Generalized Synchronization Trees

James Ferlez[1,2], Rance Cleaveland[1,3], and Steve Marcus[1,2]

[1] The Institute for Systems Research, University of Maryland
[2] Department of Electrical and Computer Engineering, University of Maryland
[3] Department of Computer Science, University of Maryland

Abstract. This paper develops a generalized theory of synchronization trees. In their original formulation, synchronization trees modeled the behavior of nondeterministic discrete-time reactive systems and served as the foundational theory upon which process algebra was based. In this work, a more general notion of tree is proposed that is intended to support the modeling of systems with a variety of notions of time, including continuous and hybrid versions. (Bi)simulation is also studied, and it is shown that two notions that coincide in the discrete setting yield different relations in the generalized framework. A CSP-like parallel composition operator for generalized trees is defined as a means of demonstrating the support for compositionality the new framework affords.

1 Introduction

Research into process algebra has been highly influential in the mathematical study of system modeling [4]. Such algebras include a collection of operators for assembling more complex systems from smaller ones, as well as notions of behavioral equivalence and refinement for determining when two systems are indistinguishable behaviorally and when one system is an elaboration of another. This principled approach to compositional modeling has inspired the development of a wealth of mechanisms for combining systems in practically interesting yet mathematically well-founded ways for event-driven systems.

Synchronization trees, as proposed originally by Milner [15], played a pivotal role in the development of process algebra. In any algebraic theory, carrier sets must be specified; operators in the algebra are then interpreted as constructions over the elements from these sets. Synchronization trees play this role in traditional process algebras. Intuitively, a synchronization tree encodes the behavior of a system: nodes in the tree correspond to states, with edges, which are labeled by events, representing execution steps. Composition operators may then be interpreted as constructions on these trees, with the result of the construction representing the behavior of the composite system. These constructions in turn may be specified co-inductively via inference rules [5]. The simplicity and flexibility of synchronization trees led several researchers to formalize Milner's original notion using different mathematical machinery [1,3,17] and indeed helped inspire seminal work on co-induction in computing [12,18].

Synchronization trees are intended to model discrete systems that evolve by engaging in atomic events and changing state. For systems with non-discrete behavior, a similarly general yet simple model for defining composition has arguably yet to emerge. Some researchers use transition systems to model such behavior [6,9], while others adopt category-theoretic [10,11] and trajectory-based models [14,19]. However, neither case has yielded the rich results for composition operators that can be found in discrete process algebra. By contrast, others have recently recognized that generalizing the notion of a tree is a profitable approach [7,8]. However, in [8], generalized trees appear only as a consequence of other system models, whereas in [7], the notions of bisimulation and CSP-parallel composition are not generalized to non-discrete behavior.

The purpose of this paper is to propose a new tree-based model of system behavior, which we call *generalized synchronization trees*, that is intended to play the same role in a generalized-time setting that synchronization trees play for discrete time. Our goal is to provide the foundation for a general, flexible theory of composition for systems that include components with a variety of different notions of time, including continuous, discrete, and their hybrids. Generalized synchronization trees also represent system behavior and subsume traditional synchronization trees, but include a flexible mechanism for modeling non-discrete models of time as well. In this paper, we define these trees and study notions of equivalence (bisimulation) and refinement (simulation) in a logical (i.e. non-real-time) setting. Our results show in particular that definitions of these behavioral relations that coincide in the discrete setting differ in the generalized setting. We also show how our trees subsume some existing models of hybrid behavior, and we initiate the study of composition operations in this theory by showing how CSP parallel composition can be extended to this framework.

The remainder of the paper is structured as follows. The next section presents mathematical preliminaries, while the section following gives the definition of generalized synchronization trees and some examples. Section 4 then studies different notions of bisimulation for this model, and the section after shows how our trees may be used to model systems in an existing hybrid process algebra. The next section considers parallel composition in the setting of our new trees, and the final section concludes with directions for future work.

2 Preliminaries

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

Definition 1 (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 2 (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 3 (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 [13].

Definition 4 (Tree [13]). *A* tree *is partial order $\langle P, \preceq \rangle$ such that for each $p \in P$, the set $\{p' \in P \mid p' \precsim p\}$ is totally ordered by $\preceq$. If there is also a $p_0 \in P$ such that $p_0 \precsim p$ 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 [7], these structures are referred to as prefix orders. The distinguishing feature of a tree implies that $\precsim$ 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 5 (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 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 Generalized Synchronization Trees

This section defines, and gives examples of, generalized synchronization trees.

3.1 Traditional Synchronization Trees

Milner introduced the notion of synchronization tree in [15]. The following quotes the definition given on p. 16 of that reference.

> "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]

(Milner also indicates that such trees may be of infinite depth. Note that because of its reference to "arcs", Milner's definition implies that synchronization trees must be discrete in the sense of Definition 5.) Intuitively, the set L of labels contains externally visible actions that systems may engage in; τ denotes a designated internal action. A tree then represents the full behavior of a system; the root represents the start state, while edges represent discrete computation steps and branching represents nondeterminism. Milner shows how the basic composition operators, including choice and parallel composition, of his Calculus of Communicating Systems (CCS) may be interpreted as constructions on these trees. Throughout his presentation Milner consciously follows a traditionally algebraic approach, making CCS one of the earliest process algebras.

Process algebras are often given a semantics in terms of *labeled transition systems*; a ST may be seen as the "unrolling" of such a system.

Milner also defined a notion of *strong equivalence* (now called *bisimulation equivalence*) on systems in order to equate synchronization trees that, while not isomorphic, nevertheless represent the same behavior. The definitions below are adapted from [16]; 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 6 (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 $\langle T_1, T_2\rangle \in R$.*

[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 paper.

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.

Milner's definition of synchronization tree may be seen as semi-formal in that trees are not formally defined. Other authors [1,3,17,20] subsequently developed the underlying mathematics fully, and in the process helped justify, as coinductive constructions, the composition operations given by Milner on infinite trees.

3.2 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 [2,5] 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 7 (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 4; and*
2. *$\mathcal{L} \in P \backslash \{p_0\} \to L$ 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 4. 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 example contain different mixtures of discrete / continuous time and discrete / continuous nondeterminism (called "choice").

Example 1 (Labeled Transition Systems as GSTs). Let $T = \langle X, L, \to, x_0 \rangle$ be a labeled transition system with state set X, label set L, transition relation $\to \subseteq X \times L \times X$, and initial state x_0. Then the behavior of T starting from

x_0 may be encoded as a discrete GST. Define an *execution* e of T to be a sequence of transitions (formally, an element of $\rightarrow^*$) such that if $e = \langle x, \ell, x' \rangle \cdot e'$ then $x = x_0$ (i.e. the first transition is always from the start state), and if $e = e_1 \cdot \langle x_1, \ell_1, x_1' \rangle \cdot \langle x_2, \ell_2, x_2' \rangle \cdot e_2$ then $x_1' = x_2$ (i.e. the next transition always starts from the target state of the last transition). Let E_T be the set of all executions of T; note that ϵ, the empty sequence of transitions, is in E_T, and that $\precsim_T$, the prefix ordering on $\rightarrow^*$, is a partial order on E_T such that $\epsilon \precsim_T e$ for all $e \in E_T$. Finally, if $e = e' \cdot \langle x, \ell, x' \rangle$, define $\mathcal{L}_T(e)$ to be ℓ (i.e. the label of the last transition in e). It is easy to see that $G_T = \langle E_T, \precsim_T, \epsilon, \mathcal{L}_T \rangle$ is a GST, and that G_T is discrete in the sense of Definition 5.

The previous construction is the classical "unrolling" method for generating trees from LTSs, and is generally associated with discrete-time modeling. Perhaps surprisingly, however, it is also applicable to formalisms that model continuous behavior via transition systems. For example, Hybrid Process Algebra (HyPA) [6] 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. The behavior of HyPA terms depends on the values of the continuous model variables; transitions are enabled only for certain valuations of thse variables, and transitions can also alter the model variables when they execute. Thus, the operational semantics, which is specified in the traditional SOS style, defines transitions for HyPA-term / variable-valuation pairs. Two types of transitions are in fact defined; zero-duration, discrete-action "jumps", and finite-duration continuous flows. The results is 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 [6]. As labeled transitions systems, hybrid transition systems can be represented as GSTs using the construction above; these GSTs are also discrete, interestingly, even though the phenomena being modeled in HyPA are not.

Difference equations with inputs also represent models that semantically give rise to transition systems. Such models often arise in the description of control systems when the quantities of interest – states and inputs, for example – are sampled in time. Such difference equations typically take the form

$$x_{k+1} = f(x_k, u_k),$$

where x_k represents state at the k^{th} sampling interval, u_k represents an input arriving between time k and $k+1$, and f is a function computing the new state based on the existing state and this input. These systems can be represented as labeled transition systems, with states given by the x and transitions labeled by u defined by f, so the above construction yields discrete GSTs in this case, too.

Example 2 (Differential Equations with Inputs as GSTs). Continuous-time, continuous - choice systems have traditionally been the standard problem considered by control theorists; these systems usually take the form of an ordinary differential equation (ODE) with inputs. A simple example of this class of systems is one derived from Newton's laws of motion. For example, consider an object with mass m that is both confined to travel in a straight line and affected by a time-varying external force u (from a motor, say). If we let x_1 represent the position of the object and we let x_2 represent its velocity, then Newton's laws can be used to derive the following state equations for the object:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u \tag{1}$$

where all the variables are functions of time and $\dot{x}$ represents the time-derivative of x. Recently, Willems *et al.* [19] have suggested that continuous-time systems be treated as a collection of time trajectories (or *behaviors*) instead of a set of state equations. If we suppose that time starts from 0, then the previous system is completely described by the set of all pairs of functions

$$\mathcal{B} \triangleq \Big\{ \langle u, x\rangle \in (\mathcal{R}^{\geq 0} \to \mathcal{R}) \times (\mathcal{R}^{\geq 0} \to \mathcal{R}^2) : u \text{ is locally integrable } \textit{and}$$
$$\exists x_0 \in \mathcal{R}^2 \text{ s.t. } x(t) = \exp(At)x_0 + \int_0^t \exp(A(t-\tau))Bu(\tau)d\tau \quad \forall t \geq 0 \Big\} \tag{2}$$

where $A = \left[\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} 0 \\ 1/m \end{smallmatrix}\right]$ and the function exp should be interpreted as the matrix exponential. The function u is assumed to be locally integrable so that the subsequent integral is well defined for all t.

This behavioral treatment of continuous-time systems facilitates the construction of GSTs using a generalized notion of "prefix". To help with this, we define a notion of truncation for functions defined on $\mathcal{R}^{\geq 0}$: given a function $f \in \mathcal{B}$, we define $f|_t$ to be the restriction of the function f to the set $[0, t]$. Now we can define a GST from $\mathcal{B}$ as follows.

- Let $P = \{\langle t, f\rangle : t \in \mathcal{R}^{\geq 0}$ *and* $f = x|_t$ for some $x \in \mathcal{B}\} \cup \{p_0\}$ where p_0 is a distinguished element not in the first set.
- The partial order $\preceq$ is defined in the following way: let $p_0 \preceq p$ for all $p \in P$ and for $p_1 \triangleq \langle t_1, f_1\rangle$, $p_2 \triangleq \langle t_2, f_2\rangle$ let $p_1 \preceq p_2$ if and only if $t_1 \leq t_2$ and $f_1 = f_2|_{t_1}$.
- The labeling function $\mathcal{L} : P \backslash \{p_0\} \to \mathcal{R}$ is defined so that $\mathcal{L}(\langle t, f\rangle) = \pi_1 f(t)$.

We close this section by remarking on correctness issues for the translations just given; in what sense are they "right"? In the absence of notions of equivalence, this question is hard to answer. The next section helps remedy the situation by defining (bi)simulation for GSTs; this permits us to revisit HyPA, for example, in Section 5 in order to give a different, more satisfactory translation of HyPA terms into GSTs.

4 (Bi)Simulations for Generalized Synchronization Trees

Section 2 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 6), in what follows we focus our attention on simulation.

4.1 Simulations for Generalized Synchronization Trees

Simulation relations in Definition 6 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 8 (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 9 (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 10 (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 11 (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 $\lambda \in T_p \to T_q$ 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.

4.2 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 1. *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 2. *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 6. The next definition defines a notion of $\xrightarrow{a}$ for discrete GSTs.

Definition 12 (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 $p' \succ p'' \succ p$.*

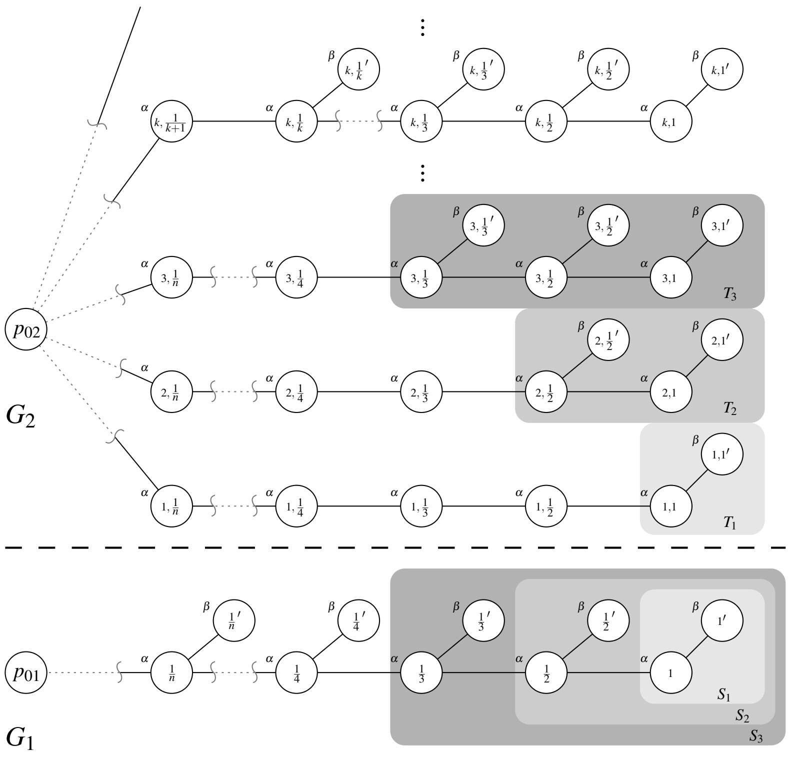


Fig. 1. Visualization of the GSTs used in proof of Theorem 4

2. *$G\lceil p$, 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' / $P'/\{p\}$.*
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, $G' = G\lceil p_0'$, 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 6 may now be applied to discrete GSTs. We have the following.

Theorem 3. *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 4. *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 Figure 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" $T = \{1/n \mid n \in \mathcal{N}\backslash\{0\}\} \cup \{0\}$, 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 αs 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 leadings from the start node of G_1 has βs enabled at every intermediate node, and this behavior does not exist in any trajectory leading from the start node of G_2. $\square$

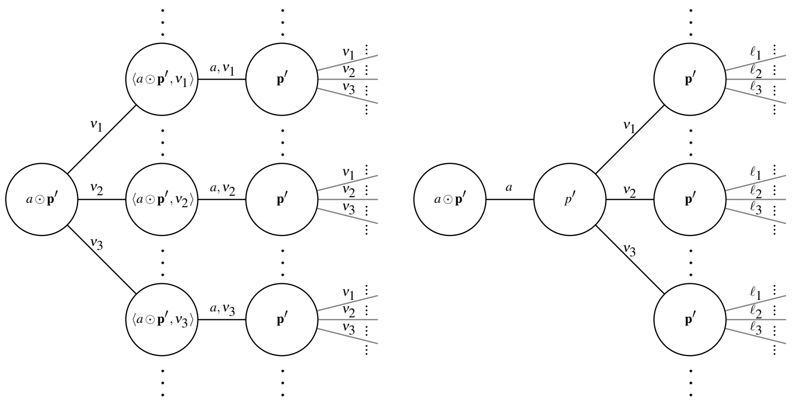
The preceding result suggests that simulation is more nuanced for GSTs than for synchronization trees. One naturally may wonder which of the two notions proposed in this section is the "right" one. Our perspective is that the strong notion possesses a sense of invariance that one might expect for simulation; 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.

5 Constructing GSTs and Implications for Bisimulation

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 [6].

In Example 1, reference was made to the operational semantics of HyPA being given as a hybrid transition system; the states in the transition were HyPA-term / variable-valuation pairs. Bisimulation may be defined as usual for such a transition system. The authors of [6] 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 as way that two HyPA terms are statelessly bisimilar iff the corresponding



$a\odot\mathbf{p}'$
v_1
v_2
v_3
$\langle a\odot\mathbf{p}', v_1\rangle$
$\langle a\odot\mathbf{p}', v_2\rangle$
$\langle a\odot\mathbf{p}', v_3\rangle$
a, v_1
a, v_2
a, v_3
$\mathbf{p}'$
a
p'
ℓ_1
ℓ_2
ℓ_3

Fig. 2. Constructing GSTs from HyPA Terms

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.

Definition 13 (Stateless bisimulation [6]). *Given a hybrid transition system with state space $\mathcal{T}(\mathcal{V}_p) \times Val$, a* ***stateless bisimulation relation*** *is a binary relation $\mathsf{R} \subseteq \mathcal{T}(\mathcal{V}_p) \times \mathcal{T}(\mathcal{V}_p)$ such that for all $\nu, \nu' \in Val$ and $\mathbf{p}\mathsf{R}\mathbf{q}$,*

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

For simplicity we ignore $\checkmark$ in what follows. The construction of GST $G_\mathbf{p}$ from HyPA term $\mathbf{p}$ is exemplified in Figure 2. 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 5. *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.*

There are yet other ways to represent HyPA processes as GSTs. For example, the behavioral systems in Example 2 suggest 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.

6 Composition of GSTs

One of the motivations for this work is to provide a framework for defining composition operators for systems having discrete / non-discrete behavior. In this section, we illustrate the potential of GSTs for this purpose by showing how a version of CSP parallel composition may be defined as a GST construction.

The parallel composition operator we consider is notated $|S|$, where S is a set of action labels. Given two (discrete) systems P and Q, $P\,|S|\,Q$ interleaves the executions of P and Q, with the following exception: actions in S must be performed by *both* P and Q in order to for $P\,|S|\,Q$ to perform them. The precise semantics of the operator may be given via the following SOS rules.

$$\boxed{\frac{P \xrightarrow{a} P' \quad a \notin S}{P\,|S|\,Q \xrightarrow{a} P'\,|S|\,Q}} \quad \boxed{\frac{Q \xrightarrow{a} Q' \quad a \notin S}{P\,|S|\,Q \xrightarrow{a} P\,|S|\,Q'}} \quad \boxed{\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q' \quad a \in S}{P\,|S|\,Q \xrightarrow{a} P'\,|S|\,Q'}}$$

Defining this operator in the GST setting requires first identifying the non-discrete analog of "interleaved execution." Recall that for a GST, the analog of an execution is a (bounded) trajectory (cf. Definition 8). Interleaving two such trajectories can then be formalized as a linearization of the partial order obtained by taking the union of the trajectories. To formalize these ideas, first recall that if two partial orders $\langle P, \preceq_P\rangle$ and $\preceq Q, \preceq_Q$ are disjoint (i.e. $P \cap Q = \emptyset$), then $\langle P \cup Q, \preceq_P \cup \preceq_Q\rangle$ is also a partial order. Interleavings can now be defined as follows.

Definition 14 (S-synchronized Interleaving). *Let $G_1 = \langle P_1, \preceq_1, p_1, \mathcal{L}_1\rangle$ and $G_2 = \langle P_2, \preceq_2, p_2, \mathcal{L}_2\rangle$ be GSTs, and WLOG assume that $P_1 \cap P_2 = \emptyset$. Also let T_1 and T_2 be trajectories (cf. Definition 8) from the roots of G_1 and G_2, respectively. Also let $S \subseteq L$. Then total order $\langle Q, \preceq_Q\rangle$ is an S-*synchronized interleaving *of T_1 and T_2 iff there exists a monotonic bijection $\lambda \in \{p \in T_1 \mid \mathcal{L}_1(p) \in S\} \to \{p \in T_2 \mid \mathcal{L}_2(p) \in S\}$ 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. *$Q = \{p \in T_1 \mid \mathcal{L}_1(p) \notin S\} \cup \{p \in T_2 \mid \mathcal{L}_2(p) \notin S\} \cup \{\langle p, \lambda(p)\rangle \mid \mathcal{L}_1(p) \in S\}$.*
3. *Define $\pi_1 \in Q \to (T_1 \cup T_2)$ by $\pi_1(p) = p$ if $p \in T_1 \cup T_2$ and $\pi_1(\langle p_1', p_2'\rangle) = p_1'$ 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.

Intuitively, an S-synchronized interleaving of two trajectories from different (disjoint) trees is a total ordering on the union of the execution that respects the individual orderings from each of the trees in isolation while requiring synchronization on events in S. The bijection λ in the definition is used to identify the synchronization partners in the trajectories. We may now define the CSP parallel composition construction on GST as follows.

Definition 15. *Let $G_1 = \langle P_1, \preceq_1, p_1, \mathcal{L}_1 \rangle$ and $G_2 = \langle P_2, \preceq_2, p_2, \mathcal{L}_2 \rangle$ be GSTs with $P_1 \cap P_2 = \emptyset$. Then the GST $G_1 \,|S|\, G_2 = \langle Q, \preceq_Q, q_0, \mathcal{L}_Q \rangle$ is given by:*

1. $Q = \{\langle p_1, p_2 \rangle\} \cup \{T \mid T \in I_S(T_1, T_2)$ for some trajectories $T_1 = (p_1, p_1']$ of G_1, $T_2 = (p_2, p_2']$ of $G_2\}$.
2. *$q \preceq_Q q'$ iff $q = \langle p_1, p_2 \rangle$, or $q = \langle r, \preceq_r \rangle$, $q' = \langle r', \preceq_{r'} \rangle$, and $\preceq_r \subseteq \preceq_{r'}$.*
3. *$q_0 = \langle p_1, p_2 \rangle$.*
4. *Let $q \in Q$ and let $p' = \sup(q)$. Then define $\mathcal{L}_Q$ according to*

$$\mathcal{L}_Q(q) = \begin{cases} \mathcal{L}_1(p') & \textit{if } p' \in P_1 \\ \mathcal{L}_2(p') & \textit{if } p' \in P_2 \\ \mathcal{L}_2(p_1') & \textit{if } p' = \langle p_1', p_2' \rangle \end{cases}$$

To justify this construction, the following theorem shows that the definition coincides with the standard one for discrete systems (i.e. those modeled using labeled transition systems).

Theorem 6. *Let G_T be the GST associated with a labeled transition system (LTS) T as given in Section 3. Then we have the following for LTSs T_1 and T_2.*

$$G_{(T_1 \,|S|\, T_2)} \sim_w G_{T_1} \,|S|\, G_{T_2}.$$

This result establishes that for discrete-time systems, the GST construction coincides with the standard one for labeled transition systems. However, Theorem 6 depends on the prohibition of unbounded trajectories in part 1 of Definition 15. This suggests that there is a non-trivial interplay between the properties of the parallel composition operator and the trajectories that are permitted in Definition 15. We regard this as an interesting direction for future research.

7 Conclusions and Directions for Future Research

This paper has defined Generalized Synchronization Trees, which are intended to provide a modeling framework for composition operations on systems that may contain non-discrete time. Like Milner's synchronization trees, GSTs are also trees, but are based on earlier, non-inductive definitions of these structures that permit discrete as well as non-discrete behavior to be modeled uniformly. The work then considers notions of simulation and bisimulation for GSTs, establishing that definitions that coincide in the purely discrete setting of synchronization trees nevertheless differ in the generalized setting. It is then shown how a hybrid process algebra can be captured cleanly in our formalism, and also how a notion of parallel composition may be interpreted at a construction on GSTs.

There are numerous directions for future work. The framework in this paper makes no mention of real-time; indeed the definitions of simulation given in this paper impose no restrictions on preserving duration information when matching up trajectories. This is by design, as one of the interesting observations to emerge is that one can have continuous as well as discrete notions of logical time. Nevertheless, enhancing the framework to accommodate metric notions of time would permit the embedding of various hybrid and real-time models into trees and the development of general notions of composition as a result. Describing other composition operations, and studying their congruence properties *vis à vis* (bi)simulation would yield useful insights into the algebra of GSTs. Finally, developing parsimonious mechanisms *à la* SOS rules for defining composition operations coinductively would simplify their definition and open up insights into the meta-theory of GSTs.

References

1. Abramsky, S.: A domain equation for bisimulation. Information and Computation 92(2), 161–218 (1991)
2. Aceto, L., Bloom, B., Vaandrager, F.: Turning SOS rules into equations. Information and Computation 111(1), 1–52 (1994)
3. Aczel, P.: Non-Well-Founded Sets. CSLI Lecture Notes, vol. 14. Center for the Study of Language and Information (1988)
4. Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.): Handbook of Process Algebra. North-Holland, Amsterdam (2001)
5. Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. J. ACM 42(1), 232–268 (1995)
6. Cuijpers, P.J.L., Reniers, M.A.: Hybrid process algebra. The Journal of Logic and Algebraic Programming 62(2), 191–245 (2005)
7. Cuijpers, P.J.L.: Prefix Orders as a General Model of Dynamics. In: 9th International Workshop on Developments in Computational Models, CONCUR (2013)
8. Davoren, J.M., Tabuada, P.: On Simulations and Bisimulations of General Flow Systems. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 145–158. Springer, Heidelberg (2007)
9. Girard, A., Pappas, G.J.: Approximate bisimulation: A bridge between computer science and control theory. Eur. J. Control 17(5-6), 568–578 (2011)
10. Haghverdi, E., Tabuada, P., Pappas, G.J.: Unifying Bisimulation Relations for Discrete and Continuous Systems. In: Proceedings of the International Symposium MTNS 2002, South (2002)
11. Haghverdi, E., Tabuada, P., Pappas, G.J.: Bisimulation relations for dynamical, control, and hybrid systems. Theoretical Comput Science 342, 229–261 (2005)
12. Jacobs, B., Rutten, J.: A tutorial on (co)algebras and (co)induction. EATCS Bulletin 62, 62–222 (1997)
13. Jech, T.: Set Theory. Academic Press (1978)
14. Julius, A.A., van der Schaft, A.J.: Bisimulation as congruence in the behavioral setting. In: Proceedings of the 44th IEEE Conference on Decision and Control and 2005 European Control Conference, pp. 814–819 (2005)
15. Milner, R.: A Calculus of Communication Systems. LNCS, vol. 92. Springer, Heidelberg (1980)

16. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
17. Rounds, W.C.: On the relationships between Scott domains, synchronization trees, and metric spaces. Information and Control 66(1-2), 6–28 (1985)
18. Sangiorgi, D.: An introduction to bisimulation and coinduction. Cambridge University Press, Cambridge (2012)
19. Willems, J.C.: On interconnections, control, and feedback. IEEE Transactions on Automatic Control 42(3), 326–339 (1997)
20. Winskel, G.: Synchronization Trees. Theoretical Computer Science 34(1-2), 33–82 (1984)