Language for Hybrid Input/Output Automata

Sayan Mitra
mitras@theory.lcs.mit.edu

Yong Wang
y_wang@mit.edu

January 11, 2003

1 Hybrid I/O Automata

In this section we briefly review the HIOA mathematical. For a complete discussion of the model refer to [5]. Earlier versions of the model appeared in [6] and [7].

1.1 The Mathematical Model

A hybrid I/O automaton captures the hybrid behavior of a system in terms of discrete transitions and continuous evolution of its state variables. Let V be the set of variables of automaton $\mathcal{A}$. Each $v \in V$ is associated with a *(static) type* defining the set of values v can assume. A *valuation* $\mathbf{v}$ for V is a function that associates each variable $v \in V$ to a value in $type(v)$. A trajectory τ of V is defined as a mapping $\tau : J \rightarrow val(V)$ where J is a left closed interval of time. If J is right closed then τ is said to be *closed* and its *limit time* is the supremum of the domain of τ, also written as $\tau.ltime$. Each variable $v \in V$ is also associated with a *dynamic type* (or *dtype*) which is the set of trajectories that v may follow. Dynamic types must satisfy the *time-shift, subinterval* and *pasting* closure properties described in [5]. A hybrid I/O automaton $\mathcal{A}$ is a tuple $(X, U, Y, Q, \Theta, H, I, O, \mathcal{D}, \mathcal{T})$ where

- X: set of *internal or state variables*, U : set of *input variables*, Y : set of *output variables*. The set of variables $V \triangleq U \cup Y \cup X$. The set of *locally controlled variables* $Z \triangleq X \cup Y$.
- H : set of *internal actions*, I : set of *input actions* , O : set of *output actions*. The set of actions $A \triangleq H \cup I \cup O$.
- $Q \subseteq val(X)$: a set of states
- $\Theta \subseteq Q$: non-empty set of *start states*.
- $\mathcal{D} \subseteq Q \times A \times Q$: set of *discrete transitions*. A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is written in short as $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$.
- $\mathcal{T}$: set of *trajectories* for V, such that for every trajectory τ in $\mathcal{T}$, and for every $t \in \tau.dom$, $\tau(t).X \in Q$. It is required that $\mathcal{T}$ is closed under prefix, suffix, and concatenation. The first state $\tau(0).X$ of trajectory τ is written as $\tau.fstate$. Similarly if $\tau.dom$ is finite then $\tau.lstate = \tau(\tau.ltime).X$.

In addition, a hybrid I/O automaton also satisfies: (1) the input action enabling property, which prevents it from blocking any input action and (2) the input trajectory enabling property, which ensures that it is able to accept any trajectory of the input variables either by allowing time to progress for the entire length of the trajectory or by reacting with some internal action before that.

An *execution* of $\mathcal{A}$ is a finite or infinite sequence of actions and trajectories $\zeta = \tau_0, a_1, \tau_1, a_2 \ldots$,where (1) each $\tau_i \in \mathcal{T}$, (2) $\tau_0.fstate \in \Theta$ and (3) if τ_i is not the last trajectory in ζ then τ_i is finite and $\tau_i.lstate \stackrel{a_{i+1}}{\rightarrow} \tau_{i+1}.fstate$. An execution is *closed* if the sequence is finite and the domain of the final trajectory is a finite closed interval. The length of an execution is the number of elements (actions and trajectories) in the sequence.

1.2 Added Structure

In the earlier works [3, 8, 4] using the HIOA model, trajectories of automata were specified using an ad hoc mixture of integral, algebraic equations and English. While this form of specification is simple to read, it does not lend itself easily to systematic analysis, nor does it enforce a consistent style in writing specifications. The specification language we use in this paper uses "state space" representation of the trajectories. This representation is concise, natural, and widely used in the analysis of dynamical systems. To make this representation work, we have introduced extra structure into the basic HIOA model of [5].

We assume that the time domain is R. A variable v is *discrete* if its dynamic type is the pasting closure of the set of constant functions from left closed intervals of time to $type(v)$. A variable is *continuous* if its dynamic type is the pasting closure of the set of continuous functions from left closed intervals of time to R. For any set S of variables, S_d and S_a refers to the discrete and continuous subsets of S respectively. The following are the first two restrictions we impose on the HIOA model:

R1 Every variable is either discrete or continuous.
R2 Locally controlled discrete variables remain constant over trajectories, that is, $\tau.lval\lceil Z_d = \tau.fval\lceil Z_d$, for all $\tau \in \mathcal{T}$.

Hence in this model, the evolution of the locally controlled variables of a HIOA is completely specified by the discrete transitions and the evolution of the variables in Z_a. The state space representation is used to specify this evolution as explained in the next part.

1.2.1 State Model and Activity

Let e be a real valued algebraic expression involving the variables in $X \cup U$. For a given trajectory τ we use $\tau.e$ to denote the function with domain $\tau.dom$ that gives the value of the expression e at all times during trajectory τ. Given that v is a locally controlled continuous variable, a trajectory τ satisfies the algebraic equation

$$v = e,$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) = \tau.e(t)$.

If an algebraic equation involves a nondeterministic choice such as

$$v \in [e_1, e_2],$$

then trajectory τ satisfies the equation if for every $t \in \tau.dom$, $\tau \downarrow v(t) \in [\tau.e_1(t), \tau.e_2(t)]$.

If the expression e is integrable when viewed as a function, then τ satisfies the differential equation

$$\dot{v} = e,$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) = \tau \downarrow v(0) + \int_0^t \tau.e(t')\ dt'$.

A *state model* of HIOA $\mathcal{A}$ consists of $|Z_a|$ number of algebraic and/or differential equations with exactly one equation having v or $d(v)$ as its left hand side. The right hand sides of the equations are algebraic expressions involving the variables in $X \cup U$. It is also required that there are no circular relationships between the state variables.

A state model specifies[1] the evolution of every variable v in Z_a from some initial valuation. A trajectory τ satisfies a state model E if at all times in $\tau.dom$, all the variables in Z_a satisfy the differential and algebraic equations in E with $\tau(0)$ defining the initial valuations.

An *activity* α of HIOA $\mathcal{A}$ consists of three components:

$P \subseteq Q$, the *operating condition*,
$P^+ \subseteq Q$, the *stopping condition*,
E, a state model for $\mathcal{A}$.

The set of trajectories defined by activity α is denoted by $[\alpha]$. A trajectory τ belongs to the set $[\alpha]$ if the following conditions hold:

- τ satisfies the state model E.
- For all $t \in \tau.dom$, $\tau \downarrow X(t) \in P$.
- If $\tau \downarrow X(t) \in P^+$ for $t \in dom(\tau)$ then τ is closed and $t = \tau.ltime$.

The set of trajectories of an automaton is defined to be the union of all the sets of trajectories specified by the activities of an automaton. Suppose automaton $\mathcal{A}$ has n activities, namely α_i for $i \in \mathcal{I}$, where $\mathcal{I}$ is an arbitrary index set. Then $\mathcal{T}_\mathcal{A} = \cup_{i \in \mathcal{I}} [\alpha_i]$.
With the present model it would be possible for an automaton to switch activities over a single trajectory in an execution. Our final restriction on the HIOA model prevents such switches. In other words, the switching of activities (or state models) is brought about only by discrete steps:

R3 Operating condition of all the activities are disjoint, that is, $P_i \cap \mathcal{P}_j = \emptyset$ if $i \neq j$.

Note that **R3** does not imply that the choice of an activity from a given state is deterministic. For instance, let $P_i = (x = i)$ be the operating condition of the α_i, $i \in \mathcal{I}$, and x is a previously unused internal variable. These activities satisfy **R3**, and if x is nondeterministically assigned to one of the values in $\mathcal{I}$, then the automaton chooses one of the activities nondeterministically.

It can be proved that a set of trajectories specified by a set of activities respecting **R1**, **R2**, and **R3**, satisfy the prefix, suffix, and concatenation closure properties.

Lemma 1 *Suppose $\mathcal{T}$ is a set of trajectories specified by the activities α_i, $i \in \mathcal{I}$, where $\mathcal{I}$ is an index set. Then $\mathcal{T}$ is closed under prefix, suffix, and concatenation.*

Proof: $\mathcal{T} = \cup_{i=1}^n [\alpha_i]$ is closed under prefix and suffix because each of the sets $[\alpha_i]$ are closed under prefix and suffix. Let $\tau_0, \tau_1, \tau_2, \ldots$ be a sequence of trajectories in $\mathcal{T}$ such that, for each non-final index i, τ_i is closed and $\tau_i.lstate = \tau_{i+1}.fstate$. From the concatenation closure requirement of $\mathcal{T}$, it is necessary that $\tau = \tau_0 \frown \tau_1 \frown \tau_2 \ldots \in \mathcal{T}$.

[1] By *specifies* we mean restricts rather than uniquely determines. Due to possible nondeterminism in the state model, unique determination might be impossible.

Let $\tau_i \in [\alpha_j]$, $\tau_{i+1} \in [\alpha_k]$, where $j, k \in \mathcal{I}$, therefore $\tau_i.lstate \in P_j$ and $\tau_{i+1}.fstate \in P_k$. Let us assume for the sake of contradiction that $j \neq k$. From **R3**, $P_j \cap P_k$ must be empty. But we have $\tau_i.lstate \in P_j \cap P_k$, which contradicts our assumption. Therefore it must be the case that $j = k$. Therefore every trajectory in the sequence belongs to the same activity, say $[\alpha_j]$. As $[\alpha_j]$ is closed under concatenation, $\tau_0 \frown \tau_1 \frown \tau_2 \frown \ldots \in \mathcal{T}$. $\square$

1.3 Language Constructs

Our specification language is based on the above modified HIOA model. Variables are declared by specifying their names, types, dtypes, and optionally their initial valuations. For input variables initial valuations cannot be specified. Varibales declared with the **analog** keyword are continuous, else they are discrete. Algebraic expressions are written using the operators $+, -, *$, and \. An expression involving nondeterministic choice, such as $v \in [e_1, e_2]$, is written as:

$$v = \mathbf{choose}[e_1, e_2]$$

The derivative of a continuous variable x is written as $d(x)$. The discrete transitions are written in the precondition—effect style of the IOA language [2]. An activity $\alpha : (P, P^+, E)$ is written as:

activity α **while** P **evolve** E **stop at** P^+.

For automata with a single activity, if the operating condition P is not specified explicitly, then it is assumed to be the entire state space of the automaton. If the stopping condition is omitted then it is assumed to empty.

Example 1.1 Consider n vehicles traveling on a single track. Figure 1 shows the specification of a single vehicle $\texttt{VEHICLE}_i$, where $i = 1, \ldots, n$. The state variables are the position x_i, the velocity $\dot{x}_i$, and $mode_i$. The values of *mode* correspond to different dynamics of a vehicle. In the `stopped` mode the vehicle is in rest with zero acceleration. In the `cruising` mode the vehicle moves with a constant velocity and in the `accelerating` mode the acceleration of the vehicle is between c_{amax} and c_{amin} such that its velocity is always positive. An external controller triggers the actions which set the value of *mode*. The *accelerate* (*resp. brake*) action sets *mode* to `accelerating` (`stopped`). The *release* action sets the *mode* to `cruising`, when the vehicle is in `accelerating` mode. The internal *cruise* action is enabled when $mode = \texttt{accel}$ and $\dot{x} = c_{vmax}$. The stopping-condition in the *accelerating* activity ensures the *cruise* action occurs when it is enabled.

2 Composing Activities

In this section we define the composition of activities for hybrid I/O automata.

2.1 Composition of pre-HIOAs

The composition of HIOAs and pre-HIOAs have been defined in [5]. A pre-HIOA $\mathcal{A}_1$ is allowed to be composed with another pre-HIOA $\mathcal{A}_2$ if (1) the sets of internal actions and variables of $\mathcal{A}_i$ are disjoint from the sets of actions and variables of $\mathcal{A}_j$ respectively, and (2) the sets of output variables and output actions of $\mathcal{A}_i$ are disjoint from those of $\mathcal{A}_j$ respectively, for $i, j \in \{1, 2\}$ and $i \neq j$. Formally, two pre-HIOAs are *compatible* if for $i \neq j$,

$$X_i \cap V_j = Y_i \cap Y_j = H_i \cap A_j = O_i \cap O_j = \emptyset$$

type MODES = **enumeration of** {accelerating, cruising, stopped}

hybridautomaton $\texttt{VEHICLE}_i \quad i \in \{1, \ldots, n\}$

actions
input $accelerate_i$, $brake_i$, $release_i$,
internal $cruise_i$

variables
internal $mode_i$: MODES := stopped,
internal analog x_i : Real := 0, $\dot{x_i}$: Real := 0,

discrete transitions
input $accelerate_i$
eff $mode_i$:= accelerating;

input $brake_i$
eff $mode_i$:= braking;

input $release$
eff if $mode_i$ = accelerating **then** $mode_i$:= cruising **fi** ;

internal $cruise$
pre $\dot{x_i} \geq c_{vmax} \wedge mode_i$ = accelerating
eff $mode_i$:= cruising **fi** ;

trajectories
activity $accelerating_i$
while $mode_i$ = accelerating $\wedge\ \dot{x_i} > 0$
evolve $d(\dot{x_i})$ = **choose** $[c_{amin}, c_{amax}]$; $d(x_i) = \dot{x_i}$
stop at $\dot{x_i} = c_{vmax}$

activity $cruising_i$
while $mode_i$ = cruising
evolve $d(\dot{x_i}) = 0$; $d(x_i) = \dot{x_i}$

activity $stopped_i$
while $mode_i$ = stopped
evolve $\dot{x_i} = 0$; $d(x_i) = \dot{x_i}$

Figure 1: HIOA specification of supervisor automaton

For compatible pre-HIOA $\mathcal{A}_1$ and $\mathcal{A}_2$, their composition $\mathcal{A}_1 || \mathcal{A}_2$ is defined to be $\mathcal{A} = (U, Y, X, Q, \Theta, I, O, H, D, \mathcal{T})$ where

- $U = (U_1 \cup U_2) - (Y_1 \cup Y_2)$,
- $Y = Y_1 \cup Y_2$,
- $X = X_1 \cup X_2$,
- $Q = \{\mathbf{x} \in val(X) \mid \mathbf{x} \lceil X_1 \in Q_1 \wedge \mathbf{x} \lceil X_2 \in Q_2\}$,
- $\Theta = \{\mathbf{x} \in Q \mid \mathbf{x} \lceil X_1 \in \Theta_1 \wedge \mathbf{x} \lceil X_2 \in \Theta_2\}$,
- $I = (I_1 \cup I_2) - (O_1 \cup O_2)$,
- $O = O_1 \cup O_2$,
- $H = H_1 \cup H_2$,
- For each $\mathbf{x}, \mathbf{x}' \in Q$ and each $a \in A$, $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$ iff for $i = 1, 2$, either (1) $a \in A_i$ and $\mathbf{x} \lceil X_i \xrightarrow{a}_i \mathbf{x}' \lceil X_i$, or (2) $a \notin A_i$ and $\mathbf{x} \lceil X_i = \mathbf{x}' \lceil X_i$.
- $\mathcal{T} \subseteq trajs(V)$ is given by $\tau \in \mathcal{T} \iff \tau \downarrow V_1 \in \mathcal{T}_1 \wedge \tau \downarrow V_2 \in \mathcal{T}_2$.

In our specification language we describe the trajectories of an automaton by a set of activities. So for composing automata we have to combine activities of compatible automata in a meaningful way. Suppose pre-HIOA $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible and their trajectories are defined as follows:

$$\mathcal{T}_i = \cup_{j=1}^{k_i} [\alpha_{ij}], \text{ where } \alpha_{ij} = (P_{ij}, P_{ij}^{+}, E_{ij}), \text{ for } i = 1, 2$$

The combination of the set of activities of $\mathcal{A}_1$ and $\mathcal{A}_2$ should give us a set of activities for the composed automaton $\mathcal{A}$ such that the set of trajectories specified by this set is equal to $\mathcal{T}$. Before defining combination of activities we describe a method for combining state models.

2.2 Merging State Models

Suppose E_1 and E_2 are state models for compatible automata $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. The *merging* of E_1 and E_2 is defined as the state model $E = E_1 \otimes E_2$ containing all the equations of E_1 and E_2 and no other equation.

Lemma 2 *If E_1 and E_2 are state models of compatible pre-HIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, then $E_1 \otimes E_2$ is a state model for the composed automaton $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$.*

Proof: Let Z_a and Z_{ia} be the set of locally controlled continuous variables of $\mathcal{A}$ and $\mathcal{A}_i$ respectively, for $i = 1, 2$. From compatibility, $Z_a = Y_a \cup X_a = Z_{1a} \cup Z_{2a}$. The merged state model E has $|Z_a|$ equations each of which is from either E_1 or E_2. Every variable $v \in Z_a$ appears in the left hand side of exactly one of the equations in E as v or $d(v)$. The right hand sides of the equations in E are expressions involving variables in $X \cup U$.

The above lemma can be generalized to the merging of n compatible state models.

Lemma 3 *If $E_{1i_1}, \ldots, E_{ni_n}$ are state models of n compatible pre-HIOAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$ respectively, then $E = E_{1i_1} \otimes \ldots \otimes E_{ni_n}$ is a state model for the composed automaton $\mathcal{A} = \mathcal{A}_1 || \ldots || \mathcal{A}_n$.*

Proof: Directly extending the above lemma.

Example 2.1 Selecting one state model for each of the n vehicles (from Example 1.1) we can merge them together to get a single state model of all n vehicles. can be merged to get a single state model for all n vehicles. Let α_{j1}, α_{j2}, and α_{j3} denote the activities $accelerating_j, cruising_j$, and $stopped_j$ respectively for $1 \leq j \leq n$. The j^{th} vehicle has three activities $\alpha_{ji_j} : (P_{ji_j}, P^+_{ji_j}, E_{ji_j})$, for each $i_j \in \{1, 2, 3\}$. Let $E_{i_1 \ldots i_n}$ be the state model obtained by combining $E_{1i_1} \otimes \ldots \otimes E_{ni_n}$, where $i_j \in \{1, 2, 3\}$, for each $1 \leq j \leq n$. The state model $E_{i_1 \ldots i_n}$ is given by the following set of equations:

For each j, in $1 \leq j \leq n$,

$$\begin{array}{lll}
d(\dot{x}_j) & = \textbf{choose } [c_{amin}, c_{amax}] & \text{if } i_j = 1 \\
d(\dot{x}_j) & = 0 & \text{if } i_j = 2 \\
\dot{x}_j & = 0 & \text{if } i_j = 3 \\
 & & \\
d(x_j) & = \dot{x}_j &
\end{array}$$

2.3 Combining Activities

The activity combination operation constructs the activities of a composed automaton from the activities of its components. If α_1 and α_2 are activities of compatible automata $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, then their *combination* $\alpha_1 \odot \alpha_1$ is defined to be the activity $\alpha : (P, P^+, E)$ where

- $P = \{\mathbf{x} \in val(X) \mid \mathbf{x}\lceil X_1 \in P_1 \wedge \mathbf{x}\lceil X_2 \in P_2\}$,
- $P^+ = \{\mathbf{x} \in val(X) \mid \mathbf{x}\lceil X_1 \in P_1^+ \vee \mathbf{x}\lceil X_2 \in P_2^+\}$,

- $E = E_1 \otimes E_2$.

Compatibility of $\mathcal{A}_1$ and $\mathcal{A}_2$ ensures that $P \subseteq Q$, $P^+ \subseteq Q$, and that $E_1 \otimes E_2$ is a state model of $\mathcal{A}$. Therefore α is an activity of automaton $\mathcal{A}$. The combination operation can be extended to n compatible activities in a straightforward way.

Lemma 4 *Suppose $\mathcal{A}_1 \ldots \mathcal{A}_n$ are compatible pre-HIOAs. Let α_{ij} be the j^{th} activity of $\mathcal{A}_i$, where $1 \leq j \leq m_i$, m_i being the number of activities of $\mathcal{A}_i$. The combination $\alpha_{1j_1} \odot \ldots \odot \alpha_{nj_n}$ is an activity of the composed automaton $\mathcal{A} = \mathcal{A}_1 || \ldots || \mathcal{A}_2$, where $1 \leq j_i \leq m_i$ for $1 \leq i \leq n$.*

Theorem 1 *Suppose α_1 and α_2 are activities of compatible pre-HIOA $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, then $[\alpha_1 \odot \alpha_2] = \{\tau \in trajs(V) \mid \tau \downarrow V_1 \in [\alpha_1] \wedge \tau \downarrow V_2 \in [\alpha_2]\}$.*

Proof: Let $\mathcal{A}$ denote $\mathcal{A}_1 || \mathcal{A}_2$ and let $\mathcal{T} = \{\tau \in trajs(V) \mid \tau \downarrow V_1 \in [\alpha_1] \wedge \tau \downarrow V_2 \in [\alpha_2]\}$. Consider a trajectory $\tau \in [\alpha_1 \odot \alpha_2]$.

- Compatibility ensures that $\tau \downarrow V_1$ and $\tau \downarrow V_2$ satisfy E_1 and E_2 respectively.
- For all $t \in \tau.dom$, $\tau \downarrow X(t) \in P$, implies that $\tau \downarrow X_1(t) \in P_1$ and $\tau \downarrow X_2(t) \in P_2$.
- If $\tau \downarrow X(t) \in P^+$ for some $t \in \tau.dom$ then either $\tau \downarrow X_1(t) \in P_1^+$ or $\tau \downarrow X_2(t) \in P_2^+$. Suppose $\tau \downarrow X_1(t) \in P_1^+$ and $\tau \downarrow X_2(t) \notin P_2^+$. As τ is closed with $\tau.ltime = t$, $\tau \downarrow V_1 \in [\alpha_1]$ and from prefix closure $\tau \downarrow V_2 \in [\alpha_{2j}]$.

From the above $\tau \in \mathcal{T}$.

Next consider trajectory $\tau \in \mathcal{T}$.

- Since $\tau \downarrow V_1$ satisfies E_1, $\tau \downarrow V_2$ satisfies E_2, compatibility ensures that τ satisfies $E_1 \otimes E_2$.
- For all $t \in \tau.dom$, $\tau \downarrow X_1 \in P_1$ and $\tau \downarrow X_2 \in P_2$,implies that $\tau \downarrow X(t) \in P$.
- Finally, if $\tau \downarrow X_i(t) \in P_i^+$ for some $t \in \tau.dom$, $i = 1$ or 2, then $\tau \downarrow X(t) \in P^+$ and τ is closed with $\tau.ltime = t$.

The above set of conditions imply that $\tau \in [\alpha_{1i} \odot \alpha_{2j}]$. □

The set of activities of the composed automaton $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$ is the union of the activities obtained by pairwise combination of all the activities of $\mathcal{A}_1$ and $\mathcal{A}_2$. .

Theorem 2 *Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible pre-HIOAs and their sets of trajectories are given by $\mathcal{T}_i = \cup_{j=1}^{m_i} [\alpha_{ij}]$ for $i = 1, 2$, where m_i is the number of activities of $\mathcal{A}_i$. Let $\mathcal{A} = \mathcal{A}_1 || \mathcal{A}_2$, and $\mathcal{T}$ be the set of trajectories of $\mathcal{A}$. Then $\mathcal{T} = \cup_{k=1}^{m_1} \cup_{l=1}^{m_2} [\alpha_{1k} \odot \alpha_{2l}]$.*

Proof: Consider any trajectory $\tau \in \mathcal{T}$, from composition $\tau \downarrow V_1 \in \mathcal{T}_1 \wedge \tau \downarrow V_2 \in \mathcal{T}_2$. Therefore for every trajectory $\tau \in \mathcal{T}$, there exists k and l such that $1 \leq k \leq m_1$, $1 \leq l \leq m_2$, $\tau \downarrow V_1 \in [\alpha_{1k}]$, and $\tau \downarrow V_2 \in [\alpha_{2l}]$. Using Theorem 1, $\tau \in [\alpha_{1k} \odot \alpha_{2l}]$, with $1 \leq k \leq m_1$, $1 \leq l \leq m_2$. Therefore $\mathcal{T} \subseteq \cup_{k=1}^{m_1} \cup_{l=1}^{m_2} [\alpha_{1,k} \odot \alpha_{2,l}]$.

Let $\tau \in [\alpha_{1k} \odot \alpha_{2l}]$ for some $1 \leq i \leq m_1$, $1 \leq j \leq m_2$. Using Theorem 1, $\tau \downarrow V_1 \in [\alpha_{1k}] \subseteq \mathcal{T}_1$ and $\tau \downarrow V_2 \in [\alpha_{2l}] \subseteq \mathcal{T}_2$, from compatibility $\tau \in \mathcal{T}$. □

Example 2.2 We denote the three activities *accelerating*$_i$, *cruising*$_i$, and *stopped*$_i$ by α_{i1}, α_{i2}, and α_{i3} respectively, where $i \in \{1, \ldots, n\}$. Let $\alpha_{k_1 \ldots k_n} = \alpha_{1k_1} \odot \ldots \odot \alpha_{nk_n}$, where each $k_i \in \{1, 2, 3\}$, for $1 \leq i \leq n$. Let $\mathcal{A}$ denote the composition of n $\texttt{VEHICLE}_i$ automata and $\mathcal{T}$ be the set of trajectories for $\mathcal{A}$.

$\mathcal{T} = \cup_{k_1=1}^{3} \cdots \cup_{k_n=1}^{3} [\alpha_{k_1 \ldots k_n}]$

Where combined activity $\alpha_{k_1 \ldots k_n} = (P, P^+, E)$ is given by:

$P = \bigwedge_{i=1}^{n}((k_i = 1) \rightarrow (\dot{x_i} > 0)) \wedge \bigwedge_{i=1}^{n}(mode_i = \mathtt{mode}_{k_i})$, where $\mathtt{mode}_1$ `= accelerating`, $\mathtt{mode}_2$ `= cruising`, $\mathtt{mode}_3$ `= stopped`,
$P^+ = \bigvee_{i=1}^{n}((k_i = 1) \rightarrow (\dot{x_i} = c_{vmax}))$
$E = E_{1k_1} \otimes \ldots \otimes E_{nk_n}$ as in Example 2.1.

References

[1] M. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems, 1998.

[2] Stephen Garland, Nancy Lynch, and Mandana Vaziri. IOA: A language for specifying, programming and validating distributed systems. Technical report, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, October 1999.

[3] Carolos Livadas, John Lygeros, and Nancy A. Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99),Phoenix, Arizona*, pages 115–125, December 1999.

[4] Nancy Lynch. A three-level analysis of a simple acceleration maneuver, with uncertainties. In *Proceedings of the Third AMAST Workshop on Real-Time Systems*, pages 1–22, Salt Lake City, Utah, March 1996. World Scientific Publishing Company.

[5] Nancy Lynch, Roberto Segala, and Frits Vaandraager. Hybrid I/O automata. Technical Report MIT-LCS-TR-827b, MIT Laboratory for Computer Science, Technical Report, Cambridge, MA 02139, February 2002. submitted for journal publication, theory.lcs.mit.edu/tds/papers/Lynch/HIOA-final.ps.

[6] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In T. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, New Brunswick, New Jersey, October 1995. Springer-Verlag.

[7] Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Hybrid I/O automata revisited. In M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors, *Proceedings Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC'01),* Rome, Italy, volume 2034 of *lncs.* springer, March 2001.

[8] H. B. Weinberg, Nancy Lynch, and Norman Delisle. Verification of automated vehicle protection systems. In T. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III: Verification and Control (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems)*, volume 1066 of *Lecture Notes in Computer Science*, pages 101–113, New Brunswick, New Jersey, October 1995. Springer-Verlag.