A Compositional Framework for Hybrid Systems

Ben Lukoschus

Institute of Computer Science and Applied Mathematics,
University of Kiel, Germany, bls@informatik.uni-kiel.de

Abstract. We present communicating linear hybrid automata (CLHA) as a modeling framework for the specification of hybrid systems. CLHA provide modular descriptions of system components and subsume most of the characteristics that are used in verification tools, e.g., discrete and continuous transitions, invariants, and communication via shared variables as well as synchronization symbols. The syntax and a compositional semantics of CLHA is defined formally, and propositional linear temporal logic is introduced as an abstract specification language for CLHA behavior. An example illustrates the use of the presented framework.

1 Introduction

Formal reasoning about hybrid systems often involves a lot of different modeling frameworks. Since many systems consist of a wide range of hardware and software subsystems with varying complexities, it is necessary to individually choose models that are precise enough to describe a certain behavior, but on the other hand as coarse as possible to avoid complexity problems during the specification and verification process.

These choices lead to the use of different verification tools and their associated specification languages, e.g., the model checker SMV (Symbolic Model Verifier) [1] for purely discrete systems, KRONOS [2] for timed automata [3], and HyTech [4] and Uppaal [5] for linear hybrid systems [6]. The models used in these tools employ different formalisms for state changes, communication and synchronization. All of them use discrete transition systems, some use continuous transitions. Communication can be implemented by shared variables, synchronization symbols, or both.

This heterogeneous field of models can make formal reasoning about the systems difficult, especially when interfaces between different subsystems are concerned. Therefore, we propose a formal modeling framework that subsumes a wide range of formalisms used in formal verification and that can be simplified on a case-by-case basis to match the input language of the tool one intends to use. This modeling framework presented in the following section.

A similar approach is followed in [7], and in a much more generalized form for (nonlinear) hybrid automata in [8].
2 Communicating Linear Hybrid Automata

This section introduces a formal modeling framework for the description of linear hybrid systems that is capable of subsuming a wide range of modeling paradigms used in formal verification, e.g., discrete or timed automata [3], discrete or timed condition/event systems [9, 10], or linear hybrid automata [6]. We define communicating linear hybrid automata which have the following characteristics:

- discrete control locations,
- input and output variables,
- communication via shared variables,
- communication via synchronization symbols (directed, one-to-many), and
- continuous variables restricted by invariants and linear rates at each control location.

Other models can be embedded into this framework by syntactic restrictions. E.g., timed automata only have one kind of continuous variables called clocks. These are restricted to the fixed rate 1 and can only be set to 0 in discrete transitions.

2.1 Variables

Let $V$ be a finite set of variables, and let $\text{type}$ be a function assigning a type, i.e., a set of possible values like $\mathbb{B}$ (Booleans) or $\mathbb{R}$ (reals), to each variable.

**Definition 1 (Evaluation).** A function $\sigma$ assigning to each variable $v \in V$ a value $\sigma(v) \in \text{type}(v)$ is called evaluation of $V$. We denote the set of all evaluations of $V$ by $\Sigma$.

**Notation.** Given some subset of the variables (e.g., $V_y \subseteq V$), we use $\Sigma$ with the same decorations (e.g., $\Sigma_y^x$) to denote the set of all evaluations of these variables and $\sigma$ with the same decorations (e.g., $\sigma_y^x$) to denote the restriction of a given $\sigma \in \Sigma$ on these variables, if not defined otherwise.

2.2 Syntax

**Definition 2 (Communicating linear hybrid automaton).** A communicating linear hybrid automaton (CLHA)

$$A = (Q, Q_0, V_{\text{out}}, \Sigma_0, R, I, L, E)$$

consists of

- a finite set $Q$ of locations,
- a set $Q_0 \subseteq Q$ of initial locations,
- a set $V_{\text{out}} \subseteq V$ of output variables, constituting the set of input variables $V_{\text{in}} = V \setminus V_{\text{out}}$,
- a set $\Sigma_0 \subseteq \Sigma$ of initial variable evaluations,
– a function \( R : Q \times V^{cout} \rightarrow \mathbb{R} \) assigning a rate at each location to each of the continuous output variables \( V^{cout} = \{ v \in V^{out} \mid \text{type}(v) = \mathbb{R} \} \),
– a function \( I : Q \rightarrow 2^{\Sigma^{cout}} \) assigning an invariant for the continuous output variables to each location,
– a finite set \( L \) of synchronization symbols, consisting of two disjoint sets of input symbols \( L^{in} \) and output symbols \( L^{out} \), and
– a set \( E \) of edges, where each edge \( e = (q, l, p, q') \in E \) consists of a source location \( q \in Q \), a destination location \( q' \in Q \), a set of synchronization symbols \( l \subseteq L \), and a variable transition relation \( \rho \subseteq \Sigma \times \Sigma^{out} \).

### 2.3 Computation Semantics

The computations of a CLHA are defined by changes in three components: its (discrete) location, its variable evaluations, and the synchronization symbols that are communicated between the CLHA and its environment.

Initially, the CLHA is in one of the initial locations \( Q_0 \), and its variables are set to one of the initial variable evaluations \( \Sigma_0 \). These are changed during two different kinds of computation steps:

1. **Discrete step**: The CLHA changes its location and its variables according to one of the edges in \( E \). The transition is instantaneous; time does not progress. Discrete steps can be triggered by synchronization symbols.
2. **Continuous step**: The CLHA remains at its current location for a finite amount of time \( t \in \mathbb{R}^+ \), and the continuous output variables change according to their rate defined in \( R \). The other output variables do not change. During the time period \( t \), the invariant of the current location has to hold. No synchronization symbols are sent during continuous steps.

**Definition 3 (Computations of a CLHA)**. Given a CLHA \( A \) as in Def. 2, a computation of \( A \) is a maximal or infinite sequence

\[
(q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1) \xrightarrow{l_2} (q_2, \sigma_2) \xrightarrow{l_3} \ldots,
\]

where \( q_0 \in Q_0 \), \( \sigma_0 \in \Sigma_0 \), \( q_i \in Q \), \( \sigma_i \in \Sigma \), \( l_i \in 2^{L} \cup \mathbb{R}^+ \), and for all \( i \), one of the following cases applies:

1. **Discrete step**: \( l_{i+1} \subseteq L \wedge \exists \rho \subseteq \Sigma \times \Sigma^{out} : (q_i, l_{i+1}, \rho, q_{i+1}) \in E \wedge (\sigma_i, \sigma^{out}_{i+1}) \in \rho \).
2. **Continuous step**: \( l_{i+1} \in \mathbb{R}^+ \wedge \sigma^{out}_{i+1} = \sigma_i \wedge q_{i+1} \in L^{out} \) \( \forall \rho \leq l_{i+1} : \sigma^{out}_{i} \oplus_R q_{i+1} t \in I(q_i) \), where

\[
(\sigma \oplus_R q \cdot t)(v) = \begin{cases} 
\sigma(v) + t \cdot R(q, v) & \text{if } v \in V^{cout}, \\
\sigma(v) & \text{otherwise}.
\end{cases}
\]

We denote the set of all computations of \( A \) by \( \text{Comp}(A) \).

Note that both discrete and continuous steps do not impose any restrictions on the values of the input variables in \( \sigma_{i+1} \); they can change arbitrarily. Thus, it is possible for \( A \) to accept any changes of these variables by the environment of \( A \) during a transition. This is the basis of the parallel composition of CLHA, which is defined next.
2.4 Parallel Composition

The parallel composition of two CLHA $A_1$ and $A_2$ formally defines how these two automata interact. An obvious prerequisite for a successful composition is that $A_1$ and $A_2$ have no outputs in common, neither variables ($V_1^{\text{out}} \cap V_2^{\text{out}} = \emptyset$) nor synchronization symbols ($L_1^{\text{out}} \cap L_2^{\text{out}} = \emptyset$), since any output should have a single source.

The parallel composition of $A_1$ and $A_2$ has the following characteristics:

- its (initial) location set is the Cartesian product of the (initial) location sets of $A_1$ and $A_2$,
- its set of output variables is the union of the output variables of $A_1$ and $A_2$,
- its set of initial variable evaluations is the intersection of the initial variable evaluations of $A_1$ and $A_2$,
- its rates for the continuous output variables are taken from $A_1$ and $A_2$,
- its invariants are taken from $A_1$ and $A_2$, and
- its set of output synchronization symbols is the union of the respective sets of $A_1$ and $A_2$, whereas the union of the input symbols is reduced by each other’s output symbols. An effect of the output symbols staying visible for other components is a directed one-to-may communication.

Two edges of $A_1$ and $A_2$ can be combined if:

- the edge labeling $l_2$ of $A_2$ contains all the synchronization symbols that the edge labeling $l_1$ of $A_1$ needs as input from $A_2$ (i.e., $l_1 \cap l_1^{\text{in}} \cap l_2^{\text{out}} \subseteq l_2$), and vice versa.

The variable transition relation of the resulting edge combines the elements of the variable transition relations of the edges of $A_1$ and $A_2$ which have a common pre-state $\sigma$.

In our experience this form of communication using sets of input/output symbols instead of undirected synchronization (often with only one symbol) as used in (timed) automata makes modeling of complex systems much easier.

**Definition 4 (Parallel composition of CLHA).** Given two CLHA $A_i = (Q_i, q_0^i, V_i^{\text{out}}, \Sigma_i, R_i, I_i, L_i, E_i), i \in \{1, 2\}$, with $V_1^{\text{out}} \cap V_2^{\text{out}} = \emptyset = L_1^{\text{out}} \cap L_2^{\text{out}}$, the parallel composition of $A_1$ and $A_2$, denoted by $A_1 \parallel A_2$, is defined as the CLHA $A = (Q, q_0, V^{\text{out}}, \Sigma, R, I, L, E)$ with

\begin{align*}
- \quad Q &= Q_1 \times Q_2, \quad Q_0 = Q_0^1 \times Q_0^2, \\
- \quad V^{\text{out}} &= V_1^{\text{out}} \cup V_2^{\text{out}}, \\
- \quad \Sigma &= \Sigma_0^1 \cap \Sigma_0^2, \\
- \quad \text{for all } q_1 \in Q_1, q_2 \in Q_2, \text{ and } v \in V^{\text{out}}, \\
R((q_1, q_2), v) &= \begin{cases} 
R_1(q_1, v) & \text{if } v \in V_1^{\text{out}}; \\
R_2(q_2, v) & \text{if } v \in V_2^{\text{out}},
\end{cases}
\end{align*}

\[ L = L_1 \cup L_2, \quad L^{\text{in}} = (L_1^{\text{in}} \setminus L_2^{\text{out}}) \cup (L_2^{\text{in}} \setminus L_1^{\text{out}}), \quad L^{\text{out}} = L_1^{\text{out}} \cup L_2^{\text{out}}, \]
A composition of the computation sets of
hold.

Lemma 1 (Parallel composition).

Let
Proof.

Definition 5 (Parallel composition of computations).

Given the CLHA

Definition 4 combines two CLHA on the syntactic layer, i.e., by combining their sets of edges. Parallel composition can also be defined semantically, by composing computations. This is formalized in the following definition. As notation we use "∥", the same operator as in Def. 4; these two can be distinguished from each other by looking at the types of the operands (CLHA or sets of computations).

Definition 5 (Parallel composition of computations). Given the CLHA

A composition of the computation sets of
hold.

Lemma 1 (Parallel composition). Given two CLHA

we have

Proof. Let

Then,

if and only if (by Def. 5)

\( \exists (q_0^1, \sigma_0^1) \xrightarrow{l_1^1} (q_1^1, \sigma_1^1) \xrightarrow{l_2^1} \ldots \in \text{Comp}(A_1), \)

\( (q_0^2, \sigma_0^2) \xrightarrow{l_1^2} (q_1^2, \sigma_1^2) \xrightarrow{l_2^2} \ldots \in \text{Comp}(A_2) : \)

\[ \forall i : q_i = (q_i^1, q_i^2) \land \sigma_i = \sigma_i^1 = \sigma_i^2 \land (l_i^1 = l_i^2 = l_i \in \mathbb{R}_{>0} \lor \]

\[ (l_i = l_i^1 \cup l_i^2 \land l_i^1 \cap L_{1i}^\text{in} \cap L_{2i}^\text{out} \subseteq l_i^2 \land l_i^2 \cap L_{2i}^\text{in} \cap L_{1i}^\text{out} \subseteq l_i^1) \]
if and only if (by Def. 3)

$$\exists (q_0, \sigma_0^0) \xrightarrow{l_1} (q_1, \sigma_1^1) \xrightarrow{l_2} \ldots : (q_0, \sigma_0^0) \in Q_0 \land \sigma_0^0 \in \Sigma_0 \land$$

$$\forall i : ((l_{i+1}) \subseteq L_1 \land \exists \rho \subseteq \Sigma \times \Sigma_{\text{out}} : (q_i, l_{i+1}, \rho, q_{i+1}) \in E_1 \land$$

$$(\sigma_i, \sigma_{i+1}^{\text{out}}) \in \rho) \lor$$

$$(l_{i+1} \in \mathbb{R}_{>0} \land q_{i+1} = q_i \land \sigma_{i+1}^{\text{out}} = \sigma_i^{\text{out}} \otimes_{R_1} l_{i+1} \land$$

$$\forall 0 \leq t < l_{i+1} : \sigma_i^{\text{out}} \otimes_{R_1} t \in I_1(q_i)) \land$$

$$\exists (q_0^2, \sigma_0^2) \xrightarrow{l_1^2} (q_1^2, \sigma_1^2) \xrightarrow{l_2^2} \ldots : (q_0^2, \sigma_0^2) \in Q_0^2 \land \sigma_0^2 \in \Sigma_0^2$$

$$\forall i : q_i = (q_i^1, q_i^2) \land \sigma_i = \sigma_i^1 \land \sigma_{i+1}^{\text{out}} = \sigma_i^{\text{out}} \otimes_{R_2} l_{i+1}^2 \land$$

$$\forall 0 \leq t < l_{i+1}^2 : \sigma_i^{\text{out}} \otimes_{R_2} t \in I_2(q_i^2)) \land$$

$$\leftrightarrow \ (\text{rearrange quantors and variables})$$

$$\exists (q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1^1) \xrightarrow{l_2} \ldots : (q_0, \sigma_0) \in Q_0 \land \sigma_0 \in \Sigma_0 \land$$

$$\forall i : q_i = (q_i^1, q_i^2) \land \sigma_i = \sigma_i^1 \land \sigma_{i+1}^{\text{out}} = \sigma_i^{\text{out}} \otimes_{R_1} l_{i+1}^1 \land$$

$$\forall 0 \leq t < l_{i+1}^1 : \sigma_i^{\text{out}} \otimes_{R_1} t \in I_1(q_i^1)) \land$$

$$\leftrightarrow \ (\text{introduce/remove } Q_0, \Sigma_0, R, I, E, \text{ using Def. 4})$$

$$\exists (q_0, q_0^2) \xrightarrow{l_1} (q_1, q_1^2) \xrightarrow{l_2} \ldots : (q_0, q_0^2) \in Q_0 \land \sigma_0 \in \Sigma_0 \land$$

$$\forall i : q_i = (q_i^1, q_i^2) \land \sigma_i = \sigma_i^1 \land \sigma_{i+1}^{\text{out}} = \sigma_i^{\text{out}} \otimes_{R_1} (q_i^1, q_i^2) \land$$

$$\forall 0 \leq t < l_{i+1}^1 : \sigma_i^{\text{out}} \otimes_{R_1} (q_i^1, q_i^2) \in I((q_i^1, q_i^2))$$

$$\leftrightarrow \ (\text{remove/introduce } q_i^1, q_i^2, \sigma_i^1, \sigma_i^2, l_{i+1}^1, l_{i+1}^2)$$

$$q_0 \in Q_0 \land \sigma_0 \in \Sigma_0 \land$$

$$\forall i : (l_{i+1} \subseteq L \land \exists \rho \subseteq \Sigma \times \Sigma_{\text{out}} : (q_i, l_{i+1}, \rho, q_{i+1}) \in E \land (\sigma_i, \sigma_{i+1}^{\text{out}}) \in \rho) \land$$

$$\forall 0 \leq t < l_{i+1} \land \sigma_{i+1}^{\text{out}} = \sigma_i^{\text{out}} \otimes_{R} l_{i+1} \land$$

$$\forall 0 \leq t < l_{i+1} : \sigma_i^{\text{out}} \otimes_{R} t \in I_1(q_i)$$
⇔ (Def. 3)

\[(q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1) \xrightarrow{l_2} \ldots \in \text{Comp}(A)\]

⇔

\[(q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1) \xrightarrow{l_2} \ldots \in \text{Comp}(A_1 \parallel A_2) . \]

\[\Box\]

2.5 Trace Semantics

While the notion of computations is well-suited to describe the way a single CLHA operates, it is not wise to use computations for specifying the interaction of several automata. Computations contain information about the changes of locations, which should be considered as internal and not observable from outside.

Therefore, we remove the locations from the computations and use the resulting sequences, called traces, for our compositional CLHA semantics.

**Definition 6 (Trace of a CLHA).** Given a CLHA \( A \) as in Def. 2, a sequence

\[\sigma_0 \xrightarrow{l_1} \sigma_1 \xrightarrow{l_2} \sigma_2 \xrightarrow{l_3} \ldots\]

is called trace of \( A \) if and only if there exist \( q_0, q_1, q_2, \ldots \in Q \) such that

\[(q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1) \xrightarrow{l_2} (q_2, \sigma_2) \xrightarrow{l_3} \ldots\]

is a computation of \( A \).

Now we define the semantics of a CLHA by its traces:

**Definition 7 (Semantics of a CLHA).** The semantics of a CLHA \( A \), denoted by \( [A] \), is the set of all traces of \( A \).

Definition 5 introduces the parallel composition of computations. Analogously, we can define the parallel composition of traces (we overload the “∥” operator one more).

**Definition 8 (Parallel composition of traces).** Given the two CLHA \( A_1 \) and \( A_2 \) as in Def. 4 and given their sets of traces \( t_1 = [A_1] \) and \( t_2 = [A_2] \), the parallel composition of \( t_1 \) and \( t_2 \), denoted by \( t_1 ∥ t_2 \), is defined as follows: The trace \( \sigma_0 \xrightarrow{l_1} \sigma_1 \xrightarrow{l_2} \ldots \) in \( t_1 \) and \( \sigma_0' \xrightarrow{l_1'} \sigma_1' \xrightarrow{l_2'} \ldots \) in \( t_2 \), where for all indices \( i \),

\[- \sigma_i = \sigma_i^1 = \sigma_i^2 \text{ and} \]

\[- l_i^1 = l_i^2 = l_i \in \mathbb{R}_{>0} \text{ or } l_i = l_i^1 \cup l_i^2 \wedge l_i^1 \cap L_i^{\text{in}} \cap L_i^{\text{out}} \subseteq l_i^2 \wedge l_i^2 \cap L_i^{\text{in}} \cap L_i^{\text{out}} \subseteq l_i^1\]

hold.
Similarly to Lemma 1 for computations we can show that the parallel composition of traces is compositional:

**Lemma 2 (Parallel composition).** Given two CLHA $A_1$ and $A_2$ as in Def. 4, we have

$$[A_1] [A_2] = [A_1 \parallel A_2] .$$

**Proof.** Let $A_1$, $A_2$, and $A$ be given as in Def. 4. Then,

$$\sigma_0 \xrightarrow{l_1} \sigma_1 \xrightarrow{l_2} \ldots \in [A_1] [A_2]$$

if and only if (by Def. 8)

$$\exists \sigma_1^1 \xrightarrow{l_1^1} \sigma_1^2 \xrightarrow{l_2^1} \ldots \in [A_1], \sigma_0^2 \xrightarrow{l_1^2} \sigma_2^2 \xrightarrow{l_2^1} \ldots \in [A_2] :$$

$$\forall i : \sigma_i = \sigma_i^1 = \sigma_i^2 \land (l_i^1 = l_i^2 = l_i \in \mathbb{R} > 0 \lor$$

$$l_i = l_i^1 \cup l_i^2 \land l_i^1 \cap L_{in}^1 \cap L_{out}^2 \subseteq l_i^2 \land l_i^2 \cap L_{in}^2 \cap L_{out}^1 \subseteq l_i^1).$$

if and only if (by Def. 7 and Def. 6)

$$\exists q_0, q_1, \ldots \in Q :$$

$$\exists (q_0^1, \sigma_0^1) \xrightarrow{l_1^1} (q_1^1, \sigma_1^1) \xrightarrow{l_2^1} \ldots \in \text{Comp}(A_1),$$

$$(q_0^2, \sigma_0^2) \xrightarrow{l_1^2} (q_1^2, \sigma_1^2) \xrightarrow{l_2^2} \ldots \in \text{Comp}(A_2) :$$

$$\forall i : q_i = (q_i^1, q_i^2) \land \sigma_i = \sigma_i^1 = \sigma_i^2 \land (l_i^1 = l_i^2 = l_i \in \mathbb{R} > 0 \lor$$

$$l_i = l_i^1 \cup l_i^2 \land l_i^1 \cap L_{in}^1 \cap L_{out}^2 \subseteq l_i^2 \land l_i^2 \cap L_{in}^2 \cap L_{out}^1 \subseteq l_i^1).$$

if and only if (by Def. 5)

$$\exists q_0, q_1, \ldots \in Q :$$

$$(q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1) \xrightarrow{l_2} \ldots \in \text{Comp}(A_1) \parallel \text{Comp}(A_2)$$

if and only if (by Lemma 1)

$$\exists q_0, q_1, \ldots \in Q :$$

$$(q_0, \sigma_0) \xrightarrow{l_1} (q_1, \sigma_1) \xrightarrow{l_2} \ldots \in \text{Comp}(A_1 \parallel A_2)$$

if and only if (by Def. 6)

$$\sigma_0 \xrightarrow{l_1} \sigma_1 \xrightarrow{l_2} \ldots \in [A_1 \parallel A_2] .$$

\[\square\]

### 3 Propositional Linear Temporal Logic

Using the trace semantics of Def. 7, we are able to describe the exact behavior of a CLHA. However, in practice a much simpler description language is sufficient, since one is usually interested in an abstract and finite way to describe the system behavior. One possibility is the use of *temporal logic*. This section introduces *propositional linear temporal logic*, which describes sequences of sets of propositions.
3.1 Syntax

**Definition 9 (PLTL syntax).** Given a countable set of propositions, the set of Propositional Linear Temporal Logic (PLTL) formulae is given by the following BNF notation:

\[
\langle \text{formula} \rangle ::= \langle \text{proposition} \rangle \mid \langle \text{formula} \rangle \land \langle \text{formula} \rangle \mid \neg \langle \text{formula} \rangle \\
\langle \text{formula} \rangle \mathcal{U} \langle \text{formula} \rangle \mid \circ \langle \text{formula} \rangle
\]

Other Boolean operators are defined as abbreviations in the usual way: \( \varphi \lor \psi \equiv \neg((\neg \varphi) \land (\neg \psi)) \), \( \varphi \Rightarrow \psi \equiv (\neg \varphi) \lor \psi \), \( \varphi \equiv \psi \equiv (\varphi \Rightarrow \psi) \land (\psi \Rightarrow \varphi) \). The Boolean constant \( \text{true} \) can be abbreviated as \( p \lor (\neg p) \) for some proposition \( p \), and \( \text{false} \equiv \neg \text{true} \).

In addition to the temporal operators \( \varphi \mathcal{U} \psi \) ("\( \varphi \) until \( \psi \)") and \( \circ \varphi \) ("next \( \varphi \)"), we introduce two temporal operators as abbreviations: \( \Diamond \varphi \equiv \text{true} \mathcal{U} \varphi \) ("eventually \( \varphi \)"), \( \square \varphi \equiv \neg(\Diamond(\neg \varphi)) \) ("always \( \varphi \)").

The temporal operators \( \mathcal{U}, \circ, \Diamond, \square \) have the highest binding power, followed by (in decreasing order) \( \neg, \land, \lor, \Rightarrow, \leftrightarrow \).

3.2 Semantics

The semantics of a PLTL formula is given by its interpretation over an infinite sequence of sets of propositions.

**Notation.** Given a sequence \( P = P_0P_1 \ldots \) and \( k \in \mathbb{N} \), we denote by \( P^k \) the sequence \( P \) without its first \( k \) elements, i.e., \( P_kP_{k+1} \ldots \).

**Definition 10 (PLTL semantics).** Given an infinite sequence of sets of propositions \( P = P_0P_1 \ldots \), the validity of a PLTL formula \( \varphi \) over \( P \), denoted as \( P \models \varphi \), is defined inductively over the structure of PLTL formulae as follows:

\[
\begin{align*}
P \models p & \quad \text{if and only if} \quad p \in P_0 \\
P \models \varphi \land \psi & \quad \text{if and only if} \quad P \models \varphi \quad \text{and} \quad P \models \psi \\
P \models \neg \varphi & \quad \text{if and only if} \quad \neg P \models \varphi \\
P \models \varphi \mathcal{U} \psi & \quad \text{if and only if} \quad \exists k \in \mathbb{N} : P_k \models \psi \quad \text{and} \\
& \quad \forall 0 \leq i < k : P^i \models \varphi \\
P \models \circ \varphi & \quad \text{if and only if} \quad P^1 \models \varphi
\end{align*}
\]

**Lemma 3 (PLTL semantics for \( \Diamond \) and \( \square \)).** By Def. 10, we have the following semantics for the \( \Diamond \) and \( \square \) operators:

\[
\begin{align*}
P \models \Diamond \varphi & \quad \text{if and only if} \quad \exists k \in \mathbb{N} : P_k \models \varphi \\
P \models \square \varphi & \quad \text{if and only if} \quad \forall k \in \mathbb{N} : P_k \models \varphi
\end{align*}
\]
3.3 PLTL for CLHA

We want to use PLTL formulae to describe the behavior of CLHA. Since the PLTL semantics operates on sequences of sets of propositions, we need to transform the semantical concept for CLHA (traces as introduced in Def. 6) into such sequences. This transformation depends on the kind of information we want to describe in the PLTL formulae. E.g., if we want to specify the changes of synchronization symbols along the run

\[ \sigma_0 \xrightarrow{l_1} \sigma_1 \xrightarrow{l_2} \sigma_2 \xrightarrow{l_3} \ldots \]

by a PLTL formula \( \varphi \), we can use the set of synchronization symbols \( L \) as the set of propositions and use the sequence of synchronization symbol sets

\[ l_{i_1} l_{i_2} l_{i_3} \ldots \]

(with \( i_{1} i_{2} i_{3} \ldots \) being the ordered sequence of indices \( i \) with \( l_i \subseteq L \), thus leaving out all continuous computation steps) to check the validity of \( \varphi \).

If we also need to reason about variable evaluations in PLTL formulae, it is necessary to choose abstractions of sets of evaluations like “\( x > 0 \)” as propositions. If each single variable evaluation was encoded as a proposition, many infinite sets of evaluations (like “\( x > 0 \)” with \( \text{type}(x) = \mathbb{R} \)) could not be expressed in a PLTL formula, since one formula can only contain finitely many propositions.

In the following we assume that we have a function \( pseq \) which transforms a CLHA run into a sequence of sets of propositions.

**Definition 11 (PLTL for CLHA).** Given a CLHA \( A \) and a PLTL formula \( \varphi \), the validity of \( \varphi \) over \( A \), denoted as \( A \models \varphi \), is defined as

\[ A \models \varphi \text{ if and only if } \forall t \in \llbracket A \rrbracket : pseq(t) \models \varphi . \]

Using the CLHA formalism presented above, components of a linear hybrid system can be specified in a unified framework. To formally verify a component, it can be translated into the input language of a model-checking tool that fits best to the features used in the component, e.g., SMV if the component only uses discrete variables, and KRONOS or UPPAAL if clocks are involved.

4 Example

We illustrate the CLHA formalism by modeling one of the product storage tanks in a chemical batch plant [11]. This tank stores a liquid substance and can be filled and drained via inlet and outlet pipes. The maximal capacity of the tank is 3.0 volume units. Filling the tank increases the volume of its content linearly by 0.05 volume units per second, while draining the tank changes the volume linearly by \(-0.10\) volume units per second. These numbers are based on measurements on the actual plant.
Our CLHA model uses five discrete locations, \textit{idle} (meaning that no in- or outflow occurs), \textit{fill} (filling the tank), \textit{drain} (draining the tank), \textit{both} (filling and draining happens simultaneously), and \textit{error} (after an error has occurred). Four self-explanatory input symbols are used to change the location: \textit{start\_fill}, \textit{stop\_fill}, \textit{start\_drain}, and \textit{stop\_drain}. One continuous variable \textit{vol} contains the current liquid volume of the tank.

There are two possibilities for the occurrence of an error: The tank overflows if \( \text{vol} \geq 3.0 \), and if \( \text{vol} \leq 0.0 \), the tank has run empty, which we consider as an “underflow” error. If an error occurs, the output symbol \textit{over}, resp., \textit{under} is sent, and the location \textit{error} is entered.

Figure 1 illustrates the CLHA model of the tank. The locations are marked with the flow rates of \textit{vol}, and locations with nonzero rates are labeled with invariants (\( \text{vol} > 0.0 \) and \( \text{vol} < 3.0 \)) to force the discrete transition to the \textit{error} location if \textit{vol} leaves its allowed range.

![CLHA tank model diagram](image)

**Fig. 1.** CLHA tank model

The system uses two real-valued variables (the variable \( t \) will be used in a second CLHA introduced later):

\[
V = \{ \text{vol}, t \}, \text{type(}\text{vol}\text{)} = \text{type(}t\text{)} = \mathbb{R} .
\]

The model of the tank is given as the CLHA

\[
T = (Q, \{ \text{idle} \}, \{ \text{vol} \}, \Sigma_0, R, I, L, E) ,
\]

where
- $Q = \{idle, fill, drain, both, error\}$,
- $\Sigma_0 = \{\sigma \in \Sigma | \sigma(vol) = 3.0\}$,
- $R((q, vol)) = \begin{cases} 
0.00, & \text{if } q \in \{idle, error\} \\
0.05, & \text{if } q = fill \\
-0.10, & \text{if } q = drain \\
-0.05, & \text{if } q = both
\end{cases}$, \quad \Sigma_{c\text{out}}, \text{ if } q \in \{idle, error\}$
- $I(q) = \begin{cases} 
\{\sigma \in \Sigma_{c\text{out}} | \sigma(vol) < 3.0\}, & \text{if } q = fill \\
\{\sigma \in \Sigma_{c\text{out}} | \sigma(vol) > 0.0\}, & \text{if } q \in \{\text{drain, both}\}
\end{cases}$, and
- $L = L^\text{in} \cup L^\text{out}$, with $L^\text{in} = \{\text{start\_fill, stop\_fill, start\_drain, stop\_drain}\}$ and $L^\text{out} = \{\text{over, under}\}$.

The set of edges is defined as follows: $E = E_{\text{idle}} \cup E_{\text{fill}} \cup E_{\text{drain}} \cup E_{\text{both}} \cup E_{\text{error}} \cup E_{\text{under}} \cup E_{\text{over}}$, where for all $l \subseteq L$, $\rho = (\sigma, \sigma_{\text{out}})$ with $\sigma \in \Sigma$, and $q, q' \in Q$,

- $(idle, l, \rho, q') \in E_{\text{idle}}$ if and only if $l \subseteq L^\text{in}$ and one of the following cases holds:
  - $\text{start\_fill} \notin l \land \text{start\_drain} \notin l \land q' = idle$
  - $\text{start\_fill} \in l \land \text{start\_drain} \in l \land q' = fill$
  - $\text{start\_fill} \notin l \land \text{start\_drain} \in l \land q' = drain$
  - $\text{start\_fill} \in l \land \text{start\_drain} \notin l \land q' = both$

- $(\text{fill}, l, \rho, q') \in E_{\text{fill}}$ if and only if $l \subseteq L^\text{in}$ and one of the following cases holds:
  - $\text{stop\_fill} \in l \land \text{start\_drain} \notin l \land q' = idle$
  - $\text{stop\_fill} \notin l \land \text{start\_drain} \notin l \land q' = fill$
  - $\text{stop\_fill} \in l \land \text{start\_drain} \in l \land q' = drain$
  - $\text{stop\_fill} \notin l \land \text{start\_drain} \in l \land q' = both$

- $(\text{drain}, l, \rho, q') \in E_{\text{drain}}$ if and only if $l \subseteq L^\text{in}$ and one of the following cases holds:
  - $\text{stop\_drain} \in l \land \text{start\_fill} \notin l \land q' = idle$
  - $\text{stop\_drain} \in l \land \text{start\_fill} \in l \land q' = fill$
  - $\text{stop\_drain} \in l \land \text{start\_fill} \in l \land q' = drain$
  - $\text{stop\_drain} \notin l \land \text{start\_fill} \notin l \land q' = both$

- $(\text{both}, l, \rho, q') \in E_{\text{both}}$ if and only if $l \subseteq L^\text{in}$ and one of the following cases holds:
  - $\text{stop\_drain} \in l \land \text{stop\_fill} \in l \land q' = idle$
  - $\text{stop\_drain} \in l \land \text{stop\_fill} \notin l \land q' = fill$
  - $\text{stop\_drain} \notin l \land \text{stop\_fill} \in l \land q' = drain$
  - $\text{stop\_drain} \notin l \land \text{stop\_fill} \notin l \land q' = both$

- $(\text{error}, l, \rho, error) \in E_{\text{error}}$

- $(q, l, \rho, error) \in E_{\text{under}}$ if and only if $q \in \{\text{drain, both}\}$, under $\in l$, over $\notin l$, and $\sigma(vol) < 0.0$

- $(\text{fill}, l, \rho, error) \in E_{\text{over}}$ if and only if over $\in l$, under $\notin l$, and $\sigma(vol) > 3.0$

Now that we have a complete CLHA model of the tank, we add a system to the tank’s environment which controls the refilling of the tank in case it impedes to run empty. The refilling process always adds a fixed amount of 1.0 units to the
tank. This controller is modeled as follows: whenever the volume of the tank’s content falls below $1.0$ units, $start\_fill$ is sent, and a clock $t$ is started. After 20 seconds ($t = 20$), $stop\_fill$ is sent. Figure 2 shows the CLHA model of the controller.

$$\begin{align*}
\text{wait} & \quad \text{vol} \leq 1.0 \\
\text{start\_fill}, t := 0 & \quad t = 20 \\
\text{refill} & \quad t < 20 \\
\text{stop\_fill} & \quad 0.0
\end{align*}$$

**Fig. 2.** CLHA controller model

The model of the controller is given as the CLHA

$$C = (Q^C, \{\text{wait}\}, \{t\}, \Sigma_0^C, R^C, I^C, L^C, E^C),$$

where

- $Q^C = \{\text{wait, refill}\}$,
- $\Sigma_0^C = \{\sigma \in \Sigma | \sigma(t) = 0.0\}$,
- $R^C(\text{wait}, t) = 0.0, R^C(\text{refill}, t) = 1.0$,
- $I^C(\text{wait}) = \Sigma^{\text{cout}}, I^C(\text{refill}) = \{\sigma \in \Sigma^{\text{cout}} | \sigma(t) < 20\}$, and
- $L^C = L^\text{in}_C \cup L^\text{out}_C$, with $L^\text{in}_C = \emptyset$ and $L^\text{out}_C = \{\text{start\_fill, stop\_fill}\}$.

The set of edges is defined as follows: for all $l \subseteq L^C$, $\rho = (\sigma, \sigma')$ with $\sigma \in \Sigma$, $\sigma' \in \Sigma^{\text{cout}}$, and $q, q' \in Q^C$, we have $(q, l, \rho, q') \in E$ if and only if one of the following cases holds:

- $q = q' = \text{wait} \land l = \emptyset \land \sigma(\text{vol} > 1.0) \land \sigma'(t) = \sigma(t)$
- $q = \text{wait} \land q' = \text{refill} \land l = \emptyset \land \sigma(\text{vol} \leq 1.0) \land \sigma'(t) = 0$
- $q = q' = \text{refill} \land l = \emptyset \land \sigma'(t) = \sigma(t)$
- $q = \text{refill} \land q' = \text{wait} \land l = \emptyset \land \sigma(\text{vol}) = \sigma'(t)$

The product automaton $T \parallel C$ is shown in Fig. 3. Product locations which are obviously not reachable from the initial location, e.g., (idle, refill) are not shown, as well as transitions which are never enabled. For readability we also left out the rates for vol and $t$, and all occurences of $start\_fill$ and $stop\_fill$.

Figure 4 shows how vol evolves over time in a scenario where the tank is drained at time 2 ($start\_drain$ is received) for a duration of 23 seconds. After 20 seconds of draining (i.e., at time 22), vol falls below 1.0 units, and the controller sends $start\_fill$, which slows down the draining to $-0.05$ units per second. 3 seconds later, the draining stops ($stop\_drain$), and the level in the tank rises. After 20 seconds (at time 42) the filling is stopped ($stop\_fill$), and vol has reached 17.0 units.
![CLHA model of T∥C]

**Fig. 3.** CLHA model of $T \parallel C$

![A run of the system $T \parallel C$]

**Fig. 4.** A run of the system $T \parallel C$
References