Compositional Expressiveness of Hybrid Automata

Jafar Akhundov, Michael Reißner, and Matthias Werner

Operating Systems Group, TU Chemnitz, Germany
jafar.akhundov | michael.reissner | matthias.werner@cs.tu-chemnitz.de

Abstract. Linear time-invariant hybrid automata (LTI-HA) have been introduced to model space missions in early design phases. One of LTI-HA's main objectives was therefore to allow a composition of (sub) models. In this paper, we evaluate the expressiveness of the composition in LTI-HA. To compare their expressiveness with the existing hybrid automata formalisms, this work proposes a formal notion of compositional expressiveness. In contrast to the more traditional proofs using simulation relations or bisimilarity to compare single models or their respective behavioral expressiveness or equivalence, compositional expressiveness relies on the complexity of the models and the composition operators enabling the engineer to invest less effort in the modeling process. The following text provides a comparative study of the LTI-HA and several other hybrid formalisms, such as linear hybrid automata and the hybrid I/O automata, with respect to their compositional expressiveness. Specifically, adequacy of their application is discussed based on the case study in space mission feasibility verification.

1 Introduction

Most of the real systems consist of a set of subsystems brought together e.g. by physical aspects or function. A compositional approach to modeling and analysis of such a system relies exclusively on the models of the subsystems, without any holistic information about the composed system, an approach that was first formalized by Gottlob Frege in 1923 [FP93] [Fre05].

For describing space systems during early design phases, a hybrid formalism was proposed in [ATW16] [ASGW16] [ATW15] - Linear Time-Invariant Hybrid Automata (LTI-HA). The adequacy of LTI-HA was demonstrated in [ARW17] by providing an operational semantics for the space mission domain-specific language proposed by Schaus et. al [STF+13]. LTI-HA address the issue of combining continuous dynamics of different discrete states by applying the superposition principle in the composition operator which is extensively applied in the classical control theory as well as the theory of hybrid systems [LA14].

While expressiveness is usually considered from the point of view of behavioral comparison of two formalisms on a meta level, comparison of expressiveness from the compositional point of view lacks the deserved attention [Cas05] [BCH+13] [SA06]. Although many extensions for compositional semantics exist

for hybrid formalisms, only few of them are discussed in terms of expressiveness of models [SY96] [Sif99] [BS00]. Although not desirable, this is an understandable state of affairs since compositional expressiveness of a modeling method is rather a question of taste and usability.

The contribution of the following paper lies in the introduction of the notion of compositional expressiveness and its application for comparison of LTI-HA with other hybrid automata (HA) formalisms, such as linear hybrid automata and hybrid I/O automata. The formal considerations in this work are supported by an use-case based comparison of expressive power of the LTI-HA.

The remaining paper is structured as follows: the next section discusses some of the related work, followed by several supporting definitions in section 3. Section 4 is the core of this paper. First, two motivational examples are provided from the application domain of early spacecraft modeling. Then, several formal metrics are introduced to support the formal definition of the compositional expressiveness which is immediately applied to the LTI-HA formalism with respect to hybrid I/O automata (HIOA) and linear as well as rectangular HA (LHA and RHA, respectively). The paper is finalized with a discussion of behavioral expressiveness of LTI-HA models and some concluding remarks.

2 Related Work

As such, LTI-HA are closely related to the switched, piecewise affine and complementarity systems [LA14]. In [GT04], a class of piecewise affine automata with superposition support is briefly introduced to be immediately applied for modeling biological protein regulatory networks. However, as discussed in [ATW16] and [ATW18], support for invariants and discrete resets could be problematic for the composition. The composition operator has also not been formally introduced for this promising formalism [GT04].

The general framework for timed compositional modeling formalism - timed automata with deadlines where invariants of the classical timed automata are replaced with the notion of deadlines - taken by Sifakis and Yovine [SY96] was extended to hybrid systems in [BS00] by Bornot and Sifakis. They discuss the possible compositional semantics of actions for hybrid systems. It is also assumed that compatible automata operate on disjunct state spaces [BS00].

Van der Shaft and Schumacher [SS01] investigate compositionality from the point of view of dynamic systems and discuss some important properties of the composition operator such as commutativity and associativity.

3 Definitions

The following definitions capture the essential terms necessary for the later discussions [ATW18].

Definition 1 (LTI-HA). *A linear time-invariant hybrid automaton $\mathcal{H}$ is a tuple $(\mathcal{L}, \mathcal{X}, \mathcal{S}_I, \mathcal{S}_O, \mathcal{T}, \mathcal{F})$, where:*

- $\mathcal{L} = \{L_1, \ldots, L_n\}$ *is a set of discrete* ***locations*** *(or* ***modes****);*
- $\mathcal{X}$ *is a set of continuous real-valued variables, called* ***state variables****. Time in this context refers to a designated variable* $t \in \mathcal{X}$*.*
- $\mathcal{S}_I$ *and* $\mathcal{S}_O$ *are two disjoint sets of* ***input and output events****, respectively, which define the event signature of the automaton;*
- $\mathcal{T} \subseteq \mathcal{L} \times 2^{C(\mathcal{X})} \times 2^{\mathcal{S}_I} \times 2^{\mathcal{S}_O} \times \mathcal{L}$ *is a guarded (not necessarily complete)* ***transition relation****, where* $\times$ *denotes a Cartesian product and* $C(\mathcal{X})$ *is a set of all possible constraints over* $\mathcal{X}$*. The* ***guard*** *is thus a triple* $(C, \mathcal{E}, \mathcal{A})$ *with a set of constraints* $C \in 2^{C(\mathcal{X})}$ *a set of input events* $\mathcal{E} \in 2^{S_I}$ *and a set of output events* $\mathcal{A} \in 2^{S_O}$*. The locations along with the transitions (with possible loops) constitute the* ***control graph*** *representing the structure of the given hybrid automaton;*
- *The change of any* $x \in \mathcal{X}$*, except for the time reference* t*, at any time point is described by the* ***flow function*** f_L *of the currently active location describing the continuous change of system state* $x_i(t) = f_{x_i,L}(t)$*. We are restricting the flow functions only to those which are valid solutions for ordinary linear time-invariant differential equations of some order* $k \geq 1$*. This restriction guarantees that flow functions fulfill the superposition property. Therefore, for two simultaneously active locations* L_1 *and* L_2 *of two concurrent hybrid automata the resulting rate of change of a global continuous variable* x_i *is defined as* $x_i(t) = f_{x_i,L_1}(t) + f_{x_i,L_2}(t)$*. Let* $\mathcal{F}$ *denote the set of flow functions for every location in* $\mathcal{L}$ *and* $\dot{t} = 1$ *for all* $L \in \mathcal{L}$*.*

Definition 2 (Initial configuration). $\mathcal{I}$ *is the* ***initial state/configuration*** *of the system* $\sigma(t_0) = (L_\mathcal{I}, V_\mathcal{I})$*, where* $L_\mathcal{I} \in \mathcal{L}$ *is the initial active mode,* $V_\mathcal{I} : \mathcal{X} \mapsto \mathbb{R}$ *is the initial valuation of all the variables in* $\mathcal{X}$ *and* $V_\mathcal{I}(t) = t_0$*.*

Definition 3 (Composition operator). *Any two LTI-HA* $\mathcal{H}^1$ *and* $\mathcal{H}^2$ *for which holds* $\forall x \in \mathcal{X}^1 \cap \mathcal{X}^2 : V_\mathcal{I}^1(x) = V_\mathcal{I}^2(x)$ *are called* compatible*. Given two compatible hybrid automata, the parallel composition* $\mathcal{H}^1 || \mathcal{H}^2$ *produces a new hybrid automaton* $\mathcal{H}^c = (\mathcal{L}^c, \mathcal{X}^c, \mathcal{S}_I^c, \mathcal{S}_O^c, \mathcal{T}^c, F^c)$ *with the initial configuration* $\mathcal{I}^c$*, where the components are defined as follows:*

1. $\mathcal{L}^c = \mathcal{L}^1 \times \mathcal{L}^2 = \{(L_1^1, L_1^2), \ldots, (L_{n_1}^1, L_{n_2}^2)\} =$
   $\{L_{11}^c, L_{12}^c, \ldots, L_{1n_2}^c, \ldots, L_{n_1 n_2}^c\}$
2. $\mathcal{X}^c = \mathcal{X}^1 \cup \mathcal{X}^2$
3. $\mathcal{S}_I^c = (\mathcal{S}_I^1 \backslash \mathcal{S}_O^2) \cup (\mathcal{S}_I^2 \backslash \mathcal{S}_O^1)$
4. $\mathcal{S}_O^c = \mathcal{S}_O^1 \cup \mathcal{S}_O^2$
5. $\forall$ *transitions* $(L_i^x, C^x, \mathcal{E}^x, \mathcal{A}^x, L_k^x), (L_j^y, C^y, \mathcal{E}^y, \mathcal{A}^y, L_l^y)$*:*
   (a) if $\mathcal{E}^x \cap \mathcal{S}_O^y = \emptyset$*:* $\forall L \in \mathcal{L}^y$*:* $\exists((L_i^x, L), C^x, \mathcal{E}^x, \mathcal{A}^x, (L_k^x, L)) \in \mathcal{T}^c$*,*
   (b) $\exists(L_{ij}, C^x \cup C^y, \mathcal{E}^x \cup (\mathcal{E}^y \backslash \mathcal{A}^x), \mathcal{A}^x \cup \mathcal{A}^y, L_{kl}) \in \mathcal{T}^c$*,*
   if $((\mathcal{E}^y \cap \mathcal{A}^x = \mathcal{E}^y \cap \mathcal{S}_O^x)$ *and* $(\mathcal{S}_O^y \cap \mathcal{E}^x = \emptyset))$
   or $((\mathcal{E}^x \cap \mathcal{S}_O^y = \emptyset)$ *and* $(\mathcal{E}^y \cap \mathcal{S}_O^x = \emptyset))$
   where either $x = 1, y = 2$*, or vice versa.*

6. $\forall f_{x_k,L_i}(t) \in \mathcal{F}^1, f_{x_k,L_j}(t) \in \mathcal{F}^2$: $f_{x_k,L_{ij}}(t) = f_{x_k,L_i}(t) + f_{x_k,L_j}(t)$
7. $\mathcal{I}^C = (L^c_{\mathcal{I}_1\mathcal{I}_2}, V^1_{\mathcal{I}} \cup V^2_{\mathcal{I}})$

The operator is not defined for not compatible LTI-HA.

4 Compositional Expressiveness

The following section provides two motivational examples: one informally discusses a use case when set-based directional event semantics is more expressive than any semantics based on a singleton labels, and the second shows how superposition principle allows a modeling engineer to apply the divide and conquer approach more effectively and therefore reduces modeling effort.

4.1 Example 1

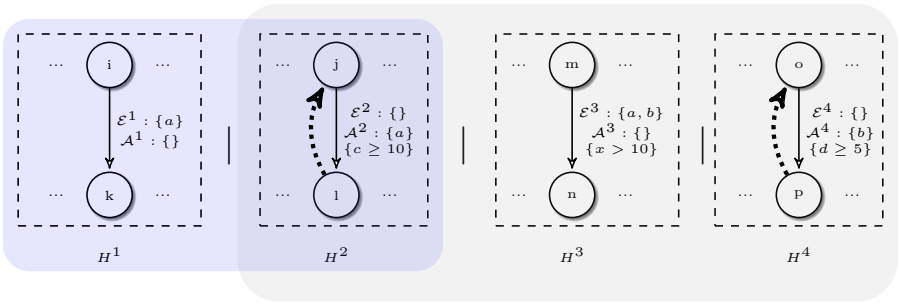

Fig. 1. Expressiveness of the LTI-HA set-based directional event semantics

Consider the interaction patterns between the four transitions in the four LTI-HA depicted in Figure 1. The automata are representing, from left to right: a downlink module of a satellite moving on a Low Earth Orbit(LEO), primary ground station A, synchronization module and the secondary ground station B. Without loss of generality, we are only considering the parts of the automata representing interactions with one another and not the specifics of their dynamics. Synchronization events a and b represent availability of the ground stations A and B, respectively. Synchronization module is responsible for synchronizing the clocks of the satellite, a procedure carried out only when both ground stations are available. This is done to eliminate possible synchronization faults (see, for example, [DHSS95] or [LMS86]).

It is clear that transitions $j \to l$ and $o \to p$ can be taken independently, while the transitions $i \to k$ and $m \to n$ cannot. The transition $m \to n$ can only be taken simultaneously with the both enabled transitions $j \to l$ and $o \to p$. The transition $i \to k$ will only be synchronized with the transition $j \to l$. The paths of non-zero length from l to j and p to o are represented with thick dotted arrows. If the locations j and o can simultaneously be active with valuations $\{c \geq 10, d < 5\}$ and $\{c \geq 10, d \geq 5\}$, then, obviously, transitions $j \to l$ and $o \to p$ are not always taken synchronously.

For HIOA or structured HIOA (SHIOA [MLL06] [Mit07]) to model this system, a relabling procedure consisting of two modifications may be performed: first, both ground stations have to be modeled as a single automaton; second, the transition with two events being simultaneous, a new event has to be introduced, $e = \{a, b\}$ which would synchronize with the $m \to n$ transition. The same holds for the timed automata if the continous dynamics could be modeled with only clock variables [BY04].

Set-based undirectional hybrid automata formalism like [RTP96] may be suitable for modeling the system described in Figure 1. However, the composition operator in [RTP96] only considers synchronous actions in case when there is a non-empty intersection between them. Let us assume that the transitions $i \to k$, $m \to n$ and $o \to p$ are labeled with $\Sigma_{i\to k} = \{a\}$, $\Sigma_{m\to n} = \{a, b\}$ and $\Sigma_{o\to p} = \{b\}$, respectively. The above interaction is indeed preserved. However, if we first compose $\mathcal{H}^1$ with $\mathcal{H}^4$ and only then with $\mathcal{H}^3$, the information about possible simultaneity of events a and b is lost. Since composition operator in [RTP96, p.7] only considers the labeled dependencies between transitions pairwise, transition with $\Sigma_{m\to n} = \{a, b\}$ will never be enabled. However, merging $m \to n$ first with either $i \to k$ or $o \to p$ solves the problem. Hence, composition operator in [RTP96, p.7] is not associative. Our composition operation is both associative and supports event directions making them more intuitional.

Obviously, not set-based undirectional based are less expressive than the last two discussed methods. This clearly demonstrates the advantage of the directional set-based event synchronization: directionality makes events more intuitional while assigning more than one label to a transition, more complex communication patterns are possible.

4.2 Example 2

As a second example, consider the two automata from [ATW18] in Figure 2.

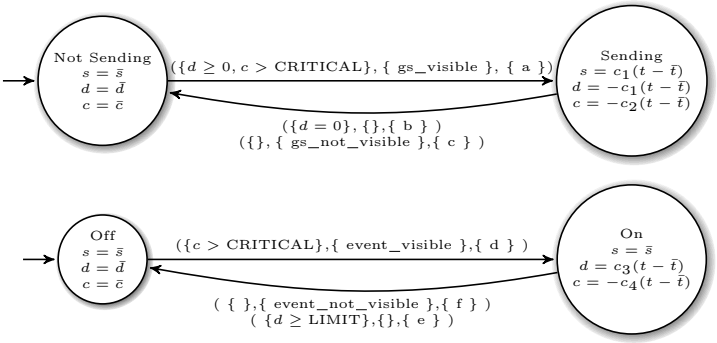

Fig. 2. Downlink Satellite Component and Experimental Payload (Camera)

In the early conceptual phase of development, the satellite downlink module which sends gathered information back to Earth is modeled as having only two distinct states: *Sending*, when a ground station is visible and there is data to send, or *Not Sending*, when either no ground station is available or there is no

data to be sent (or both). In the *Sending* state the rate of change of available data and sent data is the same with opposite signs, whereas in the *Not Sending* state both parameters remain constant.

The valuable data which is sent back to Earth comes from a camera which can rotate and fix on the events of interest. Hence, depending on whether an event is visible or not, the camera can also be in two states: On and Off. The data can only be written if there is enough free storage, therefore the constant *LIMIT* represents the maximum data that can be stored at any instance. The constant *CRITICAL* stands for the lowest level of battery charge needed for any of the components to start.

The continuous variables s, d, c stand for the amount of data sent, the data still stored on the satellites storage unit and the battery charge, respectively. c_1-c_4 are some predefined constants. The downlink component can only be activated when a ground station is visible which is represented by the *gs_visible* event.

Since LTI-HA explicitly support the superposition principle for flow functions, output trajectories do not need to be exclusive. This allows to combine effects of both automata on the same continuous variable, in this case all three of them: s, d, c by the composition operator [ATW18].

(S)HIOA explicitly forbid superposition of the output trajectories [LSV03, p.131,p.141] [Mit07, p.35]. That is, for any formalism without support for superposition to model a system where some of the components have overlapping continuous output trajectories, it is necessary to model all of those components as a single automaton with the cartesian product of all of the corresponding locations.

Other hybrid automata formalisms follow either the same strategy as (S)HIOA or imply an agreement of flow conditions between the locations [ATW18]. Thus, they would also require an engineer to explicitly specify all of the possible combinations of flows thus eliminating the advantages of compositional analysis altogether.

4.3 Generalization and Metrics

Transitions Given m LTI hybrid automata $\mathcal{H}^1, ..., \mathcal{H}^m$, we define a synchronization function sync: $\mathcal{T}_\cup \mapsto 2^{\mathcal{T}_\cup}$ that maps the transition τ^i to the set of all transitions which synchronize with it:

$$\text{sync}(\tau^i) = \begin{cases} \{\tau^j | \sigma_{\tau^i} = \sigma_{\tau^j}, i \neq j\} & \text{for non-directional event semantics (LHA [ACHH93] [NOSY93] [Hen00], RHA [Ras05] [PV94] [Hen00], HA [Áb12])} \\ \{\tau^j | \Sigma_{\tau^i} \cap L_j = \Sigma_{\tau^j} \cap L_i \neq \emptyset, i \neq j\} & \text{for set-based non-directional event semantics (LHA [RTP96])} \\ \{\tau^j | \sigma_{a,\tau^i} = \sigma_{e,\tau^j}, i \neq j\} & \text{for directional event semantics (HIOA [LSV03], TA [BY04])} \\ \{\tau^j | \mathcal{A}_{\tau^i} \cap \mathcal{E}_{\tau^j} = \mathcal{S}^i_O \cap \mathcal{E}_{\tau^j} \neq \emptyset, i \neq j\} & \text{for set-based event semantics (LTI-HA [ATW18])} \end{cases}$$

where $\mathcal{T}_\cup = \bigcup_{i=1..m} \mathcal{T}^i$ with $\mathcal{T}^i$ from $\mathcal{H}^i$, σ_{τ^i} is the label on the transition τ^i in the automaton $\mathcal{H}^i$, $\sigma_{e,\tau^i}, \sigma_{a,\tau^i}$ are the input and output labels on the

transition τ in the automaton $\mathcal{H}^i$. Σ_{τ^i} is a set of labels assigned to transition τ^i respectively. L_i is the set of labels of the LHA i [RTP96]. τ^i is any transition out of $\mathcal{T}_\cup$. Without loss of generality, we assume that for the directional event semantics, there is always only a single automaton generating every consumable event [ATW18] [LSV03].

The function $rsync : \mathcal{T}_\cup \mapsto 2^{\mathcal{T}_\cup}$ maps the set of all transitions τ^i to the set of all transitions which generate the necessary inputs for the transition at hand:

$$\text{rsync}(\tau^i) = \begin{cases} \text{same as sync}(\tau^i) & \text{for non-directional event semantics} \\ & \text{(LHA [ACHH93] [NOSY93] [Hen00],} \\ & \text{RHA [Ras05] [PV94] [Hen00], HA [Áb12])} \\ \text{same as sync}(\tau^i) & \text{for set-based non-directional event semantics} \\ & \text{(LHA [RTP96])} \\ \{\tau^j | \sigma_{e,\tau^i} = \sigma_{a,\tau^j}, i \neq j\} & \text{for directional event semantics} \\ & \text{(HIOA, TA)} \\ \{\tau^j | \mathcal{E}_{\tau^i} \cap \mathcal{A}_{\tau^j} = \mathcal{E}_{\tau^i} \cap \mathcal{S}_O^j \neq \emptyset, i \neq j\} & \text{for set-based event semantics} \\ & \text{(LTI-HA)} \end{cases}$$

Therefore, $rsync$ defines a different set of values in the cases of directional event semantics. Table 1 provides a comparative study of compositional properties defined below for HA depending on the types of event semantics: directional or non-directional, set-based or singular (singleton).

$|sync(\tau^i)|$ provides the number of transitions from other automata with whom the given transition may synchronize. For the directed event semantics, output events are usually observable by any other automata [LSV03] [ATW18]. Obviously, for all the formalisms, this number is greater equal than zero.

The $|\text{rsync}(\tau^i)|$-row represents the number of transitions having as outputs the inputs of τ^i. Since the inverse function is defined in the same way as the original for the undirectional event semantics, no change is observed here.

$\forall j : |\{\tau^j | \tau^j \in \text{rsync}(\tau^i), \mathcal{E}_{\tau^j} \cap \mathcal{A}_{\tau^j} \subset \mathcal{E}_{\tau^j} \cap \mathcal{H}^i\}|$ is the number of transitions which possibly generate the necessary input for the given transition but do not have all the events generated by the automaton containing them. This is a critical condition to be fulfilled during composition, if there is a single producer for each event. It is guaranteed by the condition 5b for the composition operator of LTI-HA and the condition (3) for the event synchronization in [RTP96, p.7].

$|\{s | s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i)$ with $\tau^l \in s', \exists \tau_j \in s : j = l\}|$ is the number of hybrid automata generating the necessary events for a given transition. In the case of singular directional event semantics it is only possible to synchronize with a single producer of an event for the HIOA [LSV03] [Mit07]. Timed automata in the UPPAAL [BLP+96] allow for several generators of a single event. This row is emphasized since this is a case of an increased expressiveness of set-based labeling formalisms in contrast to singular labeling mechanisms.

$\max(|s | s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i)$ with $\tau^l \in s', \exists \tau_j \in s : j = l\}|)$ represents the maximum number of transitions generating the necessary events for the given transition, per automaton. For all the cases, the number of those transitions can be non-zero, however, for the cases of set-based labelings,

the events should be exactly those which lie in the intersection set of the whole automaton and the given transition τ^i, since synchronization (and structural merging during composition) only occur pairwise in the existing formalisms.

The following row is another case where set-based approaches have a definite advantage with respect to singular cases: it represents the number of possible transitions from other automata which can consume *subsets* of the given event label. Obviously, no such thing exists for singular cases, and only those having strictly the same labeling will be able to sychronize.

Properties \ Event Semantics	Non-directional		Directional	
	Singular	Set-based	Singular	Set-based
Examples	[ACHH93] [NOSY93] [Hen00]	[RTP96]	[LSV03] [Mit07] [BY04]	[ATW18]
$\|sync(\tau^i)\|$	≥ 0	≥ 0	≥ 0	≥ 0
$\|\text{rsync}(\tau^i)\|$	> 0	≥ 0	> 0	≥ 0
$\forall j: \|\{\tau^j \mid \tau^j \in \text{rsync}(\tau^i), \mathcal{E}_{\tau^i} \cap \mathcal{A}_{\tau^j} \subset \mathcal{E}_{\tau^i} \cap \mathcal{S}_O^j\}\|$	-	0	-	0
$\|\{s \mid s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i) \text{ with } \tau^l \in s', \exists \tau_j \in s : j = l\}\|$	0 or 1	≥ 0	0 or 1 (> 0 [BY04])	≥ 0
$\max(\|s \mid s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i) \text{ with } \tau^l \in s', \exists \tau_j \in s : j = l\}\|)$	≥ 0	≥ 0	≥ 0	≥ 0
$\forall A^i$ from $\tau^i \in \mathcal{T}_\cup, \|\mathcal{A}^i\| > 1$: $\|\tau^j$ from $\mathcal{H}^j$, $j \neq i$ with $s = \mathcal{E}^j \cap \mathcal{A}^i \subset \mathcal{A}^i, s \neq \emptyset\|$	0	≥ 0	0	≥ 0
Circular event dependencies	no	no	yes	yes
Stutter transitions	yes	yes	no	$\mathcal{E}^i = \emptyset, \mathcal{A}^i = \emptyset, \mathcal{C}^i = \emptyset$

Table 1. Event semantics properties (B $\triangleq$ Blocking, NB $\triangleq$ Non-blocking)

Before we discuss the next row of the table, the following definitions are due.

Definition 4 (Event Dependency of Guards). *Two guards - $g_1 = (C_1, \mathcal{E}_1, \mathcal{A}_1)$ from $\mathcal{H}^1, g_2 = (C_2, \mathcal{E}_2, \mathcal{A}_2)$ from $\mathcal{H}^2$ - are said to be **event-dependent** iff $\mathcal{E}_1 \cap \mathcal{A}_2 \neq \emptyset \vee \mathcal{E}_2 \cap \mathcal{A}_1 \neq \emptyset$. The guard g_1 is said to be **event-dependent** on the second automaton iff $\mathcal{E}_1 \cap \mathcal{S}_O^2 \neq \emptyset$.*

Event (in)dependency of guards consequently implies event-(in)dependency of the corresponding transitions. It is clear from the constraint $\mathcal{S}_I \cap \mathcal{S}_O = \emptyset$ that those two guards cannot be in the same automaton.

Definition 5 (Cycle of Event Dependencies for a Set of Guards). *A set of guards G of size $k \geq 2$ has a cycle of event dependencies if there is a sequence of these guards $(g_0, ..., g_{k-1})$, such that $\forall i, 0 \leq i < k : \mathcal{A}_i \cap \mathcal{E}_{(i+1) mod\ k} \neq \emptyset$.*

Clearly, circular event dependencies are only possible for the directional cases. The last row shows wether the formalism has an explicit support for stutter transitions. HIOA have eliminated them in the later redefinitions to avoid compositionality issues [LSV03]. LTI-HA do not have implicit stutter transitions. However, they can be explicitly defined, as provided in the Table 1. These transitions, however, are always enabled and will be taken immediately when their

start-location is entered. They also pose a danger of introducing time locks (time convergence), should there be a cycle containing only stutter transitions.

The relabling procedure discussed in the first example of this section, can be formalized as follows:

$$\forall \tau^i \text{ from } \mathcal{H}^i \text{ where } |\mathcal{E}_{\tau^i}| > 1 : \ \forall \mathcal{H}^j \text{ with } \mathcal{S}_O^j \cap \mathcal{E}_{\tau^i} \neq \emptyset : \ \Big|\Big|_{\mathcal{H}^j} \tag{1}$$

The labeled event set $\mathcal{E}_{\tau^i}$ will be then renamed to a single letter in the composed automaton. Although we do not account here for the cascading dependencies of the events, it should be clear that in the worst case all of the automata $\mathcal{H}^j,\ j = 1..m$ will have to be combined with each other by the user. E.g. consider the scenario where one LTI-HA consumes all of the events generated by other LTI-HA over a single transition. Furthermore, for any set of event labels $\mathcal{A}$, all of the possible subset of those events can be consumed by a different automaton. That is, for every possible set from $2^{\mathcal{A}}$, a new transition has to be introduced in the automaton containing a transition labeled with $\mathcal{A}$.

Locations As discussed in the second example of this section, the modeling engineer has to account for all of the possible combinations of continuous flows which could become exponential in the minimum number of states of the HA, if for all $|\mathcal{L}^i|$ holds $|\mathcal{L}^i| \geq 2$. That is, for 10 LTI-HA with three states per automaton, 3^{10} locations would be required to model in any formalism not supporting superposition: i.e. (S)HIOA, LHA, RHA, HA, etc.

Formally, in the worst case for a set of m LTI hybrid automata $\mathcal{H}^1, ..., \mathcal{H}^m$:

$$\text{if } \exists X \subseteq \mathcal{X}_U, \mathcal{X}_U = \bigcup_{i=1..m} \mathcal{X}^i \text{ where } \forall x \in X, \forall L_j^i : \ \dot{x}_{L_j^i} \neq 0 \tag{2}$$

$$\text{then } \mathbb{L} = \bigtimes_{i=1}^{m} \mathcal{L}^i, \ |\mathbb{L}| = \prod_{i=1}^{m} |\mathcal{L}^i|$$

where $\mathbb{L}$ is the total set of the composed locations.

Definition of Compositional Expressiveness Considering the above metrics, we can now define the notion of compositional expressiveness for hybrid automata. It amounts to the information a modeling engineer uses to describe the system components before the composed system is built. Clearly, since the composition operator has exponential run-time and produces a structure with an exponential number of nodes, it is desirable to shift the modeling complexity into the "pre-composition phase" since the modeling effort is then linear in the number of nodes and edges.

We rely on the number of edges ($\mathcal{T}$) as well as the number of vertices ($\mathcal{L}$) needed to describe a system S. We assume that for both formalisms under comparison, a minimal representation can be achieved, and that there is some behavioral equivalence satisfied for models of S in the respective formalisms (e.g. bisimilarity).

Definition 6 (Compositional Expressiveness). *A modeling formalism A is compositionally more expressive than a modeling formalism B ($A \ni B$) if, for describing a system S (or a family of systems):*

$$1.\ \mathcal{T}_A(S) < \mathcal{T}_B(S)$$
$$2.\ \mathcal{L}_A(S) < \mathcal{L}_B(S)$$

for a minimal representations of S in A, $\mathcal{H}_A(S)$, and B, $\mathcal{H}_B(S)$, respectively.

Proposition 1. *For a family of systems where the the condition from the equation (2) holds or $\exists \tau^i$ from $\mathcal{T}_\cup$ with $|\mathcal{E}_{\tau^i}| > 1$, linear time-invariant hybrid automata are compositionally more expressive than HIOA.*

Proof. The proof follows from the discussions and metrics from the this section.

5 Discussion

This paper has introduced an initial approach at formalizing the notion of compositional expressiveness in terms of events labeling and continuous flow combinations. It has been directly applied to demonstrate the superior compositional expressiveness of LTI-HA with respect to HIOA.

Timing of transitions was never considered in this work, as well as the modal extensions of guards [SY96, p.5]. In the extended version of this work, we plan to include timeliness of events as well as the modalities for combining the guards which are assigned to the transitions into the notion of compositional expressiveness. We also intend to adopt the notion of HA comparability of Lynch et. al. [LSV03] and put the LTI-HA in the context of other formalisms in terms of behavioral expressiveness.

Associativity is an important property for composition of discrete event systems in general and hybrid automata in particular [SS01] [CL10]. Proving it for LTI-HA is not trivial due to the complex event semantics and remains as a future challenge.

The relabling procedure discussed in section 4 can potentially be extended to a full algorithm which would transform a system of LTI-HA into a system of HIOA which is a powerful modeling method with tool support [Fre05]. This algorithm would also require a behavioral expressiveness relation established between the two formalisms which we also intend as a future work. A formal meta-language that could be used to define simulation relations could be the extension theorem [BS00] [SY96] or timed transition systems [Cas05] [BCH$^+$13].

References

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.

[ARW17] Jafar Akhundov, Michael Reißner, and Matthias Werner. Using Hybrid Automata for Early Spacecraft Design Evaluation. In *Proceedings of the 26th International Workshop on Concurrency, Specification and Programming*, Warsaw, Poland, September 2017.

[ASGW16] Jafar Akhundov, Volker Schaus, Andreas Gerndt, and Matthias Werner. Using Timed Automata to Check Space Mission Feasibility in the Early Design Phases. In *IEEE Aerospace 2016 Proceedings*, Big Sky, Montana, USA, March 2016.

[ATW15] Jafar Akhundov, Peter Tröger, and Matthias Werner. Considering Concurrency in Early Spacecraft Design Studies. In *CS&P 2015 Proceedings*, pages 22–30, Rzeszow, Poland, 2015.

[ATW16] Jafar Akhundov, Peter Tröger, and Matthias Werner. Superposition Principle in the Composable Hybrid Automata. In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*, pages 125–140, Rostock, Germany, September 2016.

[ATW18] Jafar Akhundov, Peter Tröger, and Matthias Werner. Superposition Principle in Composable Hybrid Automata. *Fundamenta Informaticae*, 157(Concurrency, Specification, and Programming: Special Issue of Selected Papers of CS&P 2016):321–339, 2018.

[BCH+13] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O.H. Roux. The expressive power of time Petri nets. *Theoretical Computer Science*, 474:1–20, February 2013.

[BLP+96] Johan Bengtsson, Fredrik Larsson, Paul Pettersson, Wang Yi, Palle Christensen, Jesper Jensen, Per Jensen, Kim Larsen, and Thomas Sorensen. *UPPAAL: a Tool Suite for Validation and Verification of Real-Time Systems*. 1996.

[BS00] Sébastien Bornot and Joseph Sifakis. On the Composition of Hybrid Systems. In M. Kemal Inan and Robert P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, pages 293–322. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[BY04] Johan Bengtsson and Wang Yi. Timed Automata: Semantics, Algorithms and Tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, pages 87–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Cas05] B. Bérard F. Cassez. Comparison of the expressiveness of timed automata and time Petri nets. In *In Proc. FORMATS'05, vol. 3829 of LNCS*, pages 211–225. Springer, 2005.

[CL10] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[DHSS95] Danny Dolev, Joseph Y. Halpern, Barbara Simons, and Ray Strong. Dynamic fault-tolerant clock synchronization. *J. ACM*, 42(1):143–185, January 1995.

[FP93] G. Frege and G. Patzig. *Logische Untersuchungen*. Kleine Reihe Vandenhoeck und Ruprecht. Vandenhoeck & Ruprecht, 1993.

[Fre05] Goran Fedja Frehse. *Compositional verification of hybrid systems using simulation relations*. [Sl: sn], 2005.

[GT04] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *IEE Proceedings - Systems Biology*, 1(1):170–183, June 2004.

[Hen00] ThomasA. Henzinger. The Theory of Hybrid Automata. In M.Kemal Inan and RobertP. Kurshan, editors, *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series*, pages 265–292. Springer Berlin Heidelberg, 2000.

[LA14] Hai Lin and Panos J. Antsaklis. Hybrid Dynamical Systems: An Introduction to Control and Verification. *Foundations and Trends® in Systems and Control*, 1(1):1–172, 2014.

[LMS86] Leslie Lamport and P. M. Melliar-Smith. Byzantine Clock Synchronization. *Operating Systems Review*, 20(3):10–16, 1986.

[LSV03] Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid I/O Automata. *Inf. Comput.*, 185(1):105–157, August 2003.

[Mit07] Sayan Mitra. *A Verification Framework for Hybrid Systems*. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007.

[MLL06] Sayan Mitra, Daniel Liberzon, and Nancy Lynch. Verifying Average Dwell Time by Solving Optimization Problems. In Ashish Tiwari and Joao P. Hespanha, editors, *Hybrid Systems: Computation and Control (HSCC 06)*, LNCS, Santa Barbara, CA, March 2006. Springer.

[NOSY93] Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. An approach to the description and analysis of hybrid systems. *Hybrid Systems*, pages 149–178, 1993.

[PV94] Anuj Puri and Pravin Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In David L. Dill, editor, *Computer Aided Verification: 6th International Conference, CAV '94 Stanford, California, USA, June 21–23, 1994 Proceedings*, pages 95–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[Ras05] Jean-François Raskin. An Introduction to Hybrid Automata. In Dimitrios Hristu-Varsakelis and William S. Levine, editors, *Handbook of Networked and Embedded Control Systems*, pages 491–517. Birkhäuser Boston, Boston, MA, 2005.

[RTP96] R. Alur, T. A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, March 1996.

[SA06] S. Di Cairano and A. Bemporad. An Equivalence Result between Linear Hybrid Automata and Piecewise Affine Systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 2631–2636, December 2006.

[Sif99] Joseph Sifakis. The Compositional Specification of Timed Systems - A Tutorial. In *Proceedings of the 11th International Conference on Computer Aided Verification*, CAV '99, pages 2–7, London, UK, UK, 1999. Springer-Verlag.

[SS01] A.J. van der Schaft and J. M. Schumacher. Compositionality issues in discrete, continuous, and hybrid systems. *International Journal of Robust and Nonlinear Control*, 2001.

[STF+13] Volker Schaus, Michael Tiede, Philipp M. Fischer, Daniel Lüdtke, and Andreas Gerndt. A Continuous Verification Process in Concurrent Engineering. In *AIAA Space Conference*, September 2013.

[SY96] Joseph Sifakis and Sergio Yovine. Compositional Specification of Timed Systems (Extended Abstract). In *STACS*, 1996.

[Áb12] Erika Ábrahám. *Modeling and Analysis of Hybrid Systems: Lecture Notes*. RWTH Aachen University, April 2012.