Reachability Analysis of Hybrid Automata with Clocked Linear Dynamics

Viktorio S. el Hakim
v.s.elhakim@utwente.nl
Computer Architectures for Embedded Systems
University of Twente
Enschede, The Netherlands

Marco J. G. Bekooij
marco.bekooij@nxp.com
Algorithms and Software Innovation
NXP Semiconductors
Eindhoven, The Netherlands

Abstract

Disributed control systems often exhibit aperiodic sampling behavior due to varying communication delays and execution times. In such cases traditional analysis methods fall short because the functional and temporal behaviors need to be analyzed simultaneously. Therefore such systems are often modeled by Hybrid Automata (HA) with clock and non-clock variables, and verified using reachability analysis. However, modern reachability tools introduce a large over-approximation error because non-clock variables, as well as clock variables, are equally treated by the algorithm.

In this paper we present a reachability algorithm which exploits the explicit separation of clock and non-clock variables in the Hybrid Automata with Clocked Linear Dynamics (HA-CLD) subclass, as well as restricting that guard and invariant constraints can only be specified in the HA-CLD model for clock variables. These properties of HA-CLD allow independent computation of tight reachable set over-approximations, in the form of flow-pipes, of clock and non-clock variables. A computationaly efficient and tight intersection operation is obtained by relating segments of the clock flow-pipe with segments of the non-clock flow-pipe.

We demonstrate the effectiveness of our approach using two benchmarks, in the context of verifying stability. From the results it can be concluded that our reachability approach obtains significantly tighter results with an up-to 65 times smaller run-time compared to the start-of-the-art model checker SpaceEx.

CCS Concepts

• **General and reference** → **Verification**; • **Computing methodologies** → **Model verification and validation**; • **Hardware** → **Model checking**; • **Software and its engineering** → **Model checking**; *Software verification*; *Software safety*; *Software verification and validation*; • **Computer systems organization** → *Embedded and cyber-physical systems*.

Keywords

hybrid automata, reachability analysis

ACM Reference Format:
Viktorio S. el Hakim and Marco J. G. Bekooij. 2019. Reachability Analysis of Hybrid Automata with Clocked Linear Dynamics. In *22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES '19), May 27–28, 2019, Sankt Goar, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3323439.3323980

SCOPES '19, May 27–28, 2019, Sankt Goar, Germany

ACM ISBN 978-1-4503-6762-2/19/05…$15.00
https://doi.org/10.1145/3323439.3323980

1 Introduction

Reachability analysis of HA is an important and extensively studied problem in the hybrid systems community. The *reachability* problem is to compute the set of all the possible states that a system can reach from a given initial set, as the system evolves over time. Equivalently, it is a problem of exhaustively evaluating all possible state trajectories. This reachable set is then used to verify certain safety and liveness properties.

However computing the reachable set exactly is usually not possible and as a result it is typically derived as a set of over-approximated sub-sets, a process known as flow-pipe construction. The most common representations of the sub-sets are convex polytopes, zonotopes, ellipsoids and oriented box hulls, etc [3, 5–7, 9, 12, 13, 18]. A problem with this method of computing the reachable set is the accumulation of error due to over-approximation, also known as the wrapping effect [16]. Typical causes for this are 1) mixture of different types of variables, for which one representation may not be universally tight; 2) set operations that produce one representation from another, e.g. intersection of zonotopes; 3) and over-approximations introduced to reduce the complexity of a set representation, and reducing the number of sub-sets (clustering). While there exist methods to avoid over-approximation when performing Continuous-Time (CT) and discrete-event reachability independently for certain sub-classes of the HA, it is unavoidable when combined.

In this paper we consider Hybrid Automata with Clocked Linear Dynamics [10], and present an efficient approach for their reachability analysis. In these models the CT state-space is partitioned into two types: clock and non-clock variables. Clock variables (clocks) have simple linear dynamics in the form of $\dot{c} = 1$, while non-clock dynamics are specified by linear

Ordinary Differential Equations (ODEs). This separation of variables allows applying type-specific reachability techniques on lower-dimensional subspaces, which increases the accuracy and efficiency. Additionally, the invariants and guards are only defined for clocks, which greatly simplifies computing intersections.

In [24, 25] the decomposition of general HA models is considered, where the entire state-space is partitioned based on the complexity of the continuous dynamics. However, a key difference with the work presented in this paper is that these works do not introduce an efficient intersection technique.

We present the results of two benchmarks for control systems with sampling jitter. In the first benchmark sampled sensor data received by the controller is only delayed, whereas in the second benchmark the data might be lost. The results for these benchmarks obtained with our model checker are compared with results obtained with the state-of-the-art model checker SpaceEx [12].

The rest of this paper is organized as follows. In Section 2 we review related work. Section 3 discusses the basic idea of our approach and introduces common reachability analysis issues. Section 4 defines the semantics of HA-CLD model. In Section 5 we describe the flow-pipe construction process. In Section 6 we present the complete reachability algorithm and the key techniques that simplifies the intersection operation. Our benchmark results are presented in Section 7. Our conclusions are presented in Section 8.

2 Related Work

In this section we discuss approaches that are closely related to our work and outline the differences.

The works by Frehse and Le Guernic et al. [13, 18] present a reachability analysis approach for Linear Time Invariant (LTI) systems with extensions to Hybrid Automata with Linear Dynamics (HA-LD). Here they utilize symbolic support function representations of the over-approximated sets, allowing certain operations to be applied efficiently. The methods presented in these works have been successfully integrated into the SpaceEx model-checker [12]. However SpaceEx does not exploit separability of clock and non-clock variables for HA-CLD models to improve the accuracy and computational efficiency. In contrast our approach specifically targets HA-CLD models.

An approach by Schupp et al. [24, 25], similarly to our work, proposes a reachability algorithm (implemented using HyPro [22]) that utilizes syntactical separation of the variables into partitions with dynamic-specific classes, namely (1) zero-derivative, (2) timed (clocks), (3) constant-derivative and (4) linear. Existing class-specific flow-pipe techniques are then applied to each partition separately to achieve better efficiency and accuracy. They also observe that segments in different partitions are related and note that if a guard or invariant, i.e. a predicate, for a segment of one flow-pipe does not hold, then there is no need to evaluate the predicates of the other flow pipes, which improves the efficiency. A difference with our work is that in the HA-CLD model guards and invariants can only be defined for clocks, and checking these predicates on clocks is always easy. Furthermore, Schupp does not present an efficient way to compute over-approximated intersections. We present an efficient intersection approach which is based on the relation between the segments of clock and non-clock flowpipes.

A recent work by Bogomolov et al. [6] describes an approach which considers decomposing a highly dimensional system into 2×2 sub-systems that can be independently analyzed more efficiently and accurately. A cartesian product of the resulting reachable sub-sets is then computed, which over-approximates the reachable set of the original system. However, the approach does not make an explicit type distinction between the sub-spaces. As such, efficient guard/invariant intersection and flow-pipe construction approaches for each sub-space are not considered. A reachability algorithm is also not presented.

The works by Khatib et al. [1, 2] present a stability verification approach using reachability analysis of systems, where the temporal and functional behaviors are explicitly specified. The models considered are very similar to our HA-CLD, and the algorithms presented exploit the separability of temporal and functional variables to compute reachable sets efficiently and synthesize schedules. However, an important difference is that their approach supports only a single clock and one discrete mode, whereas our approach supports models with multiple clocks and multiple modes.

3 Basic Idea

In this section we describe common issues associated with reachability analysis and the basic idea of our approach.

3.1 Reachability of HA-LD

HA-LD are automata, of which HA-CLD is a sub-class, where the CT state variable $x \in \mathcal{X}$ in a mode $q \in Q$ evolves according to a system of first order linear ODEs. Each mode has a polyhedral set of constraints (invariants), $\text{Inv}(q)$, which are defined using systems of linear inequalities over the continuous state-space $\mathcal{X}$. The discrete transitions (edges) from each mode, $e \in E$, are equipped with linear reset maps and polyhedral constraints (guards), $G(e)$, which are also defined as systems of linear inequalities over $\mathcal{X}$

The reachability algorithm for HA-LD and HA-CLD alike is summarized in the following steps:

(1) CT reachability: for each $q \in Q$ an initial set X_k^q is intersected with the respective invariant $\text{Inv}(q)$, and a flow-pipe over-approximation is computed as a set of segments that satisfy the invariant.
(2) Discrete-Event (DE) reachability: for each edge $e \in E$ the computed flow-pipes are intersected with a respective guard $G(e)$, and a reset map is applied.
(3) Clustering and aggregation: the computed sets are clustered and aggregated in each mode, and used as initial sets in the next iteration.
(4) Steps 1-3 are repeated in subsequent iterations, until a fixed-point condition is satisfied.

3.2 Common issues

Traditionally, the set over-approximations are represented using geometrical or numerical objects, which have certain advantages and disadvantages over each other with respect to the computational effort and over-approximation error introduced by set operations. For example linear transformations, Minkowski sums and testing intersection are computationally efficient operations for zonotopes, but computing the intersection itself is not [11, 15]. Then these representations change over the course of the algorithm into further over-approximations during steps 2 and 3, and as a result amplify the error. Furthermore the error scales with the dimensionality of the system and certain representations become less tight when variables with simple dynamics are mixed into the state-space, such as clocks and Piece-Wise Constant (PWC) variables [23, 24]. Finally, when simple guard and invariant constraints are defined for e.g. clocks, an intersection and inclusion operation can be more expensive than needed, as an unnecessary complex representation also suitable for non-clock variables is used.

As an example consider a distributed cyber-physical controller implemented on a multi-processor architecture with a shared memory. Due varying execution times, the controller may be affected by sampling jitter which deteriorates its performance. To verify whether the system still functions correctly within some safety and performance margin one may derive a HA-LD model, given that the bounds on the execution time of the controller are known. This automaton consists of a single mode wherein the dynamics of the plant are described by a state-space representation, while the control law is specified by a transition with a reset map. Additionally, a clock is introduced to model the execution time of the controller. The invariant and guard are intervals over the clock variable, which specify an upper and a lower bound on the execution time. Thus the plant, controller and clock variables, $\boldsymbol{x}(t), \boldsymbol{u}(t)$ and $c(t)$, respectively, form the state-space of the automaton. In this case it is obvious that an explicit intersection of the reachable sets is not necessary, because the time duration in the given mode can be exactly derived and the clock flow-pipe can be trivially computed. However, modern reachability algorithms do not usually make this type distinction and equally apply a common technique to all variables. Intersections are also then applied explicitly on the whole state-space, which is computationally intensive and may lead to large over-approximations of the reachable set. A consequence is that the analysis may conclude that the system does not satisfy the requirements, even though in reality it does.

In the spirit of this example we take into consideration that separating clock and non-clock variables can have great benefits for the reachability analysis, and exploit this concept in our approach. For example the flow-pipe of the non-clock variables is more tightly computed since the dimensionality is reduced, because clock variables are considered seperately.

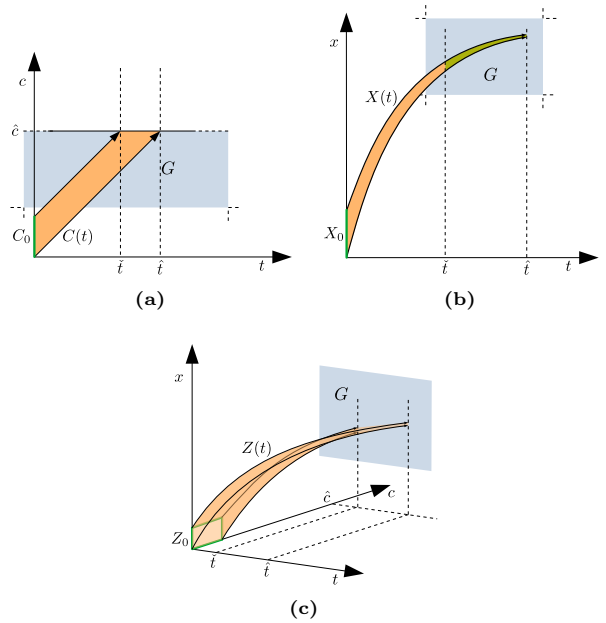


Figure 1: Disjoint flow-pipes $C(t)$ and $X(t)$ (1a and 1b respectively, and the combined flow-pipe $Z(t)$ (1c), intersected by a guard G.

3.3 Difficulties with intersections in the clock domain

Separation of variables introduce a challenge when the guard and invariant intersections with the flow-pipes need to be computed exactly. Typically, they are over-approximated instead. To show why computing exact flow-pipe intersections is difficult when the clock and non-clock domains are disjoint, consider a simple automaton with variables $c, x \in \mathbb{R}$, where c is a clock. Also consider a guard G defined only for the clocks, which enforces the constraint that $c \leq \hat{c}$. Assume that the sets C_0 and X_0 are initial state sets for c and x, respectively, which continuously evolve until a trajectory for a given initial state c_0 reaches the guard's boundary. When treated separately, the flow-pipes $C(t)$ and $X(t)$ are independently evolving from C_0 and X_0, respectively, and contain all of the possible trajectories. The flow-pipes are shown in Figures 1a and 1b, respectively, where eventually the upper and lower bound trajectories of $C(t)$, shown as arrows, intersect with the boundary of G at times $\check{t}$ and $\hat{t}$, respectively. Notice that for a trajectory $x(t) \in X(t)$ the time of exact intersection, $t' \in [\check{t}, \hat{t}]$, is not captured within the interval $[\check{t}, \hat{t}]$. Furthermore, there is no explicit relation between individual trajectories in $c(t) \in C(t)$ and the corresponding trajectories $x(t) \in X(t)$. The only information that is derived is the upper and lower bound of the interval $[\check{t}, \hat{t}]$. Because the relation between trajectories is not derived, all possible states reached within the time interval $[\check{t}, \hat{t}]$ are considered as end points for each trajectory in $X(t)$, and as result all are considered as the potential intersection boundary of $X(t)$. This is handled

as non-determinism and the flow-pipe segment within $[\check{t}, \hat{t}]$ is included in the reachable set, which is an over-approximation.

Now consider the case where the variables are not treated independently and a combined flow-pipe is computed instead. Specifically the set of trajectories for every tuple $(c, x) \in Z_0 \subseteq \mathbb{R}^2$ is considered, where $Z_0 = C_0 \times X_0$. The corresponding flow-pipe is thus $Z(t)$, and is shown in Figure 1c. Notice here that the guard G is extended to cover the domain of variable x, and is thus a half-space described by the set $\{(c, x) \in \mathbb{R}^2 \mid c \leq \hat{c}\}$. In this case the guard exactly intersects with the flow-pipe, and a time instance of intersection for the each individual trajectory can be determined.

Further in the paper we describe our solution for the computation of precise, but not exact, intersections when clock and non-clock flowpipes are computed independently.

4 Semantics of Hybrid Automata with Clocked Linear Dynamics

In this section we present the formal definition of HA-CLD and discuss its semantics.

4.1 Notation

Throughout the paper vectors and matrices are displayed with bold letters, e.g. $\boldsymbol{x} = (x_1, ..., x_n)^\intercal \in \mathbb{R}^n$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. The matrix $\boldsymbol{I}_n \in \mathbb{R}^n$ is the identity matrix with columns $\boldsymbol{e}_i, i = 1, ..., n$, where $\boldsymbol{e}_i$ is a standard basis vector with the i-th component equal to 1 and the rest 0. Multiplication of a matrix $\boldsymbol{A}$ with a set $X \subseteq \mathbb{R}^n$ is the set $\boldsymbol{A}X = \{\boldsymbol{x} \in X \mid \boldsymbol{A}\boldsymbol{x}\}$. Adding a vector $\boldsymbol{b} \in \mathbb{R}^n$ to a set X is simply $X + \boldsymbol{b} = \{\boldsymbol{x} + \boldsymbol{b} \mid \boldsymbol{x} \in X\}$, while adding a set $Y \subseteq \mathbb{R}^n$ to X, an operation called a Minkowski sum, is $X + Y = \{\boldsymbol{x} + \boldsymbol{y} \mid \boldsymbol{x} \in X \text{ and } \boldsymbol{y} \in Y\}$. If a set X is finite, then $|X|$ is the number of its elements. For a matrix $\boldsymbol{A}$ with entries a_{ij}, $|\boldsymbol{A}|$ is a matrix with entries $|a_{ij}|$. A unit ball is the set $\mathcal{B}_p = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_p \leq 1\}$, where $\|\cdot\|_p$ is any vector p-norm. The 2-norm is assumed throughout the paper and the subscript is omitted unless otherwise specified. Given a set X, then the set $\text{Conv}(X)$ is its convex hull. Given vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, then $\boldsymbol{x} \diamond \boldsymbol{y} \triangleq x_1 \diamond y_1 \wedge \cdots \wedge x_n \diamond y_n, \diamond \in \{<, \leq, =, \geq, >\}$, $\min\{\boldsymbol{x}, \boldsymbol{y}, ...\} \triangleq (\min\{x_1, y_1, ...\}, ..., \min\{x_n, y_n, ...\})^\intercal$. Given an $\boldsymbol{x}$ and $\boldsymbol{y} \geq \boldsymbol{0}$, then a box (interval hull) is the set $\text{Box}(\boldsymbol{x}, \boldsymbol{y}) = [x_1, x_1 + y_1] \times \cdots \times [x_n, x_n + y_n]$. Let $f : \mathbb{R}^{i_1} \times \cdots \times \mathbb{R}^{i_N} \to \mathbb{R}^{o_1} \times \cdots \times \mathbb{R}^{o_M}$ be a function, then given the sets $X_1 \subseteq \mathbb{R}^{i_1}, ..., X_N \subseteq \mathbb{R}^{i_N}$ and applying f to each of them results in the set $f(X_1, ..., X_N) = \{f(\boldsymbol{x}_1, ..., \boldsymbol{x}_N) \mid (\boldsymbol{x}_1, ..., \boldsymbol{x}_N) \in X_1 \times \cdots \times X_N\}$.

4.2 Definiton

We use a similar definition as the one provided in [19]:

Definition 4.1. A Hybrid Automata with Clocked Linear Dynamics is a tuple $\mathcal{T} = (Q, \mathcal{X}, \mathcal{C}, f, \text{Init}, \text{Inv}, E, G, R_x, R_c, \mathcal{U})$ where

- $Q = \{q_1, ..., q_N\}$ is a finite set of discrete states (modes);
- $\mathcal{X} = \{x_1, ..., x_n\}$ is the set of CT state variables;
- $\mathcal{C} = \{c_1, ..., c_p\}$ is the set of clock variables;
- $f : Q \times \mathcal{X} \to \mathcal{X}$ is an affine flow assignment map;
- $\text{Init} \subseteq Q \times \mathcal{X} \times \mathcal{C}$ is the set of initial states;
- $\text{Inv} : Q \to 2^{\mathcal{C}}$ is a clock invariant assignment map;
- $E \subseteq Q \times Q$ is a set of edges;
- $G : E \to 2^{\mathcal{C}}$ is a transition guard assignment map;
- $R_x : E \times \mathcal{X} \to 2^{\mathcal{X}}$ is a state reset assignment map;
- $R_c : E \times \mathcal{C} \to 2^{\mathcal{C}}$ is a clock reset assignment map;
- $\mathcal{U} : Q \to \mathbb{R}^m$ is a bounded input set assignment map;

The map f assigns an affine function for each mode $q \in Q$, such that the continuous state $\boldsymbol{x} \in \mathcal{X}$ evolves over time according to:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_q \boldsymbol{x}(t) + \boldsymbol{B}_q \boldsymbol{u}(t);\ \boldsymbol{u}(t) \in \mathcal{U}(q), \tag{1}$$

where $\boldsymbol{A}_q \in \mathbb{R}^{n \times n}, \boldsymbol{B}_q \in \mathbb{R}^{n \times m}$ and $\boldsymbol{u}(t) \in \mathbb{R}^m$ is a bounded input signal, i.e. $\mathcal{U}(q) \subseteq \mu_q \mathcal{B} + \boldsymbol{u}_q, \mu_q \in \mathbb{R}_+$. The clocks evolve in all modes according to $\dot{\boldsymbol{c}}(t) = \boldsymbol{1}$, where $\boldsymbol{1} = (1, ..., 1)^\intercal \in \mathbb{R}^p$. State evolution is only allowed as long as the clocks satisfy the mode's invariant, i.e. $\boldsymbol{c}(t) \in \text{Inv}(q)$.

For a mode q the clock invariant is an interval hull of the form $\text{Inv}(q) = \text{Box}(\check{\boldsymbol{c}}_q, \hat{\boldsymbol{c}}_q - \check{\boldsymbol{c}}_q)$. Similarly for an edge $e \in E$ the transition guard is an interval hull of the form $G(e) = \text{Box}(\check{\boldsymbol{c}}_e, \hat{\boldsymbol{c}}_e - \check{\boldsymbol{c}}_e)$.

The reset maps R_x and R_c assign affine reset functions for each edge $e = (q, q') \in E$, which are triggered whenever the transition $q \to q'$ occurs at time t'. Assuming a guard is enabled for an edge e at time t, i.e. $\boldsymbol{c}(t) \in G(e)$, then a reset assigns new values to the clock and non-clock variables at t', such that $\lim_{t' \to t^+} \boldsymbol{x}(t') = \boldsymbol{G}_e \boldsymbol{x}(t)$ and $\lim_{t' \to t^+} \boldsymbol{c}(t') = \boldsymbol{C}_e \boldsymbol{c}(t) + \bar{\boldsymbol{c}}_e$.

4.3 Reachability

Let $\mathcal{T}$ be a HA-CLD as defined in Definition 4.1. Given a state $s = (q, \boldsymbol{x}, \boldsymbol{c}) \in Q \times \mathcal{X} \times \mathcal{C}$, then the forward continuous-time reachable set (flow) relation from s is defined as:

$$\text{Flow}(s) = \{(q, \xi_q(t, \boldsymbol{x}, \mathcal{U}(q)), \boldsymbol{c} + t\boldsymbol{1}) \mid \boldsymbol{c} + t\boldsymbol{1} \in \text{Inv}(q), t \in \mathbb{R}_+\}, \tag{2}$$

where ξ_q is a trajectory function that satisfies equation (1). A discrete transition (jump) relation from a state $s \in \text{Inv}(q)$ is defined as

$$\text{Jump}(s) = \{(q', \boldsymbol{G}_e \boldsymbol{x}, \boldsymbol{C}_e \boldsymbol{c} + \bar{\boldsymbol{c}}_e) \mid \exists e = (q, q') \in E : \boldsymbol{c} \in G(e)\}. \tag{3}$$

Definition 4.2. Let $S_k = \text{Jump}(\text{Flow}(S_{k-1}))$, $k \in \mathbb{N} \setminus \{0\}$, with $S_0 = \text{Init}$, then an execution of the hybrid automaton $\mathcal{T}$ is the sequence $\{s_k\}_{k=0}^{\infty}$, such that $\forall k \in \mathbb{N} : s_k \in S_k$ and t_k is a time moment when a discrete transition $q_{k-1} \to q_k$ takes place.

The reachable set of states after k iterations is defined as

$$\mathcal{R}_k = \bigcup_{j=0}^{k} S_j = S_k \cup \mathcal{R}_{k-1}. \tag{4}$$

The reachability problem is to compute $\mathcal{R}_k$ for some finite k. Traditionally $\mathcal{R}_k$ is iteratively computed until the condition $S_k \subseteq \mathcal{R}_{k-1}$ is satisfied for some k, or a maximum number of iterations, $\hat{k}$, is reached.

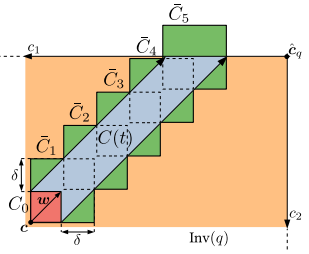
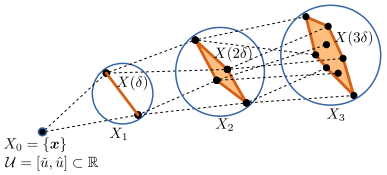


Figure 2: A clock flow-pipe $C(t)$ (light blue), and it's box over-approximation $[C_0, 5]^{\uparrow}$ (dark green) over an invariant $\text{Inv}(q)$.



(a)

Figure 3: Exact flow-pipe successors $X(i\delta)$ (orange) and the corresponding approximations X_1, X_2, X_3 (blue)

5 Continuous time reachability

In this section we describe in detail the computation of $\text{Flow}(S)$ using flow-pipe over-approximations from a single initial set $S = (q, X_0, C_0)$.

5.1 Clock flow-pipes

Clock variables in HA-CLD have simple dynamics, and thus computing the flow-pipe is easier compared to the non-clock variables. For every mode $q \in Q$ given an initial clock valuation $\boldsymbol{c}_0$, the new value at time $t \geq 0$ is symbolically represented as $\boldsymbol{c}(t) = \boldsymbol{c}_0 + t\mathbf{1}$. If one has an initial set C_0 instead, then the unconstrained flow-pipe representation is $C(t) = C_0 + t\mathbf{1}$. However, as discussed previously, this representation is not suitable further on when guard and invariant intersections need to be computed. Additionally, as it will be shown further in this section, a non-clock flow-pipe cannot be derived exactly and is instead over-approximated by a union of segments, computed over fixed time-intervals. Thus, in order to retain a discrete-time relation between clock and non-clock flow-pipes, and to simplify their guard and invariant intersections, a similar quantization technique is also applied to the clock flow-pipe which we describe below.

5.1.1 Over-approximation

Before computing the flow-pipe, a tight time interval, $[0, T]$, and a time step, $\delta \in \mathbb{R}_+$, are selected in order to guarantee that $C(t)$ eventually reaches a boundary of the invariant $\text{Inv}(q)$ for some $t \in [0, T]$. Next, the flow-pipe is over-approximated by a set of $z \in \mathbb{N} \setminus \{0\}$ segments, $[C_0, z]^{\uparrow} = \{\bar{C}_1, .., \bar{C}_z\}$, such that the flow-pipe is completely covered by their union. We use box over-approximations for the segments, since they simplify guard/invariant intersections and require minimum storage. Suppose the initial set is specified as $C_0 = \text{Box}(\boldsymbol{c}, \boldsymbol{w})$, then a segment is:

$$\bar{C}_i = \text{Box}(\boldsymbol{c}_i, \boldsymbol{w} + \delta\mathbf{1}),\ i = 1, .., z \tag{5}$$

where $\boldsymbol{c}_i = \boldsymbol{c} + (i-1)\delta\mathbf{1}$. It is easy to show that $\forall t \in [0, T]$: $C(t) \subseteq \bar{C}_1 \cup \cdots \cup \bar{C}_z$. An example flow-pipe computation is shown in Figure 2, where $z = 5$. We observed that while this representation is over-approximative, the introduced error does not greatly affect the outcome of the reachability algorithm.

5.1.2 Computing z directly

Another observation is that selecting a T is unnecessary, and the exact amount of segments z can be derived directly given a time step δ, an initial set C_0, and an invariant $\text{Inv}(q)$, provided that at least $C_0 \cap \text{Inv}(q) \neq \emptyset$. Given these preconditions, z can be directly computed according to $z = \left\lceil \frac{\max_t\{\boldsymbol{c} \in C_0 \mid \boldsymbol{c}+t\mathbf{1} \in \text{Inv}(q)\}}{\delta} \right\rceil$. Given that $C_0 = \text{Box}(\boldsymbol{c}, \boldsymbol{w})$ and $\text{Inv}(q) = \text{Box}(\check{\boldsymbol{c}}_q, \hat{\boldsymbol{c}}_q - \check{\boldsymbol{c}}_q)$, then this simplifies to:

$$z = \left\lceil \frac{\min\{\hat{\boldsymbol{c}}_q - \boldsymbol{c} - \boldsymbol{w}\}}{\delta} \right\rceil, \tag{6}$$

where the minimum in this expression is taken over the components of a vector. This is possible because the clock variables increase linearly with time toward the upper boundaries of the invariant, and never toward the lower, see Figure 2.

5.2 Non-clock flowpipes

Consider the ODE as defined in eq. (1). For clarity we assume a single-mode automaton through-out this section and omit q, with $\mathcal{U} \triangleq \mathcal{U}(q)$, unless otherwise stated. At any time, $t \in \mathbb{R}_+$, the trajectory, $\xi : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, of $\boldsymbol{x}$ from an initial state, $\boldsymbol{x}_0$, and for a disturbance signal, $\boldsymbol{u}(\tau) \in \mathcal{U}, 0 \leq \tau < t$, is defined as:

$$\xi(t, \boldsymbol{x}_0, \boldsymbol{u}) = \Phi(t)\boldsymbol{x}_0 + \int_0^t \Phi(t-\tau)\boldsymbol{B}\boldsymbol{u}(\tau)d\tau, \tag{7}$$

where $\Phi(t) = e^{\boldsymbol{A}t}$ is the state-transition matrix. The unconstrained flow-pipe is then the set of all possible trajectories starting from every initial state for all possible disturbance signals, i.e. $X(t) = \xi(t, X_0, \mathcal{U})$. As mentioned earlier, it cannot be computed exactly and instead is over-approximated by a set of z polytopic segments, $[X_0, z]^{\uparrow} = \{\bar{X}_1, .., \bar{X}_z\}$, where z is derived by the method described earlier. A segment is the polytope, $\bar{X}_i = \text{Conv}(X_{i-1} \cup X_i) + \alpha_i P, i = 1, .., z$, where X_i is a discrete-time successor set, $P \subseteq \mathbb{R}^n$ is a polytope, and α_i is a "bloating" constant [14], which guarantees that the trajectories are completely contained by the union of the segments. Formally, $\forall t \in [0, z\delta] : \xi(t, X_0, \mathcal{U}) \subseteq \bar{X}_1 \cup \cdots \cup \bar{X}_z$. Note that we assume that X_0 is a polytope, and P is chosen such that $\mathcal{B}$ is tightly inscribed inside P, i.e. the minimum distance between each face of P and any point in $\mathcal{B}$ is 0. The computation of $[X_0, z]^{\uparrow}$ is described below.

5.2.1 Flow-pipe over-approximation

We first describe how equation (7) is used to compute each X_i. Since considering all possible input signals is intractable,

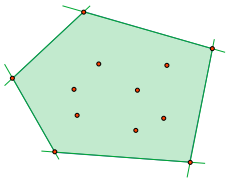


Figure 4: A set as an arrangement of hyperplanes and intersecting half-spaces (green), and as a finite number of points (orange).

it is reasonable to instead consider the subset of PWC signals, $U_c = \{\boldsymbol{u} \in \mathcal{U} \mid \forall \tau \in [0,\delta) : \boldsymbol{u}(\tau) = \boldsymbol{u}_{const} \in \mathbb{R}^m\}$. Then by evaluating the integral in equation (7):

$$X(i\delta) = \xi(\delta, X((i-1)\delta), U_c) = \Phi_\delta X((i-1)\delta) + \Gamma_\delta U_c, \quad (8)$$

where Φ_δ and Γ_δ are computed by noting that $e^{\begin{pmatrix} \boldsymbol{A}\delta & \boldsymbol{B}\delta \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}} = \begin{pmatrix} \Phi_\delta & \Gamma_\delta \\ \boldsymbol{0} & I_m \end{pmatrix}$ [8]. Next, we expand equation (8) and substitute $U_c = \boldsymbol{u}_c + \mu\mathcal{B}$ to derive:

$$X(i\delta) = \Phi_\delta^i X_0 + \sum_{j=0}^{i-1} \Phi_\delta^j \Gamma_\delta U_c = \\ = \tilde{X}_i + \mu \sum_{j=0}^{i-1} \Phi_\delta^j \Gamma_\delta \mathcal{B} \subseteq \tilde{X}_i + \mu\beta_i\mathcal{B} = X_i, \quad (9)$$

where $\tilde{X}_i = \Phi_\delta^i X_0 + \sum_{j=0}^{i-1} \Phi_\delta^j \Gamma_\delta \boldsymbol{u}_c$ and $\beta_i = \sum_{j=0}^{i-1} \|\Phi_\delta^j \Gamma_\delta\|$. In the last expression we make use of the following property: given some matrices $\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3, \ldots$, then $\boldsymbol{M}_1\mathcal{B} + \boldsymbol{M}_2\mathcal{B} + \boldsymbol{M}_3\mathcal{B} + \cdots \subseteq (\|\boldsymbol{M}_1\| + \|\boldsymbol{M}_2\| + \|\boldsymbol{M}_3\| + \cdots)\mathcal{B}$. An example of X_i is shown in Figure 3a.

Finally, using equation (9), properties of the Minkowski sum and the convex hull, a flow-pipe segment is:

$$\bar{X}_i = \mathrm{Conv}(X_{i-1} \cup X_i) + \alpha_i\mathcal{B} = \\ = \mathrm{Conv}(X_{i-1} + \alpha_i\mathcal{B} \cup X_i + \alpha_i\mathcal{B}) = \\ = \mathrm{Conv}(\tilde{X}_{i-1} + (\mu\beta_{i-1} + \alpha_i)\mathcal{B} \cup \tilde{X}_i + (\mu\beta_i + \alpha_i)\mathcal{B}). \quad (10)$$

Note that at this point the convex hull operation is purely symbolic and is not actually computed. We then proceed with replacing $\mathcal{B}$ by a polytope P, such that $\mathcal{B} \subseteq P$. Good candidates for P are the regular simplex, the 1-norm ball $\mathcal{B}_1$, and the ∞-norm ball $\mathcal{B}_\infty$. An upper bound on the bloating constant α_i is derived in [14].

5.2.2 *Set representation*

There are two possible representations for the non-clock sets: as sets of points or as an arrangement of hyperplanes (see Figure 4). In this paper we utilize the earlier representation, where an X is a finite set of points, i.e. $X = \{\boldsymbol{x}_1, .., \boldsymbol{x}_j \in \mathbb{R}^n\}$. The main advantages of this representation compared to the hyperplane representation are: 1) tighter over-approximation of the flow-pipe because flow-pipes of individual points are tighter ; 2) simplifies certain set operations such as unions and inclusions; 3) simplifies clustering and pruning; and 4) allows representing non-convex sets.

Algorithm 1 Reachability of a HA-CLD

Input: An automaton $\mathcal{T} = (Q, \mathcal{X}, \mathcal{C}, f, \mathrm{Init}, \mathrm{Inv}, E, G, R_x, R_c, \mathcal{U})$, $\hat{k} \in \mathbb{N} \setminus \{0\}$, a time step δ and an $S_0 = (q_0, X_0^{q_0}, C_0^{q_0}) \in \mathrm{Init}$.
Output: The sets $S_1, ..., S_k, k \leq \hat{k}$.

$Q_0 \leftarrow \{q_0\}$
for $k = 1, ..., \hat{k}$ **do**
  $Q_k \leftarrow Q_{k-1}$
  $\forall q \in Q : (X_k^q, C_k^q) \leftarrow (\emptyset, \emptyset)$ ▷ Flow-pipe computation
  for each $q \in Q_{k-1}$ **do**
    $\mathcal{F} \leftarrow \emptyset$
    for each $(X, C = \mathrm{Box}(\boldsymbol{c}, \boldsymbol{w})) \in (X_{k-1}^q, C_{k-1}^q)$ **do**
      if $C \cap \mathrm{Inv}(q) \neq \emptyset$ **then**
        $z = \left\lceil \frac{\min\{\hat{\boldsymbol{c}}_q - \boldsymbol{c} - \boldsymbol{w}\}}{\delta} \right\rceil$
        $\mathcal{F} \leftarrow \mathcal{F} \cup \{([X, z]^\uparrow, [C, z]^\uparrow)\}$
      end if
    end for
    for each $e := (q, q') \in E$ **do** ▷ Discrete reachability
      for each $(X, C = \mathrm{Box}(\boldsymbol{c}, \boldsymbol{w})) \in \mathcal{F}$ **do**
        $Q_k \leftarrow Q_k \cup q'$
        if $C \cap G(e) \neq \emptyset$ **then**
          $\tilde{\boldsymbol{c}} = \boldsymbol{C}_e\boldsymbol{c} + \bar{\boldsymbol{c}}_e + 0.5(\boldsymbol{C}_e - |\boldsymbol{C}_e|)\boldsymbol{w}$
          $\tilde{\boldsymbol{w}} = |\boldsymbol{C}_e|\boldsymbol{w}$
          $(X_k^{q'}, C_k^{q'}) \leftarrow (\{X_k^{q'}, \boldsymbol{G}_e X\}, \{C_k^{q'}, \mathrm{Box}(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{w}})\})$
        end if
      end for
    end for
  end for
  $\forall q \in Q_k : (X_k^q, C_k^q) \leftarrow \mathrm{ClusterSets}(X_k^q, C_k^q)$
  $S_k \leftarrow \bigcup_{q \in Q_k} (q, X_k^q, C_k^q)$
  if $\mathrm{FixedPoint}(\{S_1, ..., S_k\})$ **then**
    return $\{S_1, ..., S_k\}$
  end if
end for
return $\{S_1, ..., S_{\hat{k}}\}$

However the disadvantages of this representation are the increased computational effort and storage, since some of the points may be redundant, i.e. $\exists X_r \subset X : \mathrm{Conv}(X \setminus X_r) = \mathrm{Conv}(X)$. We note however that the extra computational effort is compensated by the fact that flow-pipes are tighter and a fixed-point is found earlier.

6 Reachability algorithm for HA-CLD

In this section we lay out the complete reachability algorithm and show how guard and invariant intersections are efficiently computed. Additionally, we describe a simple clustering algorithm the utilizes the box representation of clock sets.

6.1 Discrete reachability

Discrete reachability involves evaluating the jump relation $\mathrm{Jump}(\cdot)$, or deriving an over-approximation of it. First a set of flow-pipes, $\mathcal{F} = \{([X_1, z_1]^\uparrow, [C_1, z_1]^\uparrow), ..., ([X_j, z_j]^\uparrow, [C_j, z_j]^\uparrow)\}$, computed for a mode q is intersected with a guard $G(e)$ for each outgoing edge $e = (q, q') \in E$. Then a reset transformation is applied to the intersected flow-pipes for each edge. Finally a clustering and aggregation algorithm, ClusterSets, is applied to the sets so that their growth is reduced in subsequent iterations. For further details see Algorithm 1. While

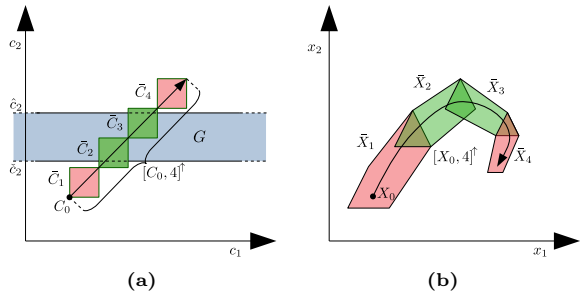


Figure 5: The over-approximated intersection of the flow-pipes $[C_0, 4]^\uparrow$ (5a) and $[X_0, 4]^\uparrow$ (5b) with G, marked in green.

Section 5 describes lines 7-12 of the algorithm, this section is dedicated to lines 13-21.

6.1.1 *Computing over-approximate intersections*

Since guards and invariants in HA-CLD are boxes only defined for clock variables, and a clock segment $\bar{C}_i$ is itself a box, intersections can be performed trivially. On the other hand non-clock segments $\bar{X}_i$ cannot be exactly intersected for the reasons described earlier in Section 3.

To allow approximate intersections we take advantage of the fact that the flow-pipes are finite sets of segments related by a common time-interval $[t_i, t_i + \delta]$ over which they are computed. Specifically, segments which do not intersect with $G(e)$ are discarded. Formally, $\mathcal{F} \cap G(e) \subseteq \{(\bar{X}, \bar{C}) \in \mathcal{F} \mid \bar{C} \cap G(e) \neq \emptyset\}$. A similar procedure is done when intersecting with an invariant $\mathrm{Inv}(q)$.

Consider as an example a system with two clocks $c_{1,2}$ and two state variables $x_{1,2}$. Let $X_0 = \{x\}$ and $C_0 = \mathrm{Box}(\boldsymbol{c}, \boldsymbol{w})$ be the initial sets, and assumed that a single clock guard $G = [-\infty, \infty] \times [\check{c}_2, \hat{c}_2]$ is specified. The flow-pipes $X(t)$ and $C(t)$, their approximations $[X_0, 4]^\uparrow$ and $[C_0, 4]^\uparrow$, and the guard G are visualized in Figure 5. Here the segment tuples $(\bar{C}_1, \bar{X}_1)$ and $(\bar{C}_4, \bar{X}_4)$, highlighted in red, are outside the guard's interior and are therefore discarded, while $(\bar{C}_2, \bar{X}_2)$ and $(\bar{C}_3, \bar{X}_3)$, highlighted in green, are kept.

6.1.2 *Reset map over-approximation*

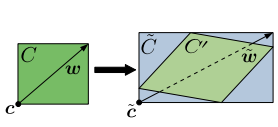


Figure 6: Example over-approximation of a clock reset.

After computing approximate intersections the reset maps as defined in Section 4 are applied to the intersected sets. However, applying a clock reset to a box $C = \mathrm{Box}(\boldsymbol{c}, \boldsymbol{w})$ results in a set C' that is not a box, see Figure 6. To resolve this we replace the standard transformation with a box over-approximation, $\tilde{C} = \mathrm{Box}(\tilde{\boldsymbol{c}}, \tilde{\boldsymbol{w}})$, as depicted in Figure 6, according to:

$$\tilde{\boldsymbol{c}} = \boldsymbol{C}_e \boldsymbol{c} + \bar{\boldsymbol{c}}_e + 0.5(\boldsymbol{C}_e - |\boldsymbol{C}_e|)\boldsymbol{w}, \tag{11}$$

$$\tilde{\boldsymbol{w}} = |\boldsymbol{C}_e| \boldsymbol{w}. \tag{12}$$

This is derived by treating C as a zonotope, and applying the over-approximation technique presented in [16].

6.2 Set clustering and grouping

Once continuous and discrete reachability is complete for an iteration k, the resulting flow-pipe segments are stored in the set arrays X_k^q and C_k^q, which are used in the next iteration to compute new flow-pipes. As a consequence each segment will start its own flow-pipe in the next iteration of the reachability algorithm. Because the number of flow-pipes grows uncontrollably with each iteration, the reachability algorithm becomes infeasible. This is worsened by the increasing number of points for each $\bar{X} \in X_k^q$.

Algorithm 2 Set clustering

function $(X', C') \leftarrow \mathrm{ClusterSets}(X, C)$

1: $(X', C') \leftarrow (\emptyset, \emptyset)$
2: **while** $C \neq \emptyset$ **do**
3: Take $(\bar{X}, \bar{C} = \mathrm{Box}(\boldsymbol{c}, \boldsymbol{w})) \in (X, C)$.
4: $(X, C) \leftarrow (X \setminus \bar{X}, C \setminus \bar{C})$
5: **for each** $(\bar{X}^*, \bar{C}^* = \mathrm{Box}(\boldsymbol{c}^*, \boldsymbol{w}^*)) \in (X, C)$ **do**
6: **if** $(\boldsymbol{c} \leq \boldsymbol{c}^* \wedge \boldsymbol{c}^* < \boldsymbol{c} + \boldsymbol{w}) \vee (\boldsymbol{w} = \boldsymbol{0} \wedge \boldsymbol{c} = \boldsymbol{c}^*)$ **then**
7: $(X, C) \leftarrow (X \setminus \bar{X}^*, C \setminus \bar{C}^*)$
8: $\boldsymbol{w}_\square = \max\{\boldsymbol{w}, \boldsymbol{c}^* + \boldsymbol{w}^* - \boldsymbol{c}\}$
9: $(\bar{X}, \bar{C}) \leftarrow (\bar{X} \cup \bar{X}^*, \mathrm{Box}(\boldsymbol{c}, \boldsymbol{w}_\square))$
10: **end if**
11: **end for**
12: $(X', C') \leftarrow (\{X', \mathrm{Conv}(\bar{X})\}, \{C', \bar{C}\})$
13: **end while**

To reduce the number of flow-pipes, clustered overlapping segments in the clock domain are grouped by a single box hull, while the related non-clock segments are merged and have their redundant points removed using a convex hull algorithm. This procedure is described in Algorithm 2, where we again take advantage of the fact that clock flow-pipe segments are boxes and $C^q = \{\bar{C}_1 = \mathrm{Box}(\boldsymbol{c}_1, \boldsymbol{w}_1), ...\}$. Specifically, any two sets $\bar{C}_{i,j}, i \neq j$, are overlapping if

$$(\boldsymbol{c}_i \leq \boldsymbol{c}_j \wedge \boldsymbol{c}_j < \boldsymbol{c}_i + \boldsymbol{w}_i) \vee (\boldsymbol{w}_i = \boldsymbol{0} \wedge \boldsymbol{c}_i = \boldsymbol{c}_j),$$

holds. If the sets overlap, then they are grouped by the box hull $\mathrm{Box}(\boldsymbol{c}_i, \boldsymbol{w}_{ij})$, where $\boldsymbol{w}_{ij} = \max\{\boldsymbol{w}_i, \boldsymbol{c}_j + \boldsymbol{w}_j - \boldsymbol{c}_i\}$, and $\mathrm{Conv}(\bar{C}_i \cup \bar{C}_j) \subseteq \mathrm{Box}(\boldsymbol{c}_i, \boldsymbol{w}_{ij})$. The corresponding sets $\bar{X}_{i,j}$ are then aggregated by computing $\mathrm{Conv}(\bar{X}_i \cup \bar{X}_j)$ using the QuickHull algorithm [4]. While this approach does not solve the problem completely, it significantly reduces the uncontrollable growth of flow-pipes.

7 Case study

In this section we present the benchmark results that were obtained using our approach and SpaceEx. In the first benchmark, sensor data received by the controller is only delayed. In the second benchmark, data can also be lost.

7.1 Evaluation setup

In the benchmarks we consider a controller that is interfaced to two sensors, see Figure 7, that are strictly periodically triggered by a clock. The data from the two sensors is pre-processed before it is sent to the controller. As a result of the variable execution time $\rho_i \in [\check{\rho}_i, \hat{\rho}_i]$ of the pre-processing

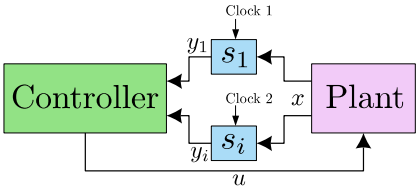


Figure 7: A multiple sensor control system setup.

task, measurement data arrives after a variable delay at the controller. The controller is also executed as a task with execution time $L \in [\check{L}, \hat{L}]$. It is assumed that this task does not wait for data from the sensors, and uses the most recent measurements y_i that are sent to the controller. We may also assume that data can be lost as well during transmission. A common problem of such systems is that verifying safety properties and stability is very challenging, because the mixture of DE and CT dynamics leads to a complex non-linear behavior.

For the considered control system we would like to verify asymptotic stability. Specifically, an initial non-clock set $X_0 = r\mathcal{B}_\infty, r \in \mathbb{R}_+$, is specified, and our fixed-point criterion is reached whenever $X_k \subset X_0$ for some k [10]. SpaceEx and our own algorithm are used to compute the reachable set. These model-checkers are compared based on the number of iterations and time required to reach a fixed-point. Another performance metric that is considered is the over-approximation error introduced by the approaches. However, since it is difficult to acquire numerical information from SpaceEx, we used the graphical plot outputs generated by the tool for our comparison.

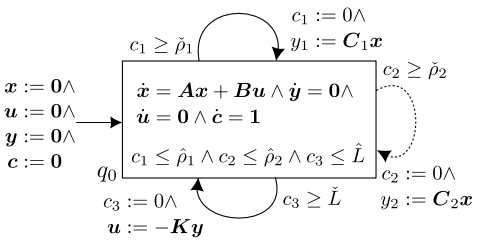


Figure 8: HA-CLD models for benchmark 1.

7.1.1 Benchmark 1 model

In the first benchmark the particular HA-CLD model that we use is shown in Figure 8. Two clocks $c_{1,2}$ model the sampling times of the sensors and c_3 the execution time of the controller task. The variable $\boldsymbol{x}$ is the state of the plant, while $\boldsymbol{u}$ and $\boldsymbol{y}$ are PWC actuation and sensor data vectors, respectively. Each transition represents the completion of a task, which is allowed when a task's respective execution timer reaches its lower period bound. The invariant enforces that a transition is taken when a task reaches its upper bound. A sensor transition stores a new measurement in the variables y_1 or y_2, while a controller transition updates the actuation variable $\boldsymbol{u}$. Similar systems are considered in [27].

The plant in this benchmark is a DC motor with a state $\boldsymbol{x} = \begin{pmatrix}\theta & i\end{pmatrix}$, where θ is the angular velocity of the shaft, and i is the armature load current. Additionally, the system matrices are $\boldsymbol{A} = \begin{pmatrix}-10 & 1\\ -0.02 & -2\end{pmatrix}, \boldsymbol{B} = \begin{pmatrix}0\\ 2\end{pmatrix}$. The motor is controlled by a proportional-gain feedback controller with gain matrix $\boldsymbol{K} = \begin{pmatrix}10 & 1\end{pmatrix}$.

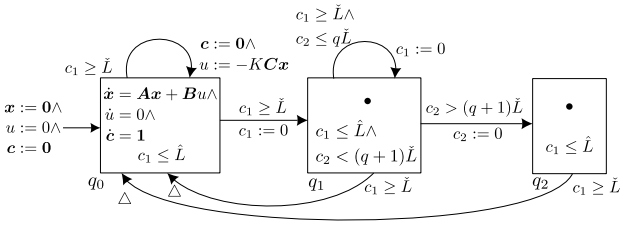


Figure 9: HA-CLD model for benchmark 2.

7.1.2 Benchmark 2

In the second benchmark a similar setup is considered as shown in Figure 7, but then with only one sensor. The other difference is that data can be lost when communicated from the pre-processing task to the controller task. Data loss can occur if a wireless connection is used between the sensor and the controller. It will also occur if previously sent data to the controller is overwritten before it can be read. If the controller fails to receive data from the sensor within an iteration, then the actuation value remains unchanged, otherwise the received value y is used to compute a new actuation. The HA-CLD model that is used for this benchmark is shown in Figure 9. In this model it is assumed that the controller will miss at most q consecutive measurements, after which at least one measurement is always received. The clock variable c_2 is used to keep track of the number of lost measurements, while c_1 represents the execution time of the controller. Note that the modes q_1 and q_2 have the same continuous dynamics as q_0, and that the flow equations are denoted with a $\bullet$. Similarly, transitions (q_1, q_0) and (q_2, q_0) have the same reset map as (q_0, q_0), and these reset maps are denoted with $\triangle$. Now consider for example the case when $q = 2$. Here if the controller does not receive data from the sensor within two consecutive iterations, then it is guaranteed to receive one in the third iteration. A similar setup is considered in [17, 20].

In this benchmark, the plant to be controlled is an aircraft. For this system an LTI state-space representation has been derived[1]. Here the state vector is $\boldsymbol{x} = \begin{pmatrix}\alpha & q & \theta\end{pmatrix} \in \mathbb{R}^3$, where α is angle of attack, and q and θ are the pitch rate and angle of the aircraft, respectively. The derived system matrices are $\boldsymbol{A} = \begin{pmatrix}-0.313 & 56.7 & 0\\ -0.0139 & -0.426 & 0\\ 0 & 56.7 & 0\end{pmatrix}, \boldsymbol{B} = \begin{pmatrix}0.232\\ 0.0203\\ 0\end{pmatrix}$. The pitch angle is the measured variable, such that $y = \boldsymbol{C}\boldsymbol{x} = \begin{pmatrix}0 & 0 & 1\end{pmatrix}\boldsymbol{x}$. It is controlled by a proportional-gain controller, with $K = 0.75$.

[1] http://ctms.engin.umich.edu/CTMS/

7.1.3 *Truck platooning*

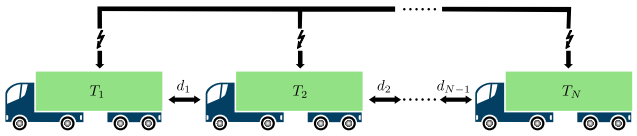
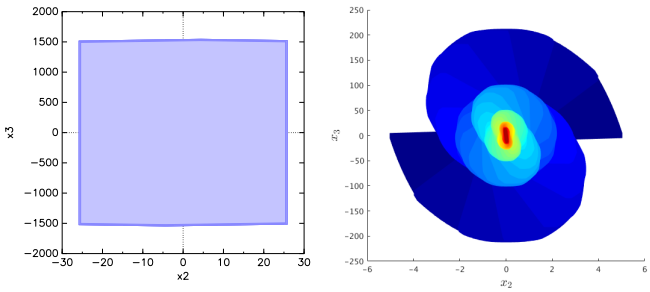


Figure 10: A typical truck platoon.

Truck platooning is a very practically relevant case similar to the benchmarks considered so far, where data-loss and delay during communication occurs quite often. Consider the example in Figure 10. Here N trucks are equipped with sensors and actuators to maintain a certain velocity and acceleration, that communicate relevant data via a wireless network, e.g. an WLAN802.11p network, between each other in order to keep certain distances d_i. A model and control technique of such a platoon is presented by Maschuw et al [20, 21], where the platoon dynamics are approximated by a first-order filter, and the controller is determined by an Linear Matrix Inequality (LMI) formulation, where the communication topology is taken into account. Here they model the events of data-loss as a switched system and find a Common Quadratic Lyapunov Function (CQLF), which gives a (sufficient) condition for asymptotic stability. Finding



(a) **(b)**

Figure 11: Reachable sets of $x_{2,3}$ computed by SpaceEx (11a), and our approach (11b) from benchmark 2, run 1 with $X_0 = 5\mathcal{B}_\infty$. The set colors in 11b range from blue to orange for $k = 1, 2, ...$

a CQLF can be very difficult [10], and thus one may opt to verify stability using reachability analysis, since it covers all of the extreme cases. This is possible since a HA-CLD model can be derived which is similar to the one for benchmark 2 in Figure 9, where a data packet drop-out within the network does occur sporadically. Because the HA-CLD model is similar, we do not present the truck platooning case as a separate benchmark in this paper. However, we do note that such a model would become too complex if the number of trucks is large.

							Our tool		SpaceEx	
Benchmark 1							Time (s)	k	Time (s)	k
$\check{\rho}_1$	$\hat{\rho}_1$	$\check{\rho}_2$	$\hat{\rho}_2$	$\check{L}$	$\hat{L}$	δ				
0.05	0.05	n/a	n/a	0.05	0.05		1.4	30	0.761	27
0.03	0.03	n/a	n/a	0.06	0.06		2.07	39	1.07	36
0.05	0.05	n/a	n/a	0.03	0.07	0.01	21	67	78.32	286
0.03	0.07	n/a	n/a	0.05	0.05		18	80	217.57	470
0.03	0.07	n/a	n/a	0.03	0.07		40	93	1076.96	1135
0.05	0.05	0.05	0.05	0.05	0.05		3.59	51	30.77	247
0.04	0.06	0.05	0.05	0.05	0.05		34	63	2181.31	1561
0.05	0.05	0.04	0.06	0.05	0.05	0.01	43	63	779.42	876
0.05	0.05	0.05	0.05	0.04	0.06		28	66	1347.97	1074
0.02	0.02	0.04	0.04	0.06	0.06		21	104	11.33	111
Benchmark 2							Time (s)	k	Time (s)	k
q				$\check{L}$	$\hat{L}$	δ				
0				0.5	0.5		41.234	79	n/a	n/a
1				0.5	0.5		98	92	n/a	n/a
2				0.5	0.5		2730	1113	n/a	n/a
0				0.3	0.7	0.05	283	143	n/a	n/a
1				0.3	0.7		655	157	n/a	n/a
2				0.3	0.7		3054	232	n/a	n/a

Table 1: Run parameters and evaluation results for benchmarks 1 and 2.

7.2 Evaluation results

We evaluate our approach and SpaceEx with the previously described benchmarks for various values of the sensor bounds on $\rho_{1,2}$, measurement miss factor q, and the controller execution time L, as summarized in Table 1. Here we show runs that result in a fixed-point found after the k-th iteration. The SpaceEx model checker is run using the `STC` scenario and octagonal template. The relative tolerance of the algorithm is set to 0.1, since lower values have occasionally caused the tool to halt. Our approach on the other hand is implemented using MATLAB, with a fixed time step δ, see Table 1. Additionally we make use of the Parallel Computing Toolbox [26] whenever possible. The evaluation PC is a laptop with 16 GB of DDR4 memory and an Intel© Core™ i7-6700HQ CPU clocked at 2.60GHz.

From Table 1 one can conclude that our approach indeed outperforms SpaceEx for the given benchmarks in terms of run-time and number of iterations, except for the first two and last runs in benchmark 1 where the only source of non-determinism is from the transitions. Unfortunately benchmark 2 proved too difficult for SpaceEx to handle, even for the most trivial case of $\check{L} = \hat{L}$, resulting in very large over-approximations of the reachable set, regardless of the options specified in the tool. This is evident from Figure 11, where much larger over-approximation error is introduced by SpaceEx compared to our approach.

The first reason why such a big difference is observed is that flow-pipe computation is done separately for the clock and non-clock variables. Also, intersections are greatly simplified by only allowing guards and invariants for the clocks, and utilizing the technique described earlier, leading to faster

run-times. Finally, we suspect that SpaceEx introduces a significant error by not making use of a tight clustering method. Specifically, instead of applying a convex hull algorithm on sets which are not represented by finite number of points, SpaceEx utilizes template polyhedra. These are polyhedra constructed using support-functions in fixed directions, and generally introduce a large over-approximation error if the number of variables of the system is large with respect to the number of directions.

8 Conclusion

In this paper we present a reachability technique that is optimized for HA-CLD, which is a sub-class of the general HA. Despite that the HA-CLD subclass is more restrictive, it is suitable for modeling and analyzing control systems that experience sampling jitter and data loss. Sampling jitter and data loss can occur in distributed control systems, and is a result of varying execution times and communication delays. From our benchmark results it can be concluded that for the HA-CLD subclass significantly tighter reachability analysis results are obtained than the state-of-the-art model checker SpaceEx can obtain, which uses the more general HA model. Furthermore, the run-time of our model checker was up-to 65 times smaller.

Tighter results were obtained by making use of the explicit separation of clock and non-clock variables in the HA-CLD model, as well as that guards and invariants can only be specified for clocks in this model. These restrictions make guard and invariant intersections trivial and allow the use of a box representation for the clock flow-pipes. The non-clock flow-pipes are represented by sets of points and are tighter, but operations on this set representation are more computationally intensive. An over-approximated intersection of the non-clock flow-pipes with guards and invariants can be obtained efficiently. This was achieved by exploiting the fact that the segments of the clock and non-clock flow-pipes are related by fixed time-invervals over which the segments are computed, and by discarding tuples of related clock and non-clock segments, provided that a clock segment in a tuple is outside the interior of a given guard or invariant.

References

[1] M. Al Khatib, A. Girard, and T. Dang. 2015. Stability Verification of Nearly Periodic Impulsive Linear Systems using Reachability Analysis. *IFAC-PapersOnLine* 48, 27 (2015), 358–363.

[2] M. Al Khatib, A. Girard, and T. Dang. 2017. Self-Triggered Control for Sampled-data Systems using Reachability Analysis. *IFAC-PapersOnLine* 50, 1 (2017), 7881–7886.

[3] S. Bak and P. S. Duggirala. 2017. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 173–178.

[4] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. 1996. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software (TOMS)* 22, 4 (1996), 469–483.

[5] L. Benvenuti et al. 2008. Reachability computation for hybrid systems with Ariadne. In *Proc. of the 17th IFAC World Congress*. 8960–8965.

[6] S. Bogomolov et al. 2018. Reach Set Approximation through Decomposition with Low-dimensional Sets and High-dimensional Matrices. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*. ACM, 41–50.

[7] X. Chen. 2015. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. Ph.D. Dissertation. PhD thesis, RWTH Aachen University.

[8] R.A. DeCarlo. 1989. *Linear Systems: A State Variable Approach with Numerical Implementation*. Prentice-Hall, Inc. 215–215 pages.

[9] P.S. Duggirala and M. Viswanathan. 2016. Parsimonious, Simulation Based Verification of Linear Systems. In *International Conference on Computer Aided Verification*. Springer, 477–494.

[10] V.S. El Hakim and M.J.G. Bekooij. 2018. Stability Verification of Self-Timed Control Systems using Model-Checking. In *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 312–319.

[11] J.A. Ferrez, K. Fukuda, and T.M. Liebling. 2001. Cuts, Zonotopes and Arrangements. *The sharpest Cut. SIAM Series on Optimization* (2001).

[12] G. Frehse et al. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV) (LNCS)*. Springer.

[13] G. Frehse, R. Kateja, and C. Le Guernic. 2013. Flowpipe Approximation and Clustering in Space-Time. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*. ACM, 203–212.

[14] A. Girard. 2005. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 291–305.

[15] A. Girard and C. Le Guernic. 2008. Zonotope/Hyperplane Intersection for Hybrid Systems Reachability Analysis. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 215–228.

[16] A. Girard, C. Le Guernic, and O. Maler. 2006. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 257–271.

[17] P. Kumar et al. 2012. A Hybrid Approach to Cyber-Physical Systems Verification. In *Proceedings of the 49th Annual Design Automation Conference*. ACM, 688–696.

[18] C. Le Guernic and A. Girard. 2010. Reachability Analysis of Linear Systems using Support Functions. *Nonlinear Analysis: Hybrid Systems* 4, 2 (2010), 250–262.

[19] J. Lygeros. 2004. Lecture notes on hybrid systems. In *Notes for an ENSIETA workshop*.

[20] J.P. Maschuw and D. Abel. 2010. Longitudinal Vehicle Guidance in Networks with changing Communication Topology. *IFAC Proceedings Volumes* 43, 7 (2010), 785–790.

[21] J.P. Maschuw, G.C. Keßler, and D. Abel. 2008. LMI-based control of vehicle platoons for robust longitudinal guidance. *IFAC Proceedings Volumes* 41, 2 (2008), 12111–12116.

[22] S. Schupp et al. 2017. HyPro: A C++ Library for State Set Representations for Hybrid Systems Reachability Analysis. In *Proc. of the 9th NASA Formal Methods Symposium (NFM'17) (LNCS)*, Vol. 10227. Springer International Publishing, 288–294.

[23] S. Schupp and E. Ábrahám. 2018. Efficient Dynamic Error Reduction for Hybrid Systems Reachability Analysis. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 287–302.

[24] S. Schupp, J. Nellen, and E. Abraham. 2017. Divide and Conquer: Variable Set Separation in Hybrid Systems Reachability Analysis. In *Proc. of the 15th Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL'17) (EPTCS)*, Vol. 250. Open Publishing Association, 1–14.

[25] S. Schupp, J. Winkens, and E. Ábrahám. 2018. Context-Dependent Reachability Analysis for Hybrid Systems. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE, 518–525.

[26] G. Sharma and J. Martin. 2009. MATLAB®: A Language for Parallel Computing. *International Journal of Parallel Programming* 37, 1 (2009), 3–36.

[27] Q. Zhu and H. Zeng. 2012. Stability Analysis of Multi-rate Switched Networked Systems with Short Time Delay. In *Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference on*. IEEE, 642–646.