Symmetry-based Abstractions for Hybrid Automata

Hussein Sibai, *member* and Sayan Mitra, *senior member*

***Abstract*— A symmetry of a dynamical system is a map that transforms any trajectory to another trajectory. Abstractions have been a key building block in the theory and practice of hybrid automata analysis. We introduce a novel abstraction for hybrid automata based on the symmetries of their modes. The abstraction procedure combines different modes of a concrete automaton $\mathscr{A}$, whose trajectories are related by symmetries, into a single mode of the constructed abstract automaton $\mathscr{B}$. The abstraction procedure sets the invariant of an abstract mode to be the union of the symmetry-transformed invariants of the concrete modes. Similarly, it sets the guard and reset of an abstract edge to be the union of the symmetry-transformed guards and resets of the concrete edges. We establish the soundness of the abstraction using a forward simulation relation (FSR) and provide a running example. The abstraction achieves an order of magnitude speedup when used for the safety verification of vehicles pursuing reach-avoid tasks.**

***Index Terms*— hybrid systems, abstraction, symmetry, formal methods, reachability analysis.**

I. Introduction

Hybrid system models bring together continuous and discrete behaviors [1]–[3], and have proven to be useful in the design and analysis of a wide variety of systems, ranging over automotive, medical, manufacturing, and robotics applications. Exact algorithmic solutions for many of the synthesis and analysis problems for hybrid systems are known to be computationally intractable [4]. Therefore, one aims to develop approximate solutions, and the main approach is to work with an *abstraction* of the hybrid system model. Roughly, an abstraction of a hybrid automaton (HA) $\mathscr{A}$ is a simpler automaton $\mathscr{B}$ that subsumes all behaviors of $\mathscr{A}$. For example, $\mathscr{B}$ may have fewer variables than $\mathscr{A}$, or fewer modes, or it may have linear or rectangular dynamics approximating $\mathscr{A}$'s nonlinear dynamics. $\mathscr{A}$ is called the *concrete*, and $\mathscr{B}$ the *abstract* or *virtual* automaton. Ideally, $\mathscr{B}$ is simpler and yet a useful over-approximation of $\mathscr{A}$, and the formal analysis and synthesis of $\mathscr{B}$ is tractable.

Several important verification and synthesis techniques for HA have relied on abstractions. The early decidability results for the verification of rectangular HA were based on creating discrete and finite state abstractions [4]–[6]. Decidability results based on abstractions for more general classes were presented in [7], [8]. In a sequence of papers [9]–[11], metric-based abstractions were developed for more general hybrid models and shown to be useful for both verification and synthesis. Techniques have been developed for automatically making abstractions more precise using counterexamples [12]–[15]. Finally, several practical approaches have been proposed for computing abstractions using linearization [16], state space partitioning [13], [17], and hybridization [18].

An important characteristic of dynamical systems that has *not* been explored for constructing abstractions is *symmetry*. Symmetry in a dynamical system $\dot{x} = f(x,p)$, with parameter p, is a map γ that transforms solutions (or *trajectories*) of the system to other trajectories. For example, consider a trajectory $\xi(t) = \xi(x_0,p,t)$ for $t \in \mathbb{R}^+$, of a vehicle, starting from x_0 and following waypoint p. When ξ is *shifted* by a, the result $\gamma_a(\xi)$ is just the trajectory of the vehicle starting from $\gamma_a(x_0)$ following $\rho_a(p)$, which is p shifted by a. We call this trajectory ξ', which is equal to $\xi(\gamma_a(x_0),\rho_a(p),\cdot)$. That is, the symmetry γ_a relates different trajectories of the system. Symmetries have been used for studying stability of feedback systems [19], designing observers [20] and controllers [21], [22], accelerating safety verification [23]–[25], performing guaranteed integration [26], analyzing neural networks [27], and deriving conservation laws [28] using Noether's theorem [29].

Since both of ξ and ξ' are solutions of the system, and γ_a can compute ξ' from ξ, then, in a sense, f has some redundancy. A simpler version of f, would only have ξ as a solution, and allows us to derive ξ' using γ_a. In this paper, we create such versions by defining *symmetry-based abstractions* for HA.

Given a HA $\mathscr{A}$ having a set of discrete states (or *modes*) P and a family of symmetry maps Φ, our abstraction partitions P to create the abstract automaton $\mathscr{A}_v$. Each *equivalent class* of the partition is represented by a single mode in $\mathscr{A}_v$. Any trajectory ξ of any mode $p \in P$, can be transformed using a particular symmetry map from Φ that is a function of p to get a trajectory ξ_v of the representative abstract mode p_v, and vice versa. Accordingly, all concrete edges between any two equivalent mode classes would be represented with a single abstract edge. A set of concrete edges represented with the same abstract edge forms an equivalent edge class. The edges of $\mathscr{A}$ are annotated with *guards* and *resets*. These dictate when the discrete transitions between modes can be taken and how the state would be updated, respectively. The abstraction transforms the guards and resets of all concrete edges using the same symmetry maps in Φ used to transform trajectories. Then, the abstraction procedure unions all of the transformed guards and resets of an edge equivalent class to get the guard

Hussein Sibai is with the Computer Science and Engineering department at Washington University in St. Louis, St. Louis MO 63130, USA (email: sibai@wustl.edu) and Sayan Mitra is with the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign, Urbana IL 61801, USA (email: mitras@illinois.edu). This work was supported by AFOSR Research Grant (FA9550-23-1-0337) HyDDRA Hybrid Dynamics - Deconstruction and Aggregation.

and reset of the corresponding abstract edge. This means that an execution of $\mathscr{A}_v$ can transition over an edge e_v if any of the transformed guards of the edges of $\mathscr{A}$ that e_v represents is satisfied. Moreover, the execution would split into several executions after a reset. Each of these executions start from a transformed version of the state defined by the reset of an edge of $\mathscr{A}$ that is represented by e_v. We establish the soundness of the abstraction using a forward simulation relation (FSR) (Theorem 3). We present a running example of a robot with a 3-dimensional kinematic single-track model following a sequence of waypoints and use translation and rotation symmetries to design its abstract automaton.

Our abstraction does not change the dimension of the state space, nor does it change the complexity of the dynamics, while resulting in an automaton with fewer modes. The abstraction can be useful for solving several problems related to tractability of synthesis and verification. In an earlier paper [30], we used this abstraction to accelerate *safety verification* without formally defining it. Safety verification requires us to check if there is any execution of a HA that starts from a predefined compact set of initial states and intersects a predefined set of unsafe states. In [30], we presented a tool called SceneChecker that implements the abstraction as well as a simple refinement algorithm. We showed that it achieves an order of magnitude speedup in verification time of several scenarios involving drones and cars navigating cluttered environments with hundreds of obstacles according to predefined plans with hundreds of waypoints. In this technical note, we formally define the abstraction using symmetries. We use SceneChecker for a simple safety verification problem, concluding the running example.

A. HA Abstractions: brief literature review

Simulation relations and abstractions have been an essential part of the formulation of HA [31], [32]. Several important verification and synthesis techniques for HA have relied on abstractions. We briefly discuss some of the existing abstraction-refinement methods for verifying HA.

Doyen et al. presented a method that abstracts affine HA using rectangular HA [33]. Jha et al. [34] presented an abstraction-refinement algorithm for linear HA that leverages the power of linear programming for counterexample validation and refinement. Bogomolov et al. [35] defined an abstraction-refinement algorithm that abstracts a given affine HA in two steps. First, their algorithm overapproximates the dynamics of each mode using a polytope-based differential inclusions. Then, it combines different concrete modes in the same abstract mode. Zutshi et al. [36] designed a counterexample-guided abstraction refinement algorithm (CEGAR) for falsification, instead of verification, for HA. Their algorithm implicitly constructs tiling-based abstractions while searching for a valid execution of the automaton that violates a given property. A line of research by Prabhakar and Roohi et al. [14], [15] presented a CEGAR-based verification algorithm for nonlinear HA. Their method abstracts nonlinear hybrid atuomata by partitioning the invariant state space of each mode and over-approximating the dynamics using polytope-based differential inclusions.

Our abstraction method for HA is the first one that uses symmetries. It maps the dynamics of the concrete modes to the representative ones of the abstract modes through symmetries instead of over-approximating them using rectangular or polytopic differential inclusions. Thus, our algorithm does not over-approximate continuous dynamics besides over-approximating the invariants of the modes, while the other methods do. That is a very important advantage for our method, especially in safety verification and synthesis applications, where over-approximation causes conservatism. Finally, the other abstraction-refinement algorithms are orthogonal to ours and can be composed with it.

II. Model and problem statement

Notations: We denote by $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}^{>0}$, and $\mathbb{R}^{\geq 0}$ the sets of natural numbers, reals, positive reals, and non-negative reals, respectively; by $|S|$, the cardinality of a finite set S; by $seq.len$, the length of a finite sequence seq; by $seq[i:j]$, its elements between i and j, inclusive; by $[N]$, the set $\{0,\ldots,N-1\}$, where $N \in \mathbb{N}$. Given two vectors $v \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$, we define $[v,u]$ to be the vector of length $n+m$ that results from appending u to v. Given two vectors $v_l, v_r \in \mathbb{R}^n$, we define $[v_l; v_r]$ to be the n-dimensional hyper-rectangle (l_∞ ball) with v_l defining the left endpoints and v_r defining the right endpoints of the per-dimension intervals. Given $\varepsilon \in (\mathbb{R}^+)^n$, we denote by $B(v,\varepsilon)$ the hyper-rectangle centered at v with the i^{th} dimension sides having a length of $2\varepsilon[i]$, for any $i \in [n]$. We denote $diag(v)$ to be the diagonal matrix with diagonal v. Given a function $\gamma : \mathbb{R}^n \to \mathbb{R}^n$ and a set $S \subseteq \mathbb{R}^n$, we define $\gamma(S) = \{\gamma(x) \mid x \in S\}$. We also define $\gamma(v) = [\gamma(v_1),\ldots,\gamma(v_k)]$, $\forall v \in \mathbb{R}^{n\times k}$. We define $\arctan_2(y,x)$ to be the phase of the complex number $x+jy$.

A. Hybrid dynamics

We will use a standard HA modeling framework [32], [37].

Definition 1: A *hybrid automaton* is a tuple $\mathscr{A} = \langle X, P, inv, E, guard, reset, f\rangle$, where

(a) $X = \mathbb{R}^n$ is the continuous state space and $P \subseteq \mathbb{R}^m$ is a (possibly infinite) set of discrete states. Continuous states are simply called *states* and the discrete ones are called *modes* or *parameters*.
(b) $inv : P \to 2^X$ defines the *invariant* set of states that the automaton's state must belong to when in a certain mode,
(c) $E \subseteq P \times P$ is the set of directed edges over modes that define mode transitions,
(d) $guard : E \to 2^X$ defines the set of states from which an edge transition is enabled,
(e) $reset : X \times E \to 2^X$ defines the updated (post) state after a transition is taken, and
(f) $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a function, or *vector field*, that defines the continuous state evolution. It is locally-Lipschitz continuous in the first argument to guarantee the existence and uniqness of solutions. We further assume that f ensures that they exist for all $t \in \mathbb{R}^{\geq 0}$.

Formally, a function $\xi : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{\geq 0} \to \mathbb{R}^n$ defines the *trajectories of* $\mathscr{A}$ if it is differentiable in its third argument, and given an $x_0 \in \mathbb{R}^n$ and a $p \in \mathbb{R}^m$, $\xi(x_0,p,0) = x_0$ and for all $t \in \mathbb{R}^{\geq 0}$,

$$\frac{d}{dt}\xi(x_0,p,t) = f(\xi(x_0,p,t),p). \tag{1}$$

For any $x_0 \in \mathbb{R}^n$ and $p \in \mathbb{R}^m$, the function of time, or *trajectory*, resulting from setting the first and second arguments of ξ to x_0 and p, respectively, is the unique solution $\xi(x_0, p, \cdot)$ of system (1) starting from x_0 following p, since f is locally-Lipschitz continuous. We assume that it exists for any $t \in \mathbb{R}^{\geq 0}$. For any time-bounded trajectory ξ, i.e., defined over a finite closed interval $[0,T]$ in the third argument, $dur(\xi)$ is T. The first and last states, i.e., $\xi(x_0,p,0)$ (or, equivalently, x_0) and $\xi(x_0,p,T)$, are denoted by $\xi.fstate$ and $\xi.lstate$, respectively.

The edge set E, the *guard*, and the *reset* together define the discrete transitions. For simplicity, for an edge $e = (p,p') \in E$, we denote its source mode p by $e.src$ and its destination mode p' by $e.dest$. Moreover, we abuse notation and denote $guard((p,p'))$ by $guard(p,p')$. Then, $guard(p,p')$ is the set from which a transition from mode p to mode p' is possible. For any state $x \in guard(p,p')$, the post-state x' after the transition has to be in $reset(x,(p,p'))$. Such state-mode pairs $((x,p),(x',p'))$ define the transitions of $\mathscr{A}$ and we write $(x,p) \to (x',p')$. Also, by representing edges as pairs of modes, we assume without loss of generality that there exists at most a single edge from any mode to any other mode. The abstraction in Section IV generalizes straightforwardly to the case where multiple edges between the same pair of modes exist. Finally, urgent transitions can be enforced using *inv*: guards may be ignored as long as the state belongs to the current mode invariant. Otherwise, the automaton transitions or terminates.

The semantics of a HA is defined by its executions which are sequences of trajectories and transitions. An *execution* of $\mathscr{A}$ is a sequence of pairs of trajectories and modes $\sigma = (\xi_0,p_0),(\xi_1,p_1),\ldots$, where (a) each ξ_i is a trajectory of $\mathscr{A}$ which belongs to $inv(p_i)$ at all times and (b) each $(\xi_i.lstate, p_i) \to (\xi_{i+1}.fstate, p_{i+1})$ is a transition as defined above. A *finite* and *time-bounded* execution has a finite number of discrete transitions and all of its trajectories are time-bounded. The duration of a finite and time-bounded execution $\sigma = (\xi_0,p_0),(\xi_1,p_1)\ldots(\xi_k,p_k)$ is $dur(\sigma) = \sum_i dur(\xi_i)$ and its last state is $\sigma.lstate = \xi_k.lstate$.

Example 1 (Robot following waypoints) Consider a robot following a sequence of waypoints $\{w_i \in \mathbb{R}^2 \mid i \in [4]\}$ on the plane connected with directed segments $\{r_i \in \mathbb{R}^4 \mid i \in [5]\}$ forming an axis-aligned rectangle centered at the origin (see Figure 1a). For any segment $r_i \in \mathbb{R}^4$, we define $r_i.src$ to be its first two coordinates $r_i[0:1]$ representing its start waypoint and $r_i.dest$ to be its last two coordinates $r_i[2:3]$ specifying its end waypoint. The robot starts from an arbitrary state in some initial set $X_{init} \subset \mathbb{R}^3$, where the first two dimensions represent its position and the third representing its heading angle, following waypoint w_0. We fix four possible rectangle dimensions: ε_0, ε_1, ε_2, $\varepsilon_{inv} \in (\mathbb{R}^{>0})^2$, with $X_{init} := B(r_0.src, \varepsilon_2)$ and $\varepsilon_{inv} \leq \varepsilon_1 \leq \varepsilon_0$ element-wise. We say that the robot reached w_0 following segment r_0 if it is located in the rectangle $B(w_0,\varepsilon_0)$. If it was following segment r_4 instead, we say it reached w_0 if it is located in the smaller rectangle $B(w_0,\varepsilon_1)$. For any $i \in \{1,2,3\}$, we say that it reached the waypoint w_i following segment r_i if it is located in $B(w_i,\varepsilon_1)$. Once the robot reaches a waypoint w_i, it may start following the next segment, or keep following the current one. However, once it is located in the rectangle $B(w_i,\varepsilon_{inv})$, it *must* switch to following the next segment.

To formalize the dynamics of the robot in this scenario, we construct a corresponding hybrid automaton. We represent the five segments using five modes in the automaton, i.e., $P = \{r_0,r_1,r_2,r_3,r_4\}$. In each mode, the robot would follow the destination waypoint of the corresponding segment. We define the resulting automaton as follows: $\mathscr{B} = \langle X,P,inv,E,guard,reset,f\rangle$ (shown in Figure 1b), where

(a) $X = \mathbb{R}^3$, representing the position and heading angle with respect to the $x[0]$-axis, and $P := \{p_i := r_i \mid i \in [5]\} \subseteq \mathbb{R}^4$, the set of segments in Figure 1a,

(b) $\forall p \in P, inv(p) := \mathbb{R}^3 \backslash (B(p.dest, \varepsilon_{inv}) \times \mathbb{R})$,

(c) $E := \{e_0 := (p_0,p_1), e_1 := (p_1,p_2), e_2 := (p_2,p_3), e_3 := (p_3,p_4), e_4 := (p_4,p_1)\}$,

(d)
$$guard(e_i) := \begin{cases} B(w_i,\varepsilon_0) \times \mathbb{R}, & \text{if } i = 0, \\ B(w_i,\varepsilon_1) \times \mathbb{R}, & \text{if } i \in \{1,2,3,4\}. \end{cases}$$

(e) $\forall x \in X, e \in E$, $reset(x,e) := \{x\}$, and

(f) $\forall x \in \mathbb{R}^3, \forall p \in \mathbb{R}^4$, $f(x,p) := [v\cos(x[2]), v\sin(x[2]), v\tan(\alpha)/L]$, where α is the steering angle which we set to $\arctan_2(p[3]-x[1], p[2]-x[0]) - x[2]$ (the heading angle tracking error) projected to a predefined interval $[\alpha_l,\alpha_r] \subset (-\pi/2,\pi/2)$, $(x[0],x[1])$ is the location of the robot, $x[2]$ is its heading angle, and v and L are its constant speed and length. These dynamics are those of the rear wheel in a kinematic single-track model, with v being the forward speed and α being the steering angle of the forward wheel [38].

III. Symmetry and Equivariant Dynamical Systems

In this section, we present an existing definition of symmetry for continuous-time dynamical systems with parameters and a sufficient condition for a map to be a corresponding symmetry.

A symmetry map γ acts on $\mathbb{R}^n$, i.e. $\gamma : \mathbb{R}^n \to \mathbb{R}^n$, such that given a solution of the system, it maps it to another valid solution. Throughout this manuscript, let Γ be a group of differentiable transformations acting on $\mathbb{R}^n$.

Definition 2 (Definition 1 in [39]) For any $\gamma \in \Gamma$, we say it is a symmetry of (1) if for any trajectory $\xi(x_0,p,\cdot)$, $\gamma(\xi(x_0,p,\cdot))$ is a trajectory as well.

By $\gamma(\xi(x_0,p,\cdot))$ in the definition above, we mean applying γ to every state in the trajectory, or more formally, $\gamma(\xi(x_0,p,\cdot)) = \gamma \circ \xi(x_0,p,\cdot)$, where $\circ$ is the function composition operator.

Definition 3 ([39]) The vector field $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is said to be Γ-equivariant if for any $\gamma \in \Gamma$, there exists $\rho : \mathbb{R}^m \to \mathbb{R}^m$ such that,

$$\forall x \in \mathbb{R}^n, \forall p \in \mathbb{R}^m, \ \frac{\partial \gamma}{\partial x} f(x,p) = f(\gamma(x), \rho(p)). \tag{2}$$

In this paper, we assume that the user of our abstraction method will provide the symmetries and the HA. The following theorem shows that it is sufficient to check the condition in equation (2) to prove that the maps in Γ are symmetries of (1).

Theorem 1 (Theorem 10 in [39]) If f in (1) is Γ-equivariant, then all maps in Γ are symmetries of (1). Moreover, for any $\gamma \in \Gamma$ and $\rho : \mathbb{R}^m \to \mathbb{R}^m$ that satisfy equation (2), $x_0 \in \mathbb{R}^n$, and $p \in \mathbb{R}^m$, $\gamma(\xi(x_0,p,\cdot)) = \xi(\gamma(x_0),\rho(p),\cdot)$.

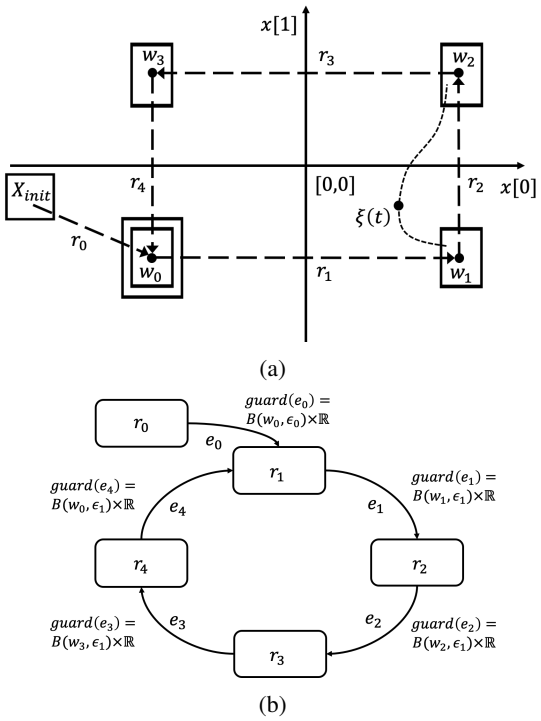


Fig. 1: (1a) A robot, with a state containing its position and orientation, following a sequence of 2D waypoints forming a rectangle starting from its initial set X_{init}. It reaches a waypoint if it reaches the rectangle centered at it. It has to reach the larger rectangle centered at w_0 when starting from X_{init}. It can transition to following the next waypoint once it reaches the one it is following, but might as well stay following the current one. Once it enters a smaller rectangle centered at the waypoint (not shown in the figure for simplicity) it must transition to following the next waypoint. (1b) The state machine representing the discrete transitions of the hybrid automaton $\mathscr{B}$ describing the scenario in Figure 1a with the segments being the modes. The resets are omitted since they are just the identity map for all the modes. The invariants are also omitted for simplicity.

Note that if the maps in Γ are linear functions, then the condition in equation (2) becomes $\gamma(f(x,p)) = f(\gamma(x),\rho(p))$.

Example 2 (Coordinate transformation symmetry) Consider the robot presented in Example 1 and the corresponding automaton of the scenario described in Figure 1a. Fix a $p^* \in \mathbb{R}^4$. We define $\gamma_{ct} : \mathbb{R}^3 \to \mathbb{R}^3$ and $\rho_{ct} : \mathbb{R}^4 \to \mathbb{R}^4$ to be the maps that transform the coordinate system of the plane where the robot and segments reside. These maps transform the coordinate system so that p^* becomes collinear with the $x[0]$-axis and $p^*.dest$ be its origin. Formally, $\forall x \in \mathbb{R}^3$ and $p \in \mathbb{R}^4$,

$$\gamma_{ct}(x) := [\mathbf{R}_\theta(x[0:1] - p^*.dest), x[2] - \theta], \quad (3)$$

$$\rho_{ct}(p) := [\mathbf{R}_\theta(p.src - p^*.dest), \mathbf{R}_\theta(p.dest - p^*.dest)], \quad (4)$$

where $\theta := \arctan_2(p^*.dest[1] - p^*.src[1], p^*.dest[0] - p^*.src[0])$ and

$$\mathbf{R}_\theta := \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad (5)$$

is the rotation matrix with angle θ. Then, we can check with simple algebra, that for all $x \in \mathbb{R}^3$ and $p \in \mathbb{R}^4$, $\frac{\partial \gamma_{ct}}{\partial x} f(x,p) = f(\gamma_{ct}(x), \rho_{ct}(p))$. An illustrating example of this coordinate transformation and the resulting new trajectory and waypoint being followed are shown in Figure 2.

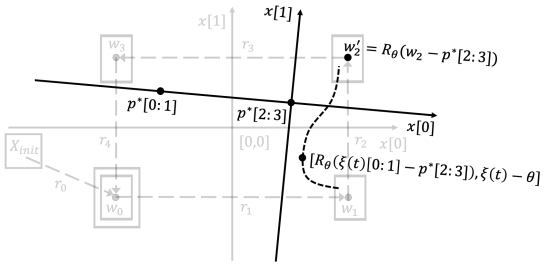


Fig. 2: Changing the axes of the coordinate system so that the segment connecting $p^*.src = p^*[0:1]$ to $p^*.dest = p^*[2:3]$ is the new $x[0]$-axis and $p^*.dest$ is the new origin. Such a transformation does not affect the intrinsic behavior of the robot, but only transforms the states in its trajectories to conform with the new coordinate system. Such a coordinate transformation is a valid symmetry.

IV. Virtual or Abstract Hybrid Automaton

In this section, we present our symmetry-based abstraction for HA along with the corresponding FSR. Our abstract, or virtual, automata have fewer numbers of modes and edges, than their concrete counterparts. Our abstraction is an extension of the concept of virtual system for parameterized dynamical systems that we defined in [24], to HA. We use subscript v to denote the components of the abstract, or virtual, automaton and no subscript for those of the concrete, or real, one.

A. Creating the abstract automaton

In order to create the abstract, or virtual, HA, a family of symmetries is needed.

Definition 4 (virtual map) Given a hybrid automaton $\mathscr{A}$ with a Γ-equivariant vector field f, a *virtual map* is a set

$$\Phi := \{(\gamma_p, \rho_p)\}_{p \in P}, \quad (6)$$

where $\{\gamma_p\}_{p\in P} \subseteq \Gamma$ and $\forall p \in P$, $\rho_p : \mathbb{R}^m \to \mathbb{R}^m$ satisfies, along with γ_p, the following condition that is similar to equation (2): $\forall x \in \mathbb{R}^n$, $\frac{\partial \gamma_p}{\partial x} f(x,p) = f(\gamma_p(x), \rho_p(p))$.

To construct the abstract HA, for any $p \in P$, we are only going to use ρ_p in Φ to transform p. Hence, for simplicity of notation, we define the function $rv : P \to P$, where $\forall p \in P$,

$$rv(p) := \rho_p(p). \quad (7)$$

This function will be used in Definition 5 to map concrete modes to representative abstract ones. Moreover, its preimage will be the equivalent classes of concrete modes that the abstract modes represent. From Theorem 1, it follows that for any $p \in P$, γ_p transforms the trajectories of the concrete HA in mode p to trajectories in mode $rv(p)$.

A virtual map Φ ideally maps multiple concrete modes to the same abstract mode. It is related to the concept of Cartan's moving frame [40]. A moving frame for a Γ-equivariant dynamical system is a map from its state space to Γ. It selects for each state a symmetry from Γ that when applied to that state, it projects it to a lower-dimensional submanifold. This concept has been used, for example, to design invariant observers [20] and to reduce the dimensionality of dynamic programming [22]. In our case, one can potentially select a submanifold of $\mathbb{R}^m$ and try to compute a moving frame that defines the virtual map. The moving frame would map each $p \in P$ to a pair (γ, ρ) that satisfies the conditions in Definition 4 and has ρ that maps p to the selected submanifold.

Definition 5 (Abstract automaton) Given a hybrid automaton $\mathscr{A}$ and a virtual map Φ, we define the *abstract (virtual) hybrid automaton* as the following tuple:

$$\mathscr{A}_v := \langle X_v, P_v, inv_v, E_v, guard_v, reset_v, f_v \rangle, \text{ where}$$

(a) $X_v := X$ and $P_v := \{rv(p) | p \in P\}$,
(b) $inv_v(p_v) := \bigcup_{p \in rv^{-1}(p_v)} \gamma_p\big(inv(p)\big)$,
(c) $E_v := rv(E) = \{(rv(p_1), rv(p_2)) | (p_1, p_2) \in E\}$,
(d) $\forall e_v \in E_v$, $guard_v(e_v) := \bigcup_{e \in rv^{-1}(e_v)} \gamma_{e.src}\big(guard(e)\big)$,
(e) $\forall x_v \in X_v, e_v \in E_v$,

$$reset_v(x_v, e_v) := \bigcup_{e \in rv^{-1}(e_v)} \gamma_{e.dest}\Big(reset\big(\gamma^{-1}_{e.src}(x_v), e\big)\Big), \text{ and}$$

(f) $\forall x_v \in \mathbb{R}^n$, $\forall p_v \in \mathbb{R}^m$, $f_v(x_v, p_v) := f(x_v, p_v)$.

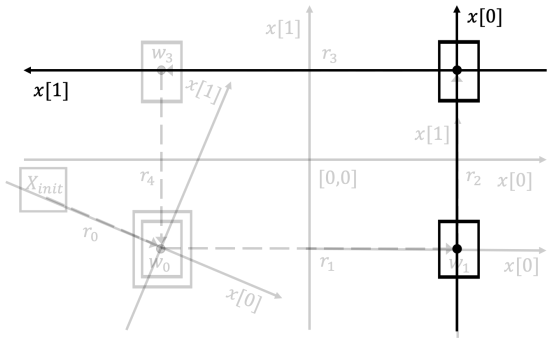
The trajectories and executions of the virtual hybrid automaton $\mathscr{A}_v$ are defined as in Definition 1. In the case where multiple edges exist between a pair of modes, the abstraction can be straightforwardly generalized to represent these edges too with the same abstract edge.

Example 3 (Abstract HA with a coordinate transformation virtual map) To construct its abstract HA for the concrete HA of Example 1, we need a virtual map Φ first. For every mode $p \in P$, we define γ_p and ρ_p to be the coordinate transformation maps γ_{ct} and ρ_{ct} that we presented in Example 2. Recall that we need to fix a mode p^* according to which γ_{ct} and ρ_{ct} transform the coordinate systems of the state and mode spaces. To construct the virtual system, when defining γ_p and ρ_p in Φ, for each mode $p \in P$, we choose p^* to be equal to p. Figure 3a shows a visualization of transforming p_2 using ρ_{p_2}. The concrete mode p_2 gets mapped to $p_{v,2}$, which is a segment on the $x[0]$-axis, as shown in Figures 3b and 3c. This means that there will be as many abstract modes as there are segments with distinct lengths of segments in the path.

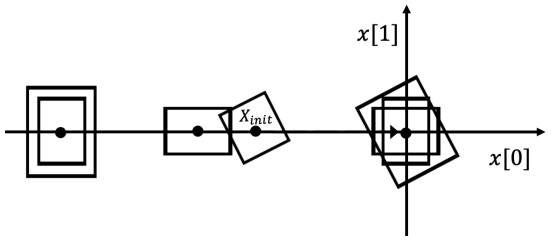
Note that the robot in this example has dynamics that only depend on the destination waypoint, instead of the whole segment. Hence, modifying ρ_{ct} to transform both the source and destination waypoints to the origin, i.e., the segment becoming a single point, while preserving γ_{ct} to be the same as that of Example 2, would also be a valid symmetry of the dynamics. Thus, a set of symmetries that map all segments to the same segment with zero length starting and ending at the origin would also be a valid virtual map for the robot in this example.

Back to the abstract automaton definition, the set $guard(e_2)$, becomes $B(0, [\varepsilon_1[1], \varepsilon_1[0]]) \times \mathbb{R}$, after applying γ_{p_2}. This rectangle will be part of the guard of the abstract edge $e_{v,2}$, per Definition 5.(d). Its projection to the first two dimensions is shown as the rectangle $B(0, [\varepsilon_1[1], \varepsilon_1[0]])$ centered at the origin in Figure 3b. Similarly, the set $inv(p_2)$ becomes $\mathbb{R}^3 \backslash (B(0, [\varepsilon_{inv}[1], \varepsilon_{inv}[0]]) \times \mathbb{R})$ after applying γ_{p_2}, which in turn becomes part of the invariant of the abstract mode $p_{v,2}$.

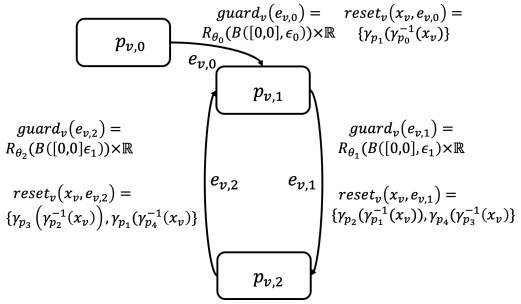
Figure 3a also shows that the rectangle $B(w_1, \varepsilon_1)$, which is $guard(e_1)[0:1]$, becomes $B(w_1 - w_2, [\varepsilon_1[1], \varepsilon_1[0]])$, a rectangle centered at $w_1 - w_2$ and rotated with $\frac{\pi}{2}$ radians, after applying γ_{p_2}. Recall that the reset of any edge of $\mathscr{B}$ is just the identity map. Hence, the rotated rectangle centered at $w_1 - w_2$ represents $\gamma_{p_2}(reset(guard(e_1)), e_1)[0:1]$. This will be part of the set of



(a) A visualization of transforming p_2 using ρ_{p_2}. The segment r_2, which is the concrete mode p_2, becomes the $x[0]$-axis and its destination waypoint $p_2.dest$ becomes the new origin. The original coordinate system and the one resulting from transforming r_0, or equivalently, p_0, using ρ_{p_0} are shaded along with the original scenario.



(b) The abstract (virtual) scenario. It consists of three segments of waypoints, all aligned with the $x[0]$-axis and ending at the origin. The (rotated) rectangles centered at the origin show the guards, projected to the $(x[0], x[1])$-plane, of the abstract edges. The rotated rectangle closest to the origin shows the abstract initial set. The further (rotated) rectangles show the abstract reset applied to the states in the abstract guards, i.e. the potential initial states of the robot following each abstract mode.



(c) The abstract automaton $\mathscr{B}_v$ modeling the scenario in Figure 3b, where $\forall i \in [0:4], \theta_i = \arctan_2(r_i.dest[1] - r_i.src[1], r_i.dest[0] - r_i.src[0])$.

Fig. 3: Constructing the abstract HA of Example 3.

possible reset states after transitioning over the virtual edge $e_{v,1}$ per Definition 5.(e). It is shown as the rectangle centered at $p_{v,2}.src$ in Figure 3b.

The illustration above for p_2 would be repeated for every $p \in P$, to construct the abstract HA shown in Figure 3c. The resulting abstract HA would be:

$$\mathscr{B}_v = \langle X_v, P_v, inv_v, E_v, guard_v, reset_v, f_v \rangle, \text{ where}$$

(a) $X_v = \mathbb{R}^3$ and $P_v = \{p_{v,0} := rv(r_0), p_{v,1} := rv(r_1), p_{v,2} := rv(r_2)\}$,
(b) $\forall i \in [3], inv_v(p_{v,i}) = \mathbb{R}^3 \backslash \gamma_{p_i}(B(p_{v,i}.dest, \varepsilon_{inv}) \times \mathbb{R})$,
(c) $E_v = \{e_{v,0} := (p_{v,0}, p_{v,1}), e_{v,1} := (p_{v,1}, p_{v,2}), e_{v,2} := (p_{v,2}, p_{v,1})\}$,

(d) $\forall i \in [3]$, let $p_{v,i} = e_{v,i}.src$, then

$$guard_v(e_{v,i}) = \begin{cases} \gamma_{p_i}(B(p_{v,i}.dest, \varepsilon_0) \times \mathbb{R}), \text{ if } i = 0, \\ \gamma_{p_i}(B(p_{v,i}.dest, \varepsilon_1) \times \mathbb{R}), \text{ if } i \in \{1,2\}, \end{cases}$$

(e) $\forall x_v \in X_v$,

$$reset_v(x_v, e_{v,i}) = \begin{cases} \{\gamma_{p_1}(\gamma_{p_0}^{-1}(x_v))\}, \text{ if } i = 0, \\ \{\gamma_{p_2}(\gamma_{p_1}^{-1}(x_v)), \gamma_{p_4}(\gamma_{p_3}^{-1}(x_v))\}, \text{ if } i = 1, \\ \{\gamma_{p_3}(\gamma_{p_2}^{-1}(x_v)), \gamma_{p_1}(\gamma_{p_4}^{-1}(x_v))\}, \text{ if } i = 2, \end{cases}$$

(f) $\forall x_v \in \mathbb{R}^3, \forall p_v \in \mathbb{R}^4, f_v(x_v, p_v) = f(x_v, p_v)$.

The resets of the guards of the three edges in E_v constitute the set of all possible reseted states. They are shown as the rectangles on the $x[0]$-axis, but not at the origin, in Figure 3b.

The abstract HA $\mathscr{B}_v$ has three modes and three edges versus the five modes and five edges of $\mathscr{B}$. In Figure 3b, the guards and reseted guards are overlapping, as well as the (not shown) invariants. This suggests that the reach set of $\mathscr{B}_v$ has a smaller volume than that of $\mathscr{B}$. That in turn means that the reach sets or reachtubes computations would be generally easier and faster for $\mathscr{B}_v$ than that for $\mathscr{B}$.

B. Forward simulation relation (FSR)

In this section, we establish a correspondence from the executions of the concrete HA to those of the abstract one through a FSR [37], [41], [42]. A FSR is a standard approach to relate the executions of two different HA.

Definition 6 (FSR [37]) A *forward simulation relation* from a HA $\mathscr{A}_1$ to another one $\mathscr{A}_2$, is a relation $\mathscr{R} \subseteq (X_1 \times P_1) \times (X_2 \times P_2)$, such that

(a) for any state $x_1 \in X_1$ and mode $p_1 \in P_1$, there exist a state $x_2 \in X_2$ and mode $p_2 \in P_2$, such that $(x_1, p_1)\mathscr{R}(x_2, p_2)$,
(b) for any discrete transition $(x_1, p_1) \to (x_1', p_1')$ of $\mathscr{A}_1$ and $(x_2, p_2) \in X_2 \times P_2$, where $(x_1, p_1)\mathscr{R}(x_2, p_2)$, there exists $(x_2', p_2') \in X_2 \times P_2$ such that $(x_2, p_2) \to (x_2', p_2')$ is a discrete transition of $\mathscr{A}_2$ and $(x_1', p_1')\mathscr{R}(x_2', p_2')$, and
(c) for any trajectory $\xi_1(x_1, p_1, \cdot)$ of $\mathscr{A}_1$ and pair $(x_2, p_2) \in X_2 \times P_2$, such that $(x_1, p_1)\mathscr{R}(x_2, p_2)$, there exists a trajectory $\xi_2(x_2, p_2, \cdot)$, where $dur(\xi_1) = dur(\xi_2)$ and $(\xi_1.lstate, p_1)\mathscr{R}(\xi_2.lstate, p_2)$.

Existence of a FSR implies that for any execution of $\mathscr{A}_1$ there is a corresponding related execution of $\mathscr{A}_2$. The following theorem is an adoption of Corollary 4.23 of [37] into our hybrid modeling framework.

Theorem 2 (Executions correspondence [37]) If there exists a FSR $\mathscr{R}$ from $\mathscr{A}_1$ to $\mathscr{A}_2$, then for every execution σ_1 of $\mathscr{A}_1$, there exists a corresponding execution σ_2 of $\mathscr{A}_2$ such that

(a) $\sigma_1.len = \sigma_2.len$,
(b) $\forall i \in [\sigma_1.len]$, $dur(\xi_{1,i}) = dur(\xi_{2,i})$, and
(c) $\forall i \in [\sigma_1.len]$, $(\xi_{1,i}.lstate, p_{1,i})\mathscr{R}(\xi_{2,i}.lstate, p_{2,i})$.

Now we introduce a FSR from the concrete hybrid automaton to the abstract one of Definition 5.

Theorem 3 (FSR: concrete to virtual) Consider the relation $\mathscr{R}_{rv} \subseteq (X \times P) \times (X_v \times P_v)$ defined as $(x, p)\mathscr{R}_{rv}(x_v, p_v)$ iff $x_v = \gamma_p(x)$ and $p_v = rv(p)$. Then, $\mathscr{R}_{rv}$ is a FSR from $\mathscr{A}$ to $\mathscr{A}_v$.

Proof: $\mathscr{R}_{rv}$ satisfies Definition 6.(a) since for any $x \in X$ and mode $p \in P$, $\gamma_p(x) \in X_v$, and $p_v = rv(p) \in P_v$, by Definition 5.(a).

To prove that $\mathscr{R}_{rv}$ satisfies Definition 6.(b), fix a discrete transition $(x, p) \to (x', p')$ of $\mathscr{A}$ and $(x_v, p_v) \in X_v \times P_v$ such that $(x, p, x_v, p_v) \in \mathscr{R}_{rv}$. We will show that if we choose $x_v' = \gamma_{p'}(x_v)$ and $p_v' = rv(p')$, then $(x_v', p_v') \in X_v \times P_v$, $(x', p', x_v', p_v') \in \mathscr{R}_{rv}$, and $(x_v, p_v) \to (x_v', p_v')$ is a valid discrete transition of $\mathscr{A}_v$.

First, $x_v' \in X_v$ and $p_v' \in P_v$, by Definition 5.(a). Second, $(x', p', x_v', p_v') \in \mathscr{R}_{rv}$ since $x_v' = \gamma_{p'}(x')$ and $p_v' = rv(p')$. Third, fix $e = (p, p')$. Then, by the definition of discrete transitions of $\mathscr{A}$, $x \in guard(e)$ and $x' \in reset(x, e)$. Also, from the definition of E_v in Definition 5.(c), the edge $e_v = (p_v, p_v') \in E_v$. By the definition of $\mathscr{R}_{rv}$ and the assumption that x and x_v are related under $\mathscr{R}_{rv}$, $x_v = \gamma_p(x)$. That means that $x_v \in \gamma_p(guard(e))$, since $x \in guard(e)$. But, by Definition 5.(d), $\gamma_p(guard(e)) \subseteq guard_v(e_v)$. Then, $x_v \in guard_v(e_v)$. Moreover, since $x' \in reset(x, e))$ and $x = \gamma_p^{-1}(x_v)$, then $x' \in reset(\gamma_p^{-1}(x_v), e)$. Hence, $x_v' = \gamma_{p'}(x') \in \gamma_{p'}(reset(\gamma_p^{-1}(x_v), e))$. Using Definition 5.(e), we know that $\gamma_{p'}(reset(\gamma_p^{-1}(x_v), e)) \subseteq reset_v(x_v, e_v)$. We have $x_v' \in reset_v(x_v, e_v)$. Therefore, $(x_v, p_v) \to (x_v', p_v')$ is a valid discrete transition of $\mathscr{A}_v$.

To prove that $\mathscr{R}_{rv}$ satisfies Definition 6.(c), fix a solution $\xi(x, p, \cdot)$ of $\mathscr{A}$ and a pair $(x_v, p_v) \in X_v \times P_v$, such that $(x, p, x_v, p_v) \in \mathscr{R}_{rv}$. Then, we will show that $dur(\xi) = dur(\xi_v)$ and $(\xi(x, p, dur(\xi)), p,\ \xi_v(x_v, p_v, dur(\xi)), p_v) \in \mathscr{R}_{rv}$. Since x and x_v are related under $\mathscr{R}_{rv}$, then $x_v = \gamma_p(x)$. Moreover, using Theorem 1, $\forall t \in dur(\xi)$, $\xi(\gamma_p(x), \rho_p(p), t) = \gamma_p(\xi(x, p, t))$. But, $rv(p) = p_v$ and using Definition 5.(f), $\xi(\gamma_p(x), \rho_p(p), \cdot) = \xi_v(\gamma_p(x), p_v, \cdot)$, which is a solution of $\mathscr{A}_v$ starting from $\gamma_p(x) = x_v$. Since $\forall t \in [0, dur(\xi)]$, $\xi(x, p, t) \in inv(p)$, then $\gamma_p(\xi(x, p, t)) \in \gamma_p(inv(p))$. Since $\xi_v(x_v, p_v, t) = \gamma_p(\xi(x, p, t))$ and $\gamma_p(inv(p)) \subseteq inv_v(p_v)$, then $\forall t \in [0, dur(\xi)]$, $\xi_v(x_v, p_v, t) \in inv_v(p_v)$, and thus ξ_v is valid as $\mathscr{A}_v$ does not have to transition or terminate because of an invariant violation. In addition, from the assumption that the guards are non-urgent, we can choose ξ_v that does not transition before $dur(\xi)$. Therefore, $\forall t \in dur(\xi)$, $(\xi(x, p, t), p,\ \xi_v(x_v, p_v, t), p_v) \in \mathscr{R}_{rv}$. ■

It is worth noting that there may not be a forward simulation relation from $\mathscr{A}_v$ to $\mathscr{A}$. The guard and reset of an edge e_v of $\mathscr{A}_v$ are the union of all the transformed versions of the guards and resets of the edges of $\mathscr{A}$ that get mapped to e_v. Hence, some discrete transitions in $\mathscr{A}_v$ may not have corresponding ones in $\mathscr{A}$. For example, consider two edges $e_1 = (p_{11}, p_{12})$ and $e_2 = (p_{21}, p_{22})$ of $\mathscr{A}$ with $rv(e_1) = rv(e_2) = e_v = (p_{v1}, p_{v2})$, an edge of $\mathscr{A}_v$. Then, a transition over e_v would be allowed in $\mathscr{A}_v$ with reseted state being $\gamma_{p_{22}}(reset(x_v, e_2))$ if $x_v \in \gamma_{p_{11}}(guard(e_1))$. Such a transition may not have a correspondent one in $\mathscr{A}$, since it resembles a transition from p_{11} to p_{22}. Similarly, the invariant of an abstract mode p_v is the union of the transformed versions of the invariants of the concrete modes mapped to it, which might allow trajectories in the abstract automaton with no corresponding concrete trajectories. Thus, some executions of $\mathscr{A}_v$ may not have corresponding executions in $\mathscr{A}$. This might lead to a scenario where $\mathscr{A}_v$ might not satisfy a property because of a spurious counter-example.

Example 4: We used SceneChecker [30] to verify the safety of the automaton $\mathscr{B}$ of Example 1 against an unsafe set of

states with and without using the symmetry-based abstraction $\mathscr{B}_v$ of Example 3. We used DryVR [43] as the reachability engine in SceneChecker. The results are shown in Figure 4. Instead of the state-based mode invariants of Example 1, for simplicity, we used time-based ones, limiting the time spent in p_0 to 3 seconds, p_1 and p_3 to 8 seconds, and p_2 and p_4 to 7 seconds. The unsafe set of states is the hyper-rectangle $U = [[-1,4,-\infty];[1,6,\infty]]$ and the initial set is $X_{init} = [[-5.1,-1.1,-0.1];[-4.9,-0.9,0.1]]$. The waypoints are located at $[-3,-2]$, $[3,-2]$, $[3,2]$, and $[-3,2]$. We set $v = 3$, $L = 2$, $\alpha_l = -\pi/4$, $\alpha_r = \pi/4$, $\varepsilon_0 = [0.5,0.4,\pi]$, $\varepsilon_1 = [0.2,0.2,\pi]$, and $\varepsilon_2 = [0.1,0.1,0.1]$. To verify the safety of $\mathscr{B}_v$, SceneChecker transforms the concrete unsafe set to corresponding unsafe sets for the abstract modes. Such a set for an abstract mode p_v would be $U_v(p_v) = \cup_{p \in r_v^{-1}(p_v)} \gamma_p(U)$. They are shown as the polygons with the red edges in Figure 4b. SceneChecker then computes the reachable set of $\mathscr{B}_v$ by calling DryVR to compute per-mode reachable sets and intersect them with the guards to compute the per-mode initial sets of states. For more details on SceneChecker, check [30]. Then, it checks for each abstract mode p_v, if its reachable set intersects $U_v(p_v)$. If none of the per-mode abstract reachable sets intersect their unsafe sets, then $\mathscr{B}_v$ is safe. Moreover, as a consequence of the FSR defined earlier, $\mathscr{B}$ is safe as well. SceneChecker had to call DryVR six times to compute per-mode reachable sets before reaching a fixed point in the reachabililty computation of $\mathscr{B}$ and verifying its safety directly (without using our abstraction), taking a total of 158 ms of computation time. In contrast, it had to call DryVR only four times to verify the safety of $\mathscr{B}_v$ (with refinements disabled), and that of $\mathscr{B}$ as a consequence, taking a total of 96 ms. The improvement in computation time increases with the number of symmetric modes, as demonstrated in [30].

V. Conclusion

We presented the first symmetry-based abstractions of hybrid automata. Our abstractions create automata with fewer number of modes and edges than the concrete ones by aggregating sets of modes into individual ones. Symmetry maps transform trajectories of a concrete mode to ones of its representative abstract one, and vice versa. We showed a forward simulation relation that proves the soundness of our abstraction (Theorem 3). We provided a running example to illustrate our approach. In our previous paper [30], we implemented the abstraction with a simple refinement algorithm to accelerate the safety verification of vehicles navigating complex environments. We presented a CEGAR algorithm based on our abstraction in [44].

References

[1] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012. [Online]. Available: http://www.jstor.org/stable/j.ctt7s02z

[2] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*. London: Springer, 2000.

[3] R. Alur, *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

[4] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Journal of Computer and System Sciences*, vol. 57, pp. 94–124, 1998.

[5] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas, "Discrete abstractions of hybrid systems," in *Proceedings of the IEEE*, 2000.

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[7] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. E. Dullerud, "STORMED hybrid systems," in *Automata, Languages and Programming, ICALP 2008*, ser. LNCS, vol. 5126. Springer, 2008, pp. 136–147.

[8] G. Lafferriere, G. J. Pappas, and S. Sastry, "O-minimal hybrid systems," *Mathematics of control, signals and systems*, vol. 13, no. 1, pp. 1–21, 2000.

[9] A. Girard, "Controller synthesis for safety and reachability via approximate bisimulation," *Automatica*, vol. 48, no. 5, pp. 947 – 953, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510981200088X

[10] P. Tabuada, G. J. Pappas, and P. U. Lima, "Composing abstractions of hybrid systems." in *HSCC 2002*, ser. LNCS, C. Tomlin and M. R. Greenstreet, Eds., vol. 2289. Springer, 2002, pp. 436–450.

[11] P. Tabuada and G. J. Pappas, "Hybrid abstractions that preserve timed languages," in *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control*, ser. HSCC '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 501–514.

[12] A. Fehnker, E. M. Clarke, S. K. Jha, and B. H. Krogh, "Refining abstractions of hybrid systems using counterexample fragments," in *HSCC'2005*, ser. Lecture Notes in Computer Science, M. Morari and L. Thiele, Eds., vol. 3414. Springer, 2005, pp. 242–257.

[13] R. Alur, T. Dang, and F. Ivančić, "Counterexample-guided predicate abstraction of hybrid systems," *Theoretical Computer Science*, vol. 354, no. 2, pp. 250–271, 2006.

[14] N. Roohi, P. Prabhakar, and M. Viswanathan, "HARE: A hybrid abstraction refinement engine for verifying non-linear hybrid automata," in *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017*, 2017, pp. 573–588. [Online]. Available: https://doi.org/10.1007/978-3-662-54577-5_33

[15] P. Prabhakar, P. S. Duggirala, S. Mitra, and M. Viswanathan, "Hybrid automata-based CEGAR for rectangular hybrid systems," *Formal Methods in System Design*, vol. 46, no. 2, pp. 105–134, 2015. [Online]. Available: https://doi.org/10.1007/s10703-015-0225-4

[16] S. Sankaranarayanan, "Change-of-bases abstractions for non-linear hybrid systems," *Nonlinear Analysis: Hybrid Systems*, vol. 19, pp. 107 – 133, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1751570X15000540

[17] K. Hsu, R. Majumdar, K. Mallik, and A.-K. Schmuck, "Lazy abstraction-based control for safety specifications," in *57th IEEE Conference on Decision and Control, CDC 2018, Miami, FL, USA, December 17-19, 2018*, 2018, pp. 4902–4907. [Online]. Available: https://doi.org/10.1109/CDC.2018.8619659

[18] T. Dang, O. Maler, and R. Testylier, "Accurate hybridization of nonlinear systems," 01 2010, pp. 11–20.

[19] P. Mehta, G. Hagen, and A. Banaszuk, "Symmetry and symmetry-breaking for a wave equation with feedback," *SIAM J. Applied Dynamical Systems*, vol. 6, pp. 549–575, 01 2007.

[20] S. Bonnabel, P. Martin, and P. Rouchon, "Symmetry-preserving observers," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2514–2526, 2008.

[21] M. W. Spong and F. Bullo, "Controlled symmetries and passive walking," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 1025–1031, July 2005.

[22] J. Maidens, A. Barrau, S. Bonnabel, and M. Arcak, "Symmetry reduction for dynamic programming," *Automatica*, vol. 97, pp. 367–375, 2018.

[23] H. Sibai, N. Mokhlesi, and S. Mitra, "Using symmetry transformations in equivariant dynamical systems for their safety verification," in *Automated Technology for Verification and Analysis*, 2019, pp. 1–17.

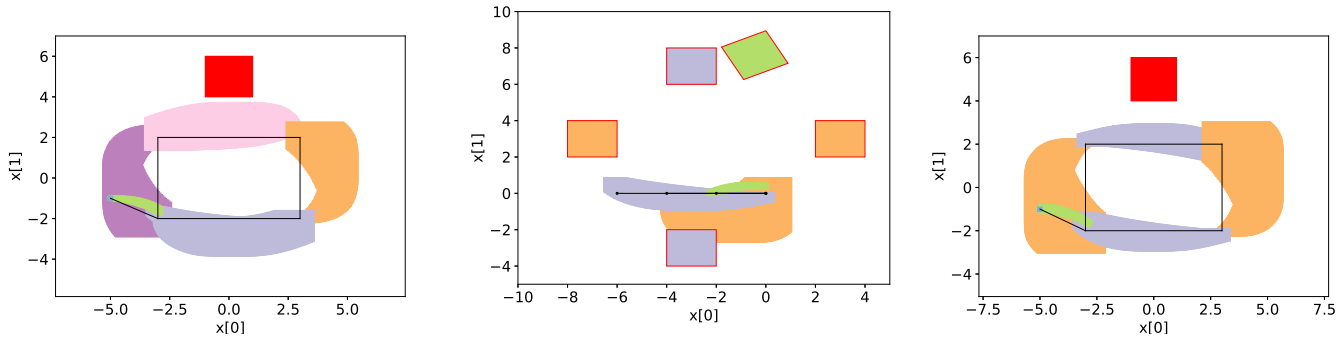
[24] H. Sibai, N. Mokhlesi, C. Fan, and S. Mitra, "Multi-agent safety verification using symmetry transformations," in *Tools and Algorithms for the Construction and Analysis of Systems*, A. Biere and D. Parker, Eds. Cham: Springer International Publishing, 2020, pp. 173–190.

[25] S. Mitra and H. Sibai, "Symmetry for boosting algorithmic proofs of cyberphysical systems," *Computer*, vol. 55, no. 10, pp. 88–93, 2022.

[26] J. Damers, L. Jaulin, and S. Rohou, "Lie symmetries applied to interval integration," *Automatica*, vol. 144, p. 110502, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109822003612

[27] L. G'erard and J.-J. E. Slotine, "Neuronal networks and controlled symmetries, a generic framework," 2006.

[28] J. Hanc, S. Tuleja, and M. Hancova, "Symmetries and conservation laws: Consequences of noether's theorem," *American Journal of Physics - AMER J PHYS*, vol. 72, pp. 428–435, 04 2004.



(a) Reachable set of the concrete automaton

(b) Reachable set of the abstract automaton

(c) Reachable set of the concrete automaton obtained through transforming those of the abstract one in Figure 4b.

Fig. 4: Reachable sets computed using SceneChecker for the automata in Examples 1 and 3. Each color corresponds to an abstract mode, with non-polgyon shapes representing reachable sets. Each reachable set has a different color in Figure 4a since each abstract mode represents a single concrete mode as no abstraction is being done. Red rectangles in Figures 4a and 4c represent the unsafe set, while the small blue rectangles represent the initial set. Polygons with red edges in Figure 4b represent the unsafe set in Figure 4a transformed to the relative coordinates defined by the segments mapped to the same abstract mode as their filling color. Black segments are the segments between waypoints.

[29] E. Noether, "Invarianten beliebiger differentialausdrücke," *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, vol. 1918, pp. 37–44, 1918. [Online]. Available: http://eudml.org/doc/59011

[30] H. Sibai, Y. Li, and S. Mitra, "Scenechecker: Boosting scenario verification using symmetry abstractions," in *Computer Aided Verification*, A. Silva and K. R. M. Leino, Eds. Cham: Springer International Publishing, 2021, pp. 580–594.

[31] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid I/O automata," *Information and Computation*, vol. 185, no. 1, pp. 105–157, August 2003.

[32] R. Alur, C. C. T. A. Henzinger, and P. H. Ho., "Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems," in *Hybrid Systems*, ser. LNCS, R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, Eds., vol. 736. Springer-Verlag, 1993, pp. 209–229.

[33] L. Doyen, T. A. Henzinger, and J.-F. Raskin, "Automatic rectangular refinement of affine hybrid systems," in *Proceedings of the Third International Conference on Formal Modeling and Analysis of Timed Systems*, ser. FORMATS'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 144–161. [Online]. Available: https://doi.org/10.1007/11603009_13

[34] S. K. Jha, B. H. Krogh, J. E. Weimer, and E. M. Clarke, "Reachability for linear hybrid automata using iterative relaxation abstraction," in *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC'07. Berlin, Heidelberg: Springer-Verlag, 2007, p. 287–300.

[35] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. Pasareanu, A. Podelski, and T. Strump, "Assume-guarantee abstraction refinement meets hybrid systems," in *Hardware and Software: Verification and Testing*, E. Yahav, Ed. Cham: Springer International Publishing, 2014, pp. 116–131.

[36] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski, "Multiple shooting, cegar-based falsification for hybrid systems," in *Proceedings of the 14th International Conference on Embedded Software*, ser. EMSOFT '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2656045.2656061

[37] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, *The Theory of Timed I/O Automata*, ser. Synthesis Lectures on Computer Science. Morgan Claypool, November 2005, also available as Technical Report MIT-LCS-TR-917, MIT.

[38] B. Paden, M. Čáp, S. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[39] G. Russo and J.-J. E. Slotine, "Symmetries, stability, and control in nonlinear systems and networks," *Physical Review E*, vol. 84, no. 4, p. 041929, 2011.

[40] E. J. Cartan and J.-L. Leray, "La théorie des groupes finis et continus et la géométrie différentielle traitées par la méthode du repère mobile : leçons professées à la sorbonne," 1937.

[41] S. Mitra, "A verification framework for hybrid systems," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007. [Online]. Available: http://people.csail.mit.edu/mitras/research/thesis.pdf

[42] A. Girard, A. A. Julius, and G. J. Pappas, "Approximate simulation relations for hybrid systems," in *IFAC Analysis and Design of Hybrid Systems*, Alghero, Italy, June 2006.

[43] C. Fan, B. Qi, S. Mitra, and M. Viswanathan, "Data-driven verification and compositional reasoning for automotive systems," in *Computer Aided Verification*. Springer International Publishing, 2017, pp. 441–461.

[44] H. Sibai, "Accelerating cerification of cyber-physical systems using symmetry," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2021.