA processes, and construct two GSTs $G_1 = \langle P, p_0, \lesssim_P$

, \mathcal{L}_P) and $G_2 = \langle Q, q_0, \lesssim_Q, \mathcal{L}_Q \rangle$ from these processes according to the preceding.

We show the reverse direction first: if G_1 and G_2 are related by a bisimulation relation \mathcal{R} that includes $\langle p_0, q_0 \rangle$, then there is a stateless bisimulation relation R that includes $\langle \mathbf{p}, \mathbf{q} \rangle$. From \mathcal{R} , we suggest that

$$R = \left\{ \langle \mathbf{p}', \mathbf{q}' \rangle \mid \exists p \mathcal{R} q \text{ s.t. } \left(p = \epsilon \bigvee |p| \geq 1 \wedge \text{last}(p) \in \rightarrow \wedge \text{proc}_2(p) = \mathbf{p}' \right) \wedge \right. \\ \left. \left(q = \epsilon \bigvee |q| \geq 1 \wedge \text{last}(q) \in \rightarrow \wedge \text{proc}_2(q) = \mathbf{q}' \right) \right\}$$

is such a stateless bisimulation relation. Clearly, R includes $\langle \mathbf{p}, \mathbf{q} \rangle$ by construction, so we need only establish that R is a stateless bisimulation relation. To prove the first simulation condition in the definition of stateless bisimulation, arbitrarily choose \mathbf{p}' and \mathbf{q}' such that $\mathbf{p}' R \mathbf{q}'$. Now choose an arbitrary valuation ν' , and let $t_{\delta p} = \langle \mathbf{p}', \nu' \rangle \xrightarrow{\ell} \langle \mathbf{p}'', \nu'' \rangle$ be any transition from state $\langle \mathbf{p}', \nu' \rangle$. It is easy to see that $p'.\nu'$ and $p'.\nu'.t_{\delta p}$ are nodes in G_1 ; it is also clear that $p' \lesssim_P p'.\nu' \lesssim_P p'.\nu'.t_{\delta p}$. Since \mathcal{R} is a simulation relation, there exists a node $q'_{\nu'} \lesssim_Q q'$ such that $p'.\nu' \mathcal{R} q'_{\nu'}$ and $\mathcal{L}_P(p'.\nu') = \mathcal{L}_Q(q'_{\nu'}) = \nu'$. As $p'.\nu'$ is obviously an immediate successor of p' , $q'_{\nu'}$ must be an immediate successor of q' ; since $\mathcal{L}_Q(q'_{\nu'}) = \nu'$, it must be the case that $q'_{\nu'} = q'.\nu'$. Now we again use the simulation property of \mathcal{R} to assert the existence of a $q'' \lesssim_Q q'.\nu'$ such that $p'.\nu'.t_{\delta p} \mathcal{R} q''$ and $\mathcal{L}_P(p'.\nu'.t_{\delta p}) = \mathcal{L}_Q(q'') = \ell$. Reasoning as above, q'' must be an immediate successor to $q'.\nu'$, and so $\text{last}(q'')$ must be a transition with $\mathcal{L}_Q(q'') = \text{label}(\text{last}(q'')) = \ell$. Since q'' is a node in the tree, it must be true that $\text{last}(q'') = \langle \mathbf{q}', \nu' \rangle \xrightarrow{\ell} \langle \mathbf{q}'', \nu''_q \rangle$ for some process \mathbf{q}'' and valuation ν''_q . This establishes that $\langle \mathbf{q}', \nu' \rangle$ can process the label ℓ ; since $p'.\nu'.t_{\delta p} \mathcal{R} q''$, we conclude that $\mathbf{p}'' R \mathbf{q}''$ by definition of the set R . The other simulation condition follows by

symmetric arguments because \mathcal{R} is a bisimulation relation. Hence, we conclude that R is a stateless bisimulation relation.

In the other direction, suppose that R is a stateless bisimulation relation such that $\mathbf{p}R\mathbf{q}$. We claim that there is a bisimulation relation \mathcal{R} between G_1 and G_2 .

First we define the set

$$\mathcal{R}' = \left\{ \langle p, q \rangle \mid \exists \mathbf{p}'R\mathbf{q}' \text{ s.t. } \left(\mathbf{p}' = \mathbf{p} \wedge p = \epsilon \vee \text{last}(p) \in \rightarrow \wedge \mathbf{p}' = \text{proc}_2(\text{last}(p)) \right) \right. \\ \left. \wedge \left(\mathbf{q}' = \mathbf{q} \wedge q = \epsilon \vee \text{last}(q) \in \rightarrow \wedge \mathbf{q}' = \text{proc}_2(\text{last}(q)) \right) \right\}.$$

Then we claim that

$$\mathcal{R} = \mathcal{R}' \cup \{ \langle p.\nu, q.\nu \rangle \mid \langle p, q \rangle \in \mathcal{R}' \text{ and } \nu \in \text{Val} \}$$

is a bisimulation relation between G_1 and G_2 . Since $\mathbf{p}R\mathbf{q}$, it is the case that $p_0\mathcal{R}q_0$.

We first show simulation in one direction. Because both trees are discrete, it suffices to show simulation only for the immediate successor of a node. In particular, arbitrarily choose p' and q' such that $p'\mathcal{R}q'$. First, suppose that either $p' = \epsilon$ or $\text{last}(p') \in \rightarrow$, i.e. $\langle p', q' \rangle \in \mathcal{R}'$. In this case, the only immediate successors to p' are of the form $p'.\nu'$, so arbitrarily choose $\nu' \in \text{Val}$. Clearly, $p'.\nu'\mathcal{R}q'.\nu'$ by definition, and $\mathcal{L}_P(p'.\nu') = \mathcal{L}_Q(q'.\nu') = \nu'$. This obviously establishes simulation for any immediate successor node of p' . Now consider $p''\mathcal{R}q''$ such that $p'' = p'.\nu'$, i.e. $\langle p'', q'' \rangle \in \mathcal{R} \setminus \mathcal{R}'$. Only nodes that end in the valuation ν' can possibly be related to p'' by \mathcal{R} , so we might as well write $q'' = q'.\nu'$. The tree property and the construction of \mathcal{R} from \mathcal{R}' ensure that $p'\mathcal{R}'q'$. Without much loss of generality, we may suppose that both p' and q' are non-empty: in that case, there exist stateless bisimilar processes \mathbf{p}'

and \mathbf{q}' such that $proc_2(last(p')) = \mathbf{p}'$ and $proc_2(last(q')) = \mathbf{q}'$. However, stateless bisimilarity implies that for every transition $t_{\delta_p} = \langle \mathbf{p}', \nu' \rangle \xrightarrow{\ell} \langle \mathbf{p}'', \nu'' \rangle$ – i.e. every immediate successor to $p'.\nu'$ – there is a transition $t_{\delta_q} = \langle \mathbf{q}', \nu' \rangle \xrightarrow{\ell} \langle \mathbf{q}'', \nu''_q \rangle$ such that $\mathbf{p}'' R \mathbf{q}''$. This implies that $p''.t_{\delta_p} \mathcal{R} q''.t_{\delta_q}$ by definition of the set \mathcal{R}' . Since $\mathcal{L}_P(p''.t_{\delta_p}) = \mathcal{L}_Q(q''.t_{\delta_q}) = \ell$, we have established simulation in one direction. Simulation in the other direction follows by symmetric arguments because R is a stateless *bisimulation* relation. \square

There are yet other ways to represent HyPA processes as GSTs. For example, encoding HYPA processes as a behavioral systems suggests that if HyPA processes are regarded in terms of execution trajectories (i.e. functions of time), yet a different GST construction can be obtained. It should be noted that such a construction necessarily has a great deal more granularity, as the resulting GSTs would be non-discrete. Consequently, our previous results about strong bisimulation would likely have significant ramifications for such a GST construction.

Chapter 7: Conclusions and Future Research

7.1 Conclusions

This dissertation introduces a new CPS modeling framework that mitigates some of the significant limitations associated with prior CPS models. In particular, we propose GSTs as a modeling framework that is general enough to encompass a wide swath of CPS systems yet inherits the rich and expressive compositionality of the process algebraic models (STs) from which it is derived. These structural advantages make GSTs a *scalable* tool for the modeling and design of safe, reliable CPSs: that is by using the inherent compositionality of GSTs to design and verify small, manageable subsystems, which can subsequently be combined into large, complex systems via flexible, principled compositionality operators.

In [Chapter 3](#), we formally defined GSTs by means of generalizing the notion of a tree; this construction analogizes the structure of Milner’s STs, and it is ultimately the source GSTs’ flexibility. Also in this chapter, we considered two of the prevailing notions of bisimulation for non-discrete systems, and we proved for the first time that these notions are semantically distinct. This has wide implications for the study of CPSs, since specifying system equivalence up to weak bisimulation will lead to different behavior than specifying system equivalence up to strong bisimulation.

In [Chapter 4](#), we considered the problem of “unrolling” state-based behavioral systems into GSTs, and we demonstrated that such unrollings are not only possible but also *preserve a notion of bisimulation*. The success of this endeavor is essential to the value of GSTs as a modeling framework. To be useful, translating conventional CPS models into GSTs must preserve some meaningful notion of semantics: otherwise, GSTs could not be used to reason about conventional system models at all.

In [Chapter 5](#), we again followed in Milner’s (and Hennessy’s) footsteps by defining a generalized HML-like modal logic for GSTs. We further established that GHML has a relationship to (weak) bisimulation that mirrors the one between HML and bisimulation over STs; this idea extends to the definition of *maximal VHHM* classes of GSTs, which are maximal classes of GSTs for which a GHML modal formula can provide a diagnostic *witness* to non-bisimilarity between the constituent GSTs. Finally, we studied the circumstances under which Linear, Time-Invariant Systems are contained in a maximal VHHM class, thus specifying for the first time the extent to which GHML formulas are expressive enough to capture weak bisimulation between such systems.

In [Chapter 6](#), we defined a number of different composition operators over GSTs, and for each of these operators, we demonstrated that bisimulation is a congruence. We defined the following operators: first, a purely synchronous operator (of the sort common for behavioral systems); second, a novel *asynchronous* composition operator applicable to arbitrary GSTs (and hence continuous and hybrid systems); and finally, operators translated from the HyPA process terms (which we

further showed are inherently discrete).

7.2 Future Research

There are many directions for future research suggested by the work in this dissertation. The most natural and immediate ones are as follows.

- Additional CPS models may be “unrolled” into GSTs in a bisimulation-preserving way; this will broaden the reach of GSTs as a flexible, unifying modeling framework.
- There are an abundance of composition operators in process algebra that have yet to be adapted as composition operators over GSTs; of course, the congruence properties of these adaptations need to be specified, and there may be unique and surprising results in this regard (especially in comparing weak and strong bisimulation).
- GHML should be studied with respect to *strong* bisimulation; this will lead to separate and interesting results about the expressivity of a simple modal logic relative to the unique semantics of strong bisimulation, especially in the context of continuous/hybrid systems.
- The characterization of GHML for LTI systems (with respect to weak bisimulation) depends on some moderately restrictive assumptions about both the system itself and the allowed inputs; another direction of future research involves weakening either or both of these restrictions.

- There is an open problem of characterizing more general continuous-time systems in terms of GHML and weak/strong bisimulation (nonlinear and time-varying systems, for example); in this dissertation, the results on LTI systems depend heavily on unique features of LTI systems that do not have immediate analogs in more general continuous-time systems.
- There is an interesting direction of research that involves specifying the properties of composition operators with respect to maximal VHHM classes of GSTs; this problem was considered by Hollenberg [27] for discrete systems, but given the (potentially) unique semantics of composition operators over GSTs, there are likely to be interesting and surprising properties when this question is posed in the GST context.

Bibliography

- [1] Jerry Hirsch. Toyota prius recall: Automaker will fix nearly 700,000 hybrids. *Los Angeles Times*, November 2012.
- [2] Robert N. Charette. This car runs on code. *IEEE Spectrum*, February 2009.
- [3] T A Henzinger. The theory of hybrid automata. In *Logic in Computer Science, 1996. LICS '96. Proceedings., Eleventh Annual IEEE Symposium On*, pages 278–292, 1996.
- [4] P J L Cuijpers and M A Reniers. Hybrid process algebra. *The Journal of Logic and Algebraic Programming*, 62(2):191–245, 2005.
- [5] Paulo Tabuada. *Verification and Control of Hybrid Systems: A symbolic approach*. Springer, 2009.
- [6] S. Strubbe and A. van der Schaft. Algorithmic bisimulation for Communicating Piecewise Deterministic Markov Processes. In *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05*, pages 6109–6114, 2005.
- [7] Sebastien Bornot and Joseph Sifakis. On the composition of hybrid systems. *Hybrid Systems: Computation and Control*, 1998.
- [8] Paulo Tabuada, George J. Pappas, and Pedro Lima. Compositional Abstractions of Hybrid Control Systems. *Discrete Event Dynamic Systems*, 14(2):203–238, 2004.
- [9] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. Languages and Tools for Hybrid Systems Design. *Found. Trends Electron. Des. Autom.*, 1(1/2):1–193, 2006.
- [10] A. J. van der Schaft and J. M. Schumacher. Compositionality issues in discrete, continuous, and hybrid systems. *International Journal of Robust and Nonlinear Control*, 11(5):417–434, 2001.

- [11] Aaron D. Ames, Alberto Sangiovanni-Vincentelli, and Shankar Sastry. Homogeneous Semantics Preserving Deployments of Heterogeneous Networks of Embedded Systems. In Panos J. Antsaklis and Paulo Tabuada, editors, *Networked Embedded Sensing and Control*, number 331 in Lecture Notes in Control and Information Science, pages 127–154. Springer Berlin Heidelberg, 2006.
- [12] Esfandiar Haghverdi, Paulo Tabuada, and George J Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Comput Science*, 342:229–261, 2005.
- [13] Jan C. Willems. On interconnections, control, and feedback. *IEEE Transactions on Automatic Control*, 42(3):326–339, 1997.
- [14] Jan Willems. The Behavioral Approach to Open and Interconnected Systems. *IEEE Control Systems Magazine*, 27(6):46–99, 2007.
- [15] James Ferlez, Rance Cleaveland, and Steve Marcus. Generalized Synchronization Trees. In *Foundations of Software Science and Computation Structures (FoSSaCS)*, pages 304–319. Springer Berlin Heidelberg, 2014.
- [16] Robin Milner. *A Calculus of Communicating Systems*. Number 92 in Lecture Notes in Computer Science. Springer-Verlag, 1980.
- [17] J. Ferlez, R. Cleaveland, and S. I. Marcus. Bisimulation in Behavioral Dynamical Systems and Generalized Synchronization Trees. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 751–758, 2018.
- [18] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 1985.
- [19] James Ferlez, Rance Cleaveland, and Steve Marcus. Bisimulation and Hennessy-Milner Logic for Generalized Synchronization Trees. *Electronic Proceedings in Theoretical Computer Science*, 255:35–50, 2017.
- [20] David Park. Concurrency and Automata on Infinite Sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183. Springer-Verlag, 1981.
- [21] Samson Abramsky. A domain equation for bisimulation. *Information and Computation*, 92(2):161 – 218, 1991.
- [22] P. Aczel. *Non-Well-Founded Sets*, volume 14 of *CSLI Lecture Notes*. Center for the Study of Language and Information, 1988.
- [23] William C. Rounds. On the relationships between Scott domains, synchronization trees, and metric spaces. *Information and Control*, 66(1–2):6 – 28, 1985.

- [24] Glynn Winskel. Synchronization Trees. *Theoretical Computer Science*, 34:33–82, 1984.
- [25] A. A. Julius and A J van der Schaft. Bisimulation as congruence in the behavioral setting. In *Proceedings of the 44th IEEE Conference on Decision and Control and 2005 European Control Conference*, pages 814–819, 2005.
- [26] Robert Goldblatt. *Logics of Time and Computation*. CSLI Lecture Notes, second edition, 1992.
- [27] Marco Hollenberg. Hennessy-Milner Classes and Process Algebra. In Alban Ponse, Maarten de Rijke, and Yde Venema, editors, *Modal Logic and Process Algebra: A Bisimulation Perspective*, Center for the Study of Language and Information Publication Lecture Notes, pages 187–216. Cambridge University Press, 1995.
- [28] Thomas Jech. *Set Theory*. Springer Monographs in Mathematics. Springer-Verlag, third millenium edition, 2003.
- [29] P.J.L. Cuijpers. Prefix Orders as a General Model of Dynamics. *\ldots on Developments in Computational Models \ldots*, 2013.
- [30] Luca Aceto, Bard Bloom, and Frits Vaandrager. Turning SOS rules into equations. *Information and Computation*, 111(1):1–52, 1994.
- [31] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can’t be traced. *J. ACM*, 42(1):232–268, 1995.
- [32] Valentin Goranko and Martin Otto. Model theory of modal logic. In Johan Van Benthem and Frank Wolter Patrick Blackburn, editor, *Studies in Logic and Practical Reasoning*, volume 3 of *Handbook of Modal Logic*, pages 249–329. Elsevier, 2007.
- [33] J M Davoren and Paulo Tabuada. On Simulations and Bisimulations of General Flow Systems. In *Hybrid Systems: Computation and Control*, pages 145–158. Springer Berlin Heidelberg, 2007.
- [34] Steven G. Krantz and Harold R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser Advanced Texts Basler Lehrbücher. Birkhäuser Basel, 2 edition, 2002.
- [35] Halsey Royden and Patrick Fitzpatrick. *Real Analysis*. Pearson, 4th edition, 2010.
- [36] A. Nonnengart. Modal frame characterization by way of auxiliary modalities. *Logic Journal of IGPL*, 6(6):875–899, 1998.
- [37] Robert. Goldblatt. *Mathematics of Modality*. CSLI lecture notes ; no. 43; CSLI lecture notes ; no. 43. CSLI Publications, 1993.