# Verification of Continuous Dynamical Systems by Timed Automata

Sloth, Christoffer; Wisniewski, Rafael

*Published in:*
Formal Methods in System Design

*DOI (link to publication from Publisher):*
10.1007/s10703-011-0118-0

*Publication date:*
2011

*Document Version*
Publisher's PDF, also known as Version of record

Form Methods Syst Des
DOI 10.1007/s10703-011-0118-0

# Verification of continuous dynamical systems by timed automata

**Christoffer Sloth · Rafael Wisniewski**



**Abstract** This paper presents a method for abstracting continuous dynamical systems by timed automata. The abstraction is based on partitioning the state space of a dynamical system using positive invariant sets, which form cells that represent locations of a timed automaton. The abstraction is intended to enable formal verification of temporal properties of dynamical systems without simulating any system trajectory, which is currently not possible. Therefore, conditions for obtaining sound, complete, and refinable abstractions are set up.

The novelty of the method is the partitioning of the state space, which is generated utilizing sub-level sets of Lyapunov functions, as they are positive invariant sets. It is shown that this partition generates sound and complete abstractions. Furthermore, the complete abstractions can be composed of multiple timed automata, allowing parallelization of the verification process. The proposed abstraction is applied to two examples, which illustrate how sound and complete abstractions are generated and the type of specification we can check. Finally, an example shows how the compositionality of the abstraction can be used to analyze a high-dimensional system.

**Keywords** Timed automata · Verification · Reachability · Lyapunov functions

## 1 Introduction

The verification of properties such as safety is important for any system. Such verification involves reachability calculations or approximations. The reachable sets of continuous and

This work was supported by MT-LAB, a VKR Centre of Excellence.

C. Sloth (✉)
Department of Computer Science, Aalborg University, Aalborg, Denmark
e-mail: csloth@cs.aau.dk

R. Wisniewski
Section for Automation & Control, Aalborg University, Aalborg, Denmark
e-mail: raf@es.aau.dk

Published online: 29 April 2011

hybrid systems are in general incomputable [6]. Therefore, much research effort has been spent on the approximation of especially reachable sets for continuous systems [16]. Yet, reachability is decidable for discrete systems such as automata and timed automata; consequently, there exists a rich set of tools aimed at verifying properties of such systems. Therefore, abstracting dynamical systems by discrete systems would enable the verification of dynamical systems using the tools for discrete systems.

There are basically two methods for verifying continuous and hybrid systems. The first is to over-approximate the reachable states by convex sets as in [14, 20, 31]. The second method is to abstract the original system by a system of reduced complexity. Both methods rely on reach set computations, which according to [16] limits the capabilities of automatic analysis, due to their complexity. An abstraction method for continuous systems is presented in [21], and an abstraction method for hybrid systems is presented in [27]. The models used in these methods are called symbolic models if equivalence classes of states are used instead of individual states [1]; for more insight in symbolic dynamics see [24]. To reduce the computational effort, the verification process is often accomplished by first choosing a coarse partitioning and then refining it until the system can be verified. One such algorithm is proposed in [13], where piecewise affine systems with real eigenvalues are considered.

The goal of this paper is to abstract continuous systems by timed automata, since efficient tools such as UPPAAL can verify this type of models [7]. The verification of temporal requirements with a bounded time horizon is well known and studied in computer science, via checking if some model satisfies, e.g., a Timed Computation Tree Logic (TCTL) specification [2], but not in control theory. In control theory almost all requirements are related to convergence, i.e., system properties when time goes to infinity. This implies that the formal verification of temporal requirements of dynamical systems would replace the need for simulations to verify the transient behavior of the systems. Hence, the proposed method has relevance for even the most simple dynamical system models.

The concept of the abstraction is inspired by [21], where slices are introduced to improve the accuracy of abstractions of continuous systems. In short, the aim of [21] is to abstract autonomous continuous systems by timed automata via partitioning their state spaces into cubes along the coordinate axis as shown in Fig. 1. As a result, each cube is associated with a discrete location of the timed automaton. Slices are used in addition to the cells in the generation of the abstraction, to increase the precision of the abstraction. A slice is a collection of cells, as illustrated in Fig. 1 by the collection of shaded cells.

Previous abstraction techniques are based on over-approximations, where the upper bound on the error in general is unknown. This implies that the quality of the abstraction is unknown and that falsification of safety is not possible unless refinements or under-approximations are considered. However, this is not considered in this paper.

In this paper, which is an extension of [26], continuous systems are abstracted by timed automata by considering both cells and slices for generating the abstractions. A new partitioning is proposed, where the functions used for the partitioning are chosen in accordance with the vector field of the dynamical system. By this, we mean that the vector field cannot be tangent to a boundary of a cell. This requirement is not satisfied in the partition illustrated in Fig. 1, as some of the blue arrows (illustration of the vector field) are tangent to the boundaries of cells. This approach is different from most previous work on abstractions of continuous systems; however, this may reduce the size of the symbolic models, which is currently the focus of other research [15]. Additionally, model checking can be done in a compositional manner using the proposed method, due to the use of slices. This may reduce the computational complexity for high-dimensional systems. Our objective is to show that by partitioning the state space in accordance with the vector field of the dynamical

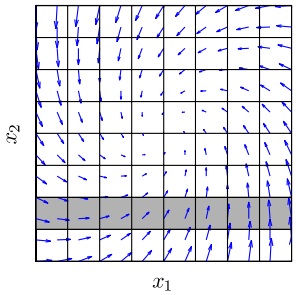


**Fig. 1** The figure illustrates a vector field of a dynamical system (*arrows*) and a partition of its state space into cubes. The *shaded cubes* represent a slice

system, it is possible to prove safety of Morse-Smale systems using the proposed abstraction and additionally to falsify safety for linear systems. Remark that falsification of safety properties may also be possible by, e.g., refining the abstraction or doing analysis based on under-approximations. Furthermore, for linear systems, it is possible to calculate an a priori upper bound of the size of the over-approximation of the reachable set and reduce this upper bound to an arbitrary small value, by refining the partitioning. Hence, we can obtain an abstraction with arbitrary precision of the reachable set. In conclusion, the following problem is formulated.

**Problem 1** Given an autonomous dynamical system, find a partition of its state space, which allows over-approximation with arbitrary accuracy of its reachable set by a timed automaton.

To ease the flow of the article, some definitions and the proofs of the presented propositions are located in Appendices A and B.

This paper is organized as follows. Section 2 contains preliminary definitions utilized throughout the paper. Then the general idea of the abstraction is explained in Sect. 3 and Sect. 4. First, we determine how a partition of a state space can be generated by positive invariant sets, and then we show how a timed automaton is generated from this partition. In Sect. 5, properties of the generated timed automaton are derived. In Sect. 6, conditions for the partition are deduced, and LMI conditions for their satisfaction are provided. Afterwards, a method for synthesizing such a partition is proposed in Sect. 7. Examples are provided in Sect. 8; and Sect. 9 comprises conclusions.

### 1.1 Notation

The set $\{1, \ldots, k\}$ is denoted $\boldsymbol{k}$. $B^A$ is the set of maps $A \to B$. The power set of $A$ is denoted $2^A$. Given a vector $a \in \mathbb{R}^n$, $a(j)$ denotes the $j$th coordinate of $a$. Given a set $A$, the cardinality of the set is denoted $|A|$. We consider the Euclidean space $(\mathbb{R}^n, \langle , \rangle)$, where $\langle , \rangle$ is the scalar product. Whenever $f : X \to \mathbb{R}$ is a function and $a \in \mathbb{R}$, we write $f^{-1}(a)$ to shorten the notation of $f^{-1}(\{a\})$. $\mathfrak{X}^r(M)$ is the space of $C^r$ vector fields.

## 2 Preliminaries

The purpose of this section is to provide definitions related to dynamical systems and timed automata. Especially, we give a detailed description of the considered class of dynamical systems.

### 2.1 Dynamical systems

This subsection provides a definition of an autonomous dynamical system and its solution. This is followed by definitions of reachable set and Lyapunov function used for the partitioning. Finally, explanations of the considered class of systems are provided.

**Definition 1** (Autonomous dynamical system) An autonomous dynamical system $\Gamma = (X, f)$, with state space $X \subseteq \mathbb{R}^n$ and $f : X \to \mathbb{R}^n$ a continuous map, has dynamics described by ordinary differential equations

$$\dot{x} = f(x). \tag{1}$$

Let $\phi_\Gamma : [0, \epsilon] \times X_0 \to X$, $\epsilon > 0$ be the flow map satisfying

$$\frac{d\phi_\Gamma(t, x_0)}{dt} = f\left(\phi_\Gamma(t, x_0)\right) \tag{2}$$

for all $t \in [0, \epsilon]$. In other words, $\phi_\Gamma(t, x_0)$ is the solution of (1), from an initial state $x_0 \in X_0 \subseteq X$ for $t \in [0, \epsilon]$.

It is assumed that (1) has a solution for each $x_0 \in X_0$ and for all time $t \in \mathbb{R}$, and that this solution is unique. Hence, the function $f$ is continuous, locally Lipschitz, and has linear growth [11].

The reachable set of a dynamical system is defined in the following.

**Definition 2** (Reachable set of dynamical system) The reachable states of a system $\Gamma$ from a set of initial states $X_0 \subseteq X$ on the time interval $[t_1, t_2]$ is defined as

$$\text{Reach}_{[t_1,t_2]}(\Gamma, X_0) = \{x \in X | \exists t \in [t_1, t_2], \exists x_0 \in X_0, \text{such that } x = \phi_\Gamma(t, x_0)\} \tag{3}$$

We define Lyapunov function in the following, as these are used in the partitioning [23].

**Definition 3** (Lyapunov function) Let $X$ be an open connected subset of $\mathbb{R}^n$. Suppose $f : X \to \mathbb{R}^n$ is continuous and let $\text{Cr}(f)$ be the set of critical points of $f$. Then a real non-degenerate (see [22, p. 1]) differentiable function $\varphi : X \to \mathbb{R}$ is said to be a Lyapunov function for $f$ if

$$p \text{ is a critical point of } f \quad \Leftrightarrow \quad p \text{ is a critical point of } \varphi,$$

$$\dot{\varphi}(x) \equiv \sum_{j=1}^{n} \frac{\partial \varphi}{\partial x_j}(x) f^j(x), \tag{4a}$$

$$\dot{\varphi}(x) = 0 \quad \forall x \in \text{Cr}(f), \tag{4b}$$

$$\dot{\varphi}(x) < 0 \quad \forall x \in X \backslash \text{Cr}(f), \tag{4c}$$

and there exists $\alpha > 0$ and an open neighborhood of each critical point $p \in \text{Cr}(f)$, where

$$\|\dot{\varphi}(x)\| \geq \alpha \|x - p\|^2. \tag{5}$$

Notice that we only require the vector field to be transversal to the level curves of a Lyapunov function $\varphi$, i.e., $\dot{\varphi}(x) = \langle \nabla \varphi(x), f(x) \rangle < 0$ for all $x \in X \backslash \text{Cr}(f)$, and does not use Lyapunov

functions in the usual sense, where the existence of a Lyapunov function implies stability, but uses a more general notion from [23]. Assume that a Lyapunov function $\varphi(x)$ is positive definite, then its sub-level sets generate positive invariant sets.

**Definition 4** (Positive invariant set) Given a system $\Gamma = (X, f)$, a set $\mathcal{X} \subseteq X$ is said to be positively invariant if for all $x_0 \in \mathcal{X}$ and for all $t \geq 0$

$$\phi_\Gamma(t, x_0) \in \mathcal{X}. \tag{6}$$

*2.1.1 Considered class of systems*

In this subsection we describe the considered class of systems and motivate why this class of systems is sufficient for this work.

The most general class of systems considered in this work is Morse-Smale systems, defined formally in Definition 32. However, some of the results only apply for linear systems. A linear system $\Gamma = (X, f)$ is a system with linear $f$, i.e.,

$$\dot{x} = Ax, \tag{7}$$

where $A$ is an $n \times n$ non-singular matrix.

To explain Morse-Smale systems we first explain topological equivalence of systems and structural stability of systems.

Two systems (or vector fields) are topological equivalent, see Definition 33, if there exists a correspondence between their solution trajectories, given by a continuous deformation (homeomorphism, i.e., a continuous bijection with continuous inverse). Hence, the exists a homeomorphism $h$ such that the solution of a vector field $\eta$ from some initial state can be described by a solution of a topologically equivalent vector field $\xi$ as $h \circ \phi_\xi(\mathbb{R}, x_0) = \phi_\eta(\mathbb{R}, h(x_0))$.

Next, we define structural stability of a vector field, which is an important property of Morse-Smale systems.

**Definition 5** (Structurally stable vector field [19]) A vector field $\xi \in \mathfrak{X}^r(M)$ is structurally stable if there exists a neighborhood $V$ of $\xi$ in $\mathfrak{X}^r(M)$ such that every $\eta \in V$ is topologically equivalent to $\xi$.

A vector field is structurally stable if its quantitative behavior does not change after the vector field has been slightly perturbed. In the following, we provide some intuition in structurally stable systems, by showing an example of a system that is not structurally stable.

Consider the linear system with purely complex eigenvalues $\{-i, i\}$, i.e., the real parts of the eigenvalues are zero

$$\dot{x} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} x. \tag{8}$$

Two trajectories of the system are drawn in the left subplot of Fig. 2. If we slightly perturb the system (this perturbation is given by a smooth map, see Theorem 2.1 in [17]), the eigenvalues of the system may become positive (trajectory in the middle subplot) or negative (trajectory in the right subplot). As a consequence, it is no longer possible to describe the solution trajectories of the middle or right subplot by a continuous deformation (homeomorphism) of solution trajectories of the left subplot. Therefore, the system shown in (8) is not structurally stable.

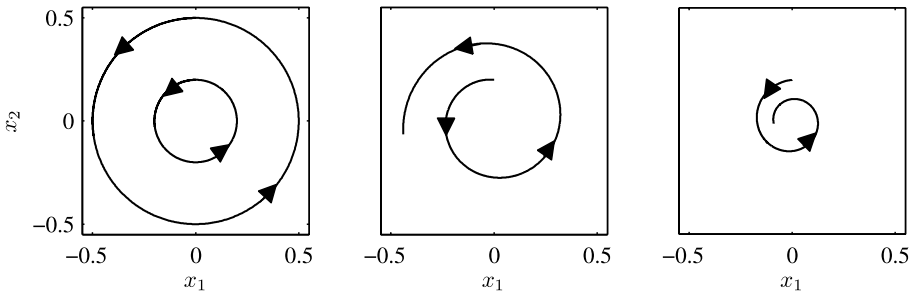


**Fig. 2** Trajectories of three dynamical systems

In relation to numerical simulation of systems that are not structurally stable, even the smallest rounding error in the representation of the system may significantly alter its behavior.

Two important restrictions of Morse-Smale systems are exploited in this paper. First, the vector field has a finite number of singular points, each hyperbolic. A *hyperbolic singular point* is a singular point such that in local coordinates the matrix of partial derivatives of the vector field has eigenvalues with nonzero real parts (the eigenvalues of (8) are not hyperbolic). This property allows the system to be split up into a stable and an unstable subsystem, which can be analyzed separately. Second, the stable and unstable manifolds associated with a singular point have transversal intersection. This restriction is necessary to obtain a structurally stable vector field.

*Remark 1* To the best of authors' knowledge, there exists no method to check if a system is Morse-Smale. However, if the system is second order and one finds a Lyapunov function, it is Morse-Smale. For a system of dimension greater than two, if there is a Lyapunov function then it can be approximated arbitrary closely by a Morse-Smale vector field [23]. For linear systems, it is necessary and sufficient to check if all eigenvalues are hyperbolic.

To summarize, it is chosen to only consider Morse-Smale systems in this paper, as they are structurally stable, and there exists an open set of Lyapunov functions for all Morse-Smale systems, which is important for the partitioning.

In the following, we reason about the size of the class of Morse-Smale systems (or vector fields).

Morse-Smale systems are dense in systems of less than or equal to two dimensions according to the following two theorems.

**Theorem 1** (See [25]) *In order that the vector field $\xi$ be structurally stable on the compact 2-dimensional manifold $M$ it is necessary and sufficient that the vector field is Morse-Smale.*

**Theorem 2** (See [25]) *The set of all structurally stable systems is open and dense in the space of all systems defined on a 2-dimensional manifold $M$.*

From Theorem 1 we conclude that Morse-Smale systems are robust in the sense that its behavior does not change dramatically after being perturbed slightly, as it is structurally stable (explained next). This also implies that the partitioning will be insensitive to small perturbations. This is a desirable property, as small perturbations always exist for physical

systems. From Theorem 2 we conclude that for two dimensions all dynamical systems can be approximated by the Morse-Smale system with arbitrary accuracy.

We are of course interested in studying systems having dimension greater than two; however, note that we want to verify control systems. Control systems are almost always designed using some Lyapunov-based method, i.e., there exists a Lyapunov function for the considered system, which implies that it is Morse-Smale. Therefore, we conclude that it is reasonable to only study Morse-Smale systems in this analysis.

### 2.2 Timed automata

We abstract dynamical systems by timed automata. Therefore, a definition of a timed automaton is provided in the following Alur et al. [4]. Before defining it, a set of diagonal-free clock constraints $\Psi(C)$ for the set $C$ of clocks is defined. $\Psi(C)$ contains all invariants and guards of the timed automaton; consequently, it is described by the following grammar

$$\psi ::= c \bowtie k | \psi_1 \wedge \psi_2, \tag{9a}$$

where

$$c \in C, \quad k \in \mathbb{R}_{\geq 0}, \quad \text{and} \quad \bowtie \in \{\leq, <, =, >, \geq\}. \tag{9b}$$

Note that the clock constraint $k$ should usually be a rational number, but in this paper, no effort is done to convert the clock constraints into rational numbers. However, any real number can be approximated by a rational number with an arbitrary small error $\epsilon > 0$.

**Definition 6** (Timed automaton) A timed automaton $\mathcal{A}$ is a tuple $(L, L_0, C, \Sigma, I, \Delta)$, where

- $L$ is a finite set of locations, and $L_0 \subseteq L$ is the set of initial locations.
- $C$ is a finite set of clocks.
- $\Sigma$ is the input alphabet.
- $I : L \to \Psi(C)$ assigns invariants to locations, where $\Psi(C)$ is the set of all clock constraints in (9).
- $\Delta \subseteq L \times \Psi(C) \times \Sigma \times 2^C \times L$ is a finite set of transition relations. A transition relation is a tuple $(l, G_{l \to l'}, \sigma, R_{l \to l'}, l')$ which assigns an edge between two locations, where $l$ is the source location, $l'$ is the destination location, $G_{l \to l'} \in \Psi(C)$ is the guard set, $\sigma$ is a symbol in the alphabet $\Sigma$, and $R_{l \to l'} \subseteq C$ is a subset of clocks.

The semantics of a timed automaton is defined in the following, adopting the notion of [12].

**Definition 7** (Clock valuation) A clock valuation on a set of clocks $C$ is a mapping $v : C \to \mathbb{R}_{\geq 0}$. The initial valuation $v_0$ is given by $v_0(c) = 0$ for all $c \in C$. For a valuation $v$, $d \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + d$ and $v[R]$ are defined as

$$(v + d)(c) = v(c) + d, \tag{10a}$$

$$v[R](c) = \begin{cases} 0 & \text{for } c \in R, \\ v(c) & \text{otherwise.} \end{cases} \tag{10b}$$

**Definition 8** (Semantics of clock constraint) A clock constraint in $\Psi(C)$ is a set of clock valuations $\{v : C \to \mathbb{R}_{\geq 0}\}$ given by

$$[\![c \bowtie k]\!] = \{v : C \to \mathbb{R}_{\geq 0} | v(c) \bowtie k\}, \tag{11a}$$

$$[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]. \tag{11b}$$

For convenience we denote $v \in [\![\psi]\!]$ by $v \models \psi$.

**Definition 9** (Semantics of timed automaton) The semantics of a timed automaton $\mathcal{A} = (L, L_0, C, \Sigma, I, \Delta)$ is the transition system $[\![\mathcal{A}]\!] = (S, S_0, \Sigma \cup \mathbb{R}_{\geq 0}, T_\mathrm{s} \cup T_\mathrm{d})$ given by

$$S = \{(l, v) \in L \times \mathbb{R}^C_{\geq 0} | v \models I(l)\},$$

$$S_0 = \{(l, v) \in L_0 \times 0^C\},$$

$$T_\mathrm{s} = \{(l, v) \xrightarrow{\sigma} (l', v') | \exists (l, G_{l \to l'}, \sigma, R_{l \to l'}, l') \in \Delta : v \models G_{l \to l'}, v' = v[R_{l \to l'}]\},$$

$$T_\mathrm{d} = \{(l, v) \xrightarrow{d} (l, v + d) | \forall d' \in [0, d] : v + d' \models I(l)\}.$$

Analog to the solution of (1), shown in (2), is a run of a timed automaton, which is defined in the following.

**Definition 10** (Run of timed automaton) A run of a timed automaton $\mathcal{A}$ is a possibly infinite sequence of alternations between time steps and discrete steps on the following form

$$(l_0, v_0) \xrightarrow{d_1} (l_0, v_1) \xrightarrow{\sigma_1} (l_1, v_2) \xrightarrow{d_2} \cdots, \tag{12}$$

where $d_i \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma$. The multifunction describing runs of a timed automaton $\phi_\mathcal{A} : \mathbb{R}_{\geq 0} \times L_0 \to 2^L$, defined by $l \in \phi_\mathcal{A}(t, l_0)$ if and only if there exists a path in $[\![\mathcal{A}]\!]$ initialized in $(l_0, v_0)$ that reaches the location $l$ at time $t = \sum_i d_i$.

We use the notation of the run of a timed automaton, to define its reachable locations.

**Definition 11** (Reachable set of timed automaton) The reachable set of a timed automaton $\mathcal{A}$ with initial locations $L_0$ on the time interval $[t_1, t_2]$ is defined by

$$\mathrm{Reach}_{[t_1, t_2]}(\mathcal{A}, L_0) = \{l \in L | \exists t \in [t_1, t_2], \exists l_0 \in L_0, \text{such that } l \in \phi_\mathcal{A}(t, l_0)\}. \tag{13}$$

This concludes the preliminary definitions. The next section explains how the state space of a dynamical system should be partitioned.

## 3 Generation of finite partition

The main idea of this work is to design an abstraction procedure, which exploits the knowledge of the flow of the dynamical system. Therefore, it is proposed to partition the state space into a finite number of cells by intersecting slices defined as the set-difference of positive and negative invariant sets. This ensures a unidirectional flow through the boundaries of the cells, see Definition 4. Another approach to partitioning the state space of a hybrid

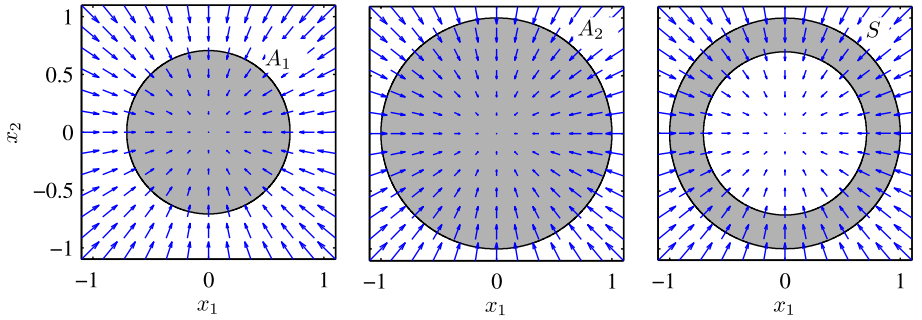


**Fig. 3** Illustration of a phase plot of a dynamical system (*arrows*), two positive invariant sets, $A_1$ and $A_2$ (*shaded disks*), and a slice $S = \mathrm{cl}(A_2 \backslash A_1)$ to the right

system by a family of functions is developed in [10]. In this work, the state space is partitioned by a family of immersed submanifolds, so called leafs of the foliation. Each leave is of codimension 1, and it is transversal to the studied vector field. Such a foliation can be generated as an inverse image of a regular value under a function. Consequently, a collection of foliations defines a partitioning. A regular point of a map $f$ is a point where the differential of $f$ is surjective. A point $v$, in the image of $f$, is a regular value if its pre-image contains regular points only (i.e. no critical points), see Definition 34.

In most previously proposed methods for abstracting dynamical systems, the partitioning of the state space is done without considering the dynamics of the system. In contrast to this, the proposed method is mainly concerned with the partitioning of the state space according to the system dynamics.

**Definition 12** (Slice) A nonempty set $S$ is a slice if there exist two open sets $A_1$ and $A_2$ such that

1. $A_1$ and $A_2$ are positively or negatively invariant,
2. $A_1$ is a proper subset of $A_2$, and
3. $S = \mathrm{cl}(A_2 \backslash A_1)$.

The slices are defined to be set-differences of positive or negative invariant sets to ensure that the vector field is transversal, see Definition 15, to the boundaries of the slices. Figure 3 illustrates a slice and the two positive invariant sets, $A_1$ and $A_2$, which generate it. This construction of slices ensures a nonzero minimum time for staying in each slice unless it contains an equilibrium point. We know from the definition of the Lyapunov function that there are no limit cycles in the set $A_2 \backslash A_1$. If there was a limit cycle and $a$ and $b$ were two points on it, then the system would first reach $a$, then $b$ and $a$ again. However, as the Lyapunov function is strictly decreasing it would imply that $\varphi(a) > \varphi(b) > \varphi(a)$, which cannot be true.

To devise a partition of a state space, we need to define finite collections of slices. These collections are called slice-families.

**Definition 13** (Slice-family) A slice-family $\mathcal{S}$ is a finite collection of slices generated by the positive or negative invariant open sets $A_1 \subset A_2 \subset \cdots \subset A_k$ covering the entire state space of the system $\Gamma = (X, f)$. Thereby, $S_1 = \mathrm{cl}(A_1)$, $S_2 = \mathrm{cl}(A_2 \backslash A_1), \ldots, S_k = \mathrm{cl}(A_k \backslash A_{k-1})$, and $X \subseteq A_k$.

For convenience $|\mathcal{S}|$ is defined to be the number of slices in the slice-family $\mathcal{S}$.

We say that the slice-family $\mathcal{S}$ is generated by the sets $\{A_i | i \in \boldsymbol{k}\}$. We address the existence and generation of these sets in Sect. 7.

A function is associated to each slice-family $\mathcal{S}$ to provide an easy way of describing the boundary of a slice. Such a function is called a partitioning function.

**Definition 14** (Partitioning function) Let $\mathcal{S}$ be a slice-family, then a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ smooth on $\mathbb{R}^n \backslash \mathrm{Cr}(f)$ is a partitioning function associated to $\mathcal{S}$ if for any positive or negative invariant set $A_i$ generating $\mathcal{S}$ there exist $a_i, a_i' \in \mathbb{R} \cup \{-\infty, \infty\}$ such that

$$\varphi^{-1}([a_i, a_i']) = \mathrm{cl}(A_i) \tag{14}$$

and $a_i, a_i'$ are regular values of $\varphi$, see Definition 34. By regular level set theorem, the boundary $\varphi^{-1}(a_i)$ of $A_i$ is an embedded smooth submanifold of $\mathbb{R}^n$ [28]. We index the regular values such that $a_i < a_j$ if and only if $i < j$.

The proposed method heavily relies on transversal intersections; therefore, a formal definition of transversal intersection is provided in the following. After the definition, it is geometrically interpreted to clarify.

**Definition 15** (Transversal intersection [17]) Suppose that $N_1$ and $N_2$ are embedded submanifolds of $M$. We say that $N_1$ intersects $N_2$ transversally if, whenever $p \in N_1 \cap N_2$, we have $T_p(N_1) + T_p(N_2) = T_p(M)$. (The sum is not direct, just the set of sums of vectors, one from each of the two subspaces of the tangent space $T_p(M)$.)

In the left subplot of Fig. 4, level sets of two partitioning functions (hence two embedded submanifolds of $\mathbb{R}^2$) are illustrated. They intersect in the point $p$; however, their tangents (black lines) are identical. This implies that their tangent vectors only span one dimension at $p$, i.e., $T_p(N_1) + T_p(N_2) \neq T_p(M)$. Therefore, this intersection is not transversal. Note that with an arbitrary small perturbation, the intersection of the two level sets will be empty, as shown in the middle subplot (this perturbation is given by a smooth map, see Theorem 2.1 in [17]).

In the right subplot Fig. 4, two level sets intersecting at point $p$ are illustrated. Their tangent vectors (black lines) span $\mathbb{R}^2$, i.e., the level sets intersect transversally. Note that two manifolds that do not intersect are also transversal.

It is desired to obtain cells from the slices, and this is done by intersecting slices.

**Definition 16** (Transversal intersection of slices) We say that the slices $S_1$ and $S_2$ intersect each other transversally and write

$$S_1 \pitchfork S_2 = S_1 \cap S_2 \tag{15}$$

if their boundaries, $\mathrm{bd}(S_1)$ and $\mathrm{bd}(S_2)$, intersect each other transversally.

*Remark 2* A partition is robust, if any two slices intersect each other transversally, since they still intersect each other transversally after an arbitrary small perturbation. Hence, robustness is the reason for considering only transversal intersections in the proposed partitioning.

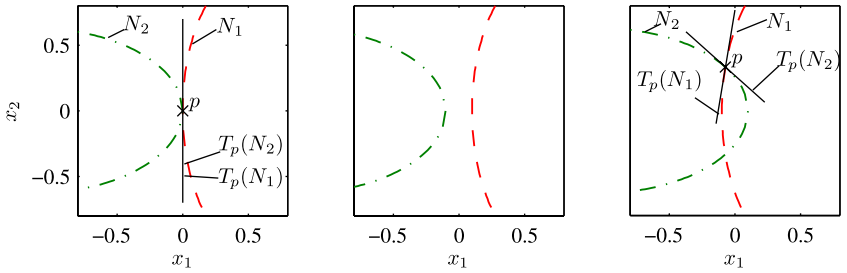


**Fig. 4** Illustration of two manifolds $N_1$ (*red dashed line*) and $N_2$ (*green dash-dot line*). In the *left subplot* $N_1$ and $N_2$ intersect; however, not transversally. In the *middle subplot* $N_1$ and $N_2$ do not intersect, but are transversal. In the *right subplot* $N_1$ and $N_2$ intersect transversally

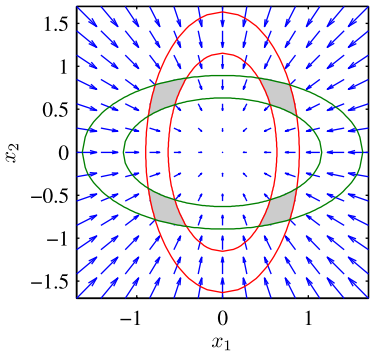


**Fig. 5** Phase plot (*arrows*) of a two-dimensional system and a partition generated utilizing two slice-families. The *shaded area* illustrates one extended cell, which consists of 4 connected components

**Definition 17** (Extended cell) Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of $k$ slice-families and let $\mathcal{G}(\mathcal{S}) \equiv \{1, \ldots, |\mathcal{S}^1|\} \times \cdots \times \{1, \ldots, |\mathcal{S}^k|\}$.
Denote the $j$th slice in $\mathcal{S}^i$ by $S^i_j$ and let $g \in \mathcal{G}(\mathcal{S})$. Then

$$e_{\text{ex},g} = \pitchfork_{i=1}^{k} S^i_{g_i}, \tag{16}$$

where $g_i$ is the $i$th element of the vector $g$. Any nonempty set $e_{\text{ex},g}$ is called an extended cell.

The set $\mathcal{G}(\mathcal{S})$, defined above, is used in the remainder of the paper.

The cells in (16) are denoted by extended cells, since the transversal intersection of slices may form multiple disjoint sets in the state space. This is illustrated in Fig. 5, where a two-dimensional state space is partitioned utilizing two slice-families (red and green).

The next example clarifies the indexing of the extended cells shown in Definition 17.

*Example 1* (Indexing extended cells) Given three slice-families $\{\mathcal{S}^i | i \in \{1, 2, 3\}\}$, an extended cell is indexed according to the ordering of the slices defining it, as shown below

$$e_{\text{ex},(9,5,27)} = S^1_9 \pitchfork S^2_5 \pitchfork S^3_{27}. \tag{17}$$

The vector $g$ from Definition 17 equals $(9, 5, 27)$ in this example, since $e_{\text{ex},(9,5,27)}$ is generated from slice number 9 in $\mathcal{S}^1$, slice number 5 in $\mathcal{S}^2$, and slice number 27 in $\mathcal{S}^3$.

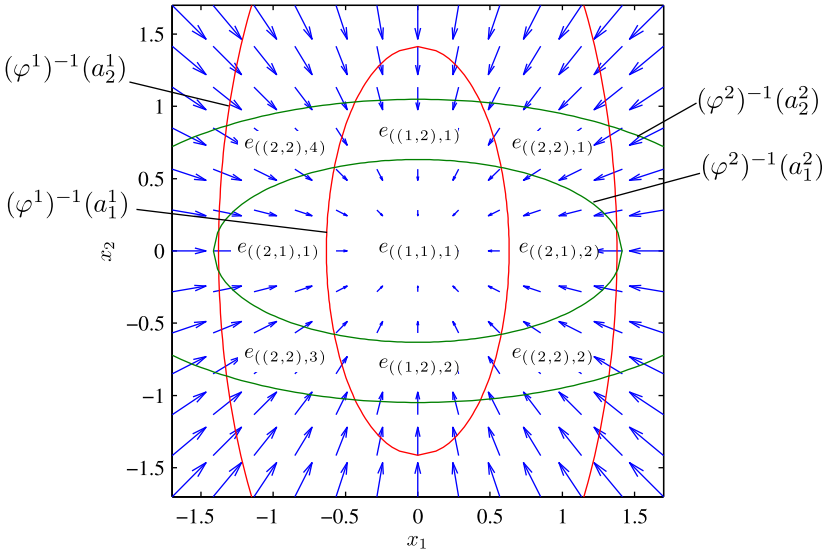


**Fig. 6** (Color online) Phase plot (*arrows*) of a two-dimensional state space. The state space is partitioned utilizing two slice-families, generated by the partitioning functions $\varphi^1$ (*red*) and $\varphi^2$ (*green*). Each cell of the partition is numbered according to Definition 17 and Definition 18

It is desired to have cells, which are connected. Therefore, the following is defined.

**Definition 18** (Cell) A cell is a connected component of an extended cell

$$\bigcup_h e_{(g,h)} = e_{\mathrm{ex},g}, \tag{18a}$$

where

$$e_{(g,h)} \cap e_{(g,h')} = \emptyset \quad \forall h \neq h'. \tag{18b}$$

We say that the slices $S^1_{g_1}, \ldots, S^k_{g_k}$ generate the cell $e_{(g,h)}$.

Figure 6 illustrates a two-dimensional state space partitioned using two slice-families and provides a geometric interpretation of the definitions related to the partitioning of a state space. The symbols of cells and level-sets of partitioning functions from the previous definitions are added to clarify their meaning. Note that subscript of $a^i_j$ refers to the number in the slice families, whereas the superscript is the number of slice family, i.e., $a^i_j$ is the $j$th regular value associated with partitioning function $\varphi^i$.

A finite partition based on the transversal intersection of slices is defined in the following.

**Definition 19** (Finite partition) Let $\mathcal{S}$ be a collection of slice-families, $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. We define a finite partition $K(\mathcal{S})$ by

$$e \in K(\mathcal{S}) \tag{19}$$

if and only if $e$ is a connected component of an extended cell.

Based on the definitions provided in this section, a procedure for obtaining a timed automaton from a finite partition of a state space is presented in the next section.

## 4 Generation of timed automaton from finite partition

The purpose of this section is to explain how a timed automaton $\mathcal{A}$ is generated from a finite partition $K(\mathcal{S})$ of the state space of a system $\Gamma$. For this, we use the abstraction procedure presented in [21]; nevertheless, we exclude the clock and constraints related to the time of traversing a cell. However, these can be added to improve accuracy. The reason why these clocks are removed is that they destroy the compositional structure of the timed automaton and requires computation of more guards and invariants.

First, we define an abstraction function associating each cell of the partition $K(\mathcal{S})$ to a location of a timed automaton.

**Definition 20** (Abstraction function) Let $K(\mathcal{S}) = \{e_i | i \in \boldsymbol{m}\}$ for some $m \in \mathbb{N}$ be a finite partition of the state space $X \in \mathbb{R}^n$ and $L(\mathcal{S}) = \{l_i | i \in \boldsymbol{m}\}$ be the locations of a timed automaton. Then an abstraction function for $(X, K(\mathcal{S}))$ is a multifunction $\alpha_K : X \to 2^{L(\mathcal{S})}$ defined by

$$\alpha_K(x) = \{l_i \in L(\mathcal{S}) \mid x \in e_i\}. \tag{20}$$

We use this abstraction function and generate a timed automaton according to the following procedure.

**Procedure 1** (Generation of a timed automaton) *Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a finite collection of slice-families. Then the timed automaton $\mathcal{A} = (L, L_0, C, \Sigma, I, \Delta)$ generated by the partition $K(\mathcal{S})$ is defined by*

**Locations:** *The locations of $\mathcal{A}$ are given by*

$$L = L(\mathcal{S}) \tag{21}$$

*This means that a location $l_{(g,h)}$ is associated with the cell $e_{(g,h)} = \alpha_K^{-1}(l_{(g,h)})$ of the partition $K(\mathcal{S})$, see Definition 20.*

**Clocks:** *The number of clocks equals the number of slice-families, i.e., $C = \{c^i | i \in \boldsymbol{k}\}$.*

**Invariants:** *In each location $l_{(g,h)}$, there are up to k invariants. We impose an invariant, whenever there is an upper bound for the time of staying in a slice generating the cell $e_{(g,h)}$*

$$I(l_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \preceq \overline{t}_{S^i_{g_i}} \tag{22}$$

*where $\overline{t}_{S^i_{g_i}} \in \mathbb{R}_{\geq 0}$ is an upper bound on the time for staying in $S^i_{g_i}$.*

**Input Alphabet:** *The input alphabet $\Sigma$ consists of k symbols $\{\sigma^i | i \in \boldsymbol{k}\}$. Note that $\sigma_i$ is associated with transitions between two slices in the slice-family $\mathcal{S}^i$.*

**Transition relations:** *If a pair of locations $l_{(g,h)}$ and $l_{(g',h')}$, associated with the cells $e_{(g,h)}$ and $e_{(g',h')}$, satisfy the following two conditions*

1. *$e_{(g,h)}$ and $e_{(g',h')}$ are adjacent, that is $e_{(g,h)} \cap e_{(g',h')} \neq \emptyset$, and*

2. $g_i' \leq g_i$ *for all* $i \in \boldsymbol{k}$.

*Then there is a transition relation*

$$\delta_{(g,h)\to(g',h')} = (l_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma, R_{(g,h)\to(g',h')}, l_{(g',h')}), \tag{23a}$$

*where*

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}_{S^i_{g_i}} & \text{if } g_i - g_i' = 1, \\ c^i \geq 0 & \text{otherwise} \end{cases} \tag{23b}$$

*and* $\underline{t}_{S^i_{g_i}} \in \mathbb{R}_{\geq 0}$ *is a lower bound on the time for staying in* $S^i_{g_i}$. *Note that* $g_i - g_i' = 1$ *whenever a transition labeled* $\sigma^i$ *is taken. If* $g_i - g_i' = 0$, *we stay in the slice from the* $i$*th slice-family generating cell* $e_{(g,h)}$; *hence, no active guards are on* $c^i$.
*Note that there is only constraints on the* $i$ *clock if* $g_i - g_i' = 1$, *since otherwise we do not exit a slice from the* $i$*th slice-family.*
*Let* $i \in \boldsymbol{k}$. *We define* $R_{(g,h)\to(g',h')}$ *by*

$$c^i \in R_{(g,h)\to(g',h')} \tag{23c}$$

*iff* $g_i - g_i' = 1$.

*The calculation of* $\underline{t}_{S^i_{g_i}}$ *and* $\overline{t}_{S^i_{g_i}}$ *can be accomplished in multiple ways, yielding different properties of the abstraction. This is explained in details in Sect.* 6, *where conditions for generating sound and complete abstractions are provided.*

For convenience the following notion is introduced.

**Definition 21** Let $\mathcal{S}$ be a finite collection of slice-families $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then $\mathcal{A}(\mathcal{S})$ is the timed automaton generated by $\mathcal{S}$ according to Procedure 1.

**Definition 22** A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ has locations given by

$$L = L_{\text{ex}}(\mathcal{S}) \tag{24}$$

where a location $l_{\text{ex},g} \in L_{\text{ex}}(\mathcal{S})$ is associated with the extended cell $e_{\text{ex},g}$ generated by the slice-family $\mathcal{S}$; hence, $e_{\text{ex},g} = \alpha^{-1}_{K_{\text{ex}}}(l_{\text{ex},g})$.

## 5 Properties of the generated timed automaton

A timed automaton generated from the presented procedure possesses some salient properties, due to the way state spaces are partitioned. Some of these properties are presented in this section. The section consists of four subsections, each explaining one of the properties.

### 5.1 Determinism of abstractions

The considered systems have unique maximal solutions, see Definition 1. Therefore, we also determine when a partition generates a deterministic timed automaton.

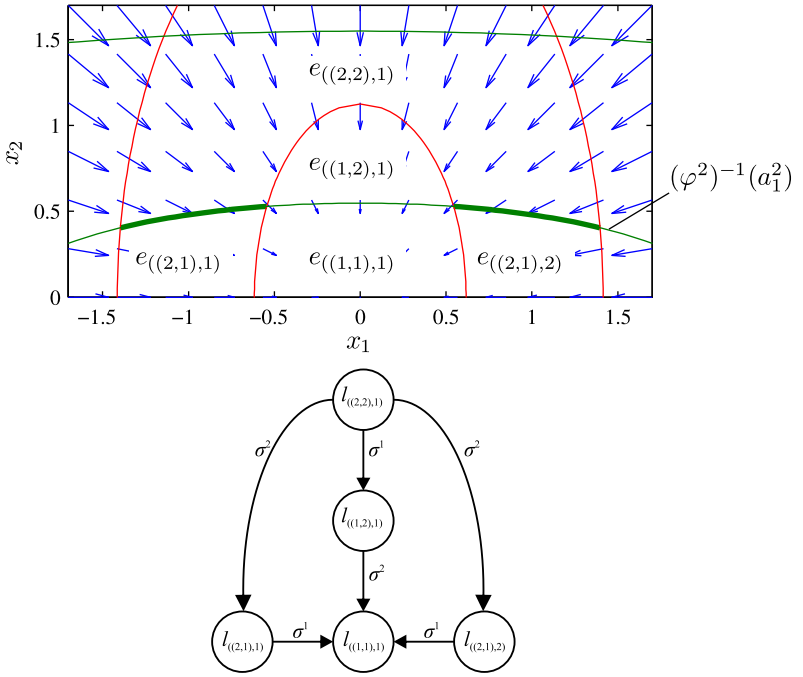


**Fig. 7** (Color online) Partition of a two-dimensional state space using two slice-families $\mathcal{S}^1$ (*red*) and $\mathcal{S}^2$ (*green*), and an illustration of the resulting nondeterministic timed automaton (for simplicity only the names of the locations and symbols on the transitions are shown in the figure). The set $e_{((2,2),1)} \cap (\varphi^2)^{-1}(a_1^2)$ is illustrated with a *bold green line*, to emphasize that Proposition 1 is not satisfied, as the set (25) is not connected

**Proposition 1** (Deterministic timed automaton) *The timed automaton $\mathcal{A}(\mathcal{S})$ is deterministic, if and only if for each cell $e_{(g,h)} \in K(\mathcal{S})$ and for all $i \in \boldsymbol{k}$ the set*

$$e_{(g,h)} \cap (\varphi^i)^{-1}(a^i_{g_i-1}) \tag{25}$$

*is connected.*

We show how to check if an abstraction is deterministic in Proposition 3; however, first we present an example to clarify Proposition 1.

*Example 2* Consider the partition of a two-dimensional state space shown in Fig. 7 generated using two slice-families $\mathcal{S}^1$ (red) and $\mathcal{S}^2$ (green). A timed automaton generated from the partition is shown in the bottom of the figure, excluding time information.

From the figure it is seen that the symbol $\sigma^2$ determines two transitions in the timed automaton from $l_{((2,2),1)}$: A transition to $l_{((2,1),1)}$ and a transition to $l_{((2,1),2)}$. The reason for this is that two cells are reachable from $e_{((2,2),1)}$ by crossing $(\varphi^2)^{-1}(a_1^2)$, as $e_{((2,2),1)} \cap (\varphi^2)^{-1}(a_1^2)$ (bold green line) consists of two connected components. This implies that Proposition 1 is not satisfied; hence, a timed automaton generated by this partition is nondeterministic.

For linear systems, it is possible to check if Proposition 1 is satisfied when quadratic Lyapunov functions are used as partitioning functions. This is explained in the following, using a proposition from [8] that determine when two quadratic forms intersect.

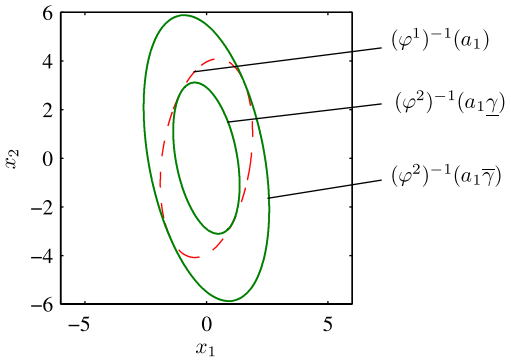


**Fig. 8** Illustration of a level set $(\varphi^1)^{-1}(a_1)$ (*dashed red*) and the level sets $(\varphi^2)^{-1}(a_1\underline{\gamma})$, and $(\varphi^2)^{-1}(a_1\overline{\gamma})$ (*solid green*) intersecting $(\varphi^1)^{-1}(a_1)$

**Proposition 2** (See [8]) *Suppose* $P^1 = (P^1)^{\mathrm{T}} > 0$, $P^2 = (P^2)^{\mathrm{T}} > 0$, *and let* $\varphi^1(x) = x^{\mathrm{T}} P^1 x$ *and* $\varphi^2(x) = x^{\mathrm{T}} P^2 x$. *Define* $\overline{\gamma}$ *to be the solution to the following optimization problem*

$$\textit{minimize} \quad \overline{\gamma} \tag{26a}$$

$$\textit{subject to} \quad P^2 - \overline{\gamma} P^1 \leq 0, \quad \overline{\gamma} > 0 \tag{26b}$$

*and define* $\underline{\gamma}$ *to be the solution to the following optimization problem*

$$\textit{maximize} \quad \underline{\gamma} \tag{27a}$$

$$\textit{subject to} \quad P^1 \underline{\gamma} - P^2 \leq 0, \quad \underline{\gamma} > 0. \tag{27b}$$

*Then the level set* $(\varphi^2)^{-1}(a_2)$ *generated for a regular value* $a_2$ *intersects* $(\varphi^1)^{-1}(a_1)$ *if and only if* $a_2 \in [a_1\underline{\gamma}, a_1\overline{\gamma}]$, *and* $(\varphi^2)^{-1}(a_2)$ *intersects* $(\varphi^1)^{-1}(a_1)$ *transversally if and only if* $a_2 \in (a_1\underline{\gamma}, a_1\overline{\gamma})$.

This is illustrated in Fig. 8, where the level sets $(\varphi^1)^{-1}(a_1)$, $(\varphi^2)^{-1}(a_1\underline{\gamma})$, and $(\varphi^2)^{-1}(a_1\overline{\gamma})$ are drawn.

From Fig. 7 it is seen that the timed automaton becomes nondeterministic, when a level set only intersects some, but not all level sets from the other slice families. Using this fact, the following proposition follows directly from Proposition 2, and is left without proof.

**Proposition 3** *Let* $\mathcal{A}(\mathcal{S})$ *be an abstraction of a linear system* $\Gamma = (X, f)$. *Let* $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ *be a collection of slice-families. Associate to each slice-family* $\mathcal{S}^i$ *a quadratic partitioning function* $\varphi^i(x) = x^{\mathrm{T}} P^i x$, *and let* $\mathcal{S}^i$ *be generated using the regular values* $\{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\}$. *For all* $i, j \in \boldsymbol{k}$, *define* $\overline{\gamma}^{ij}$ *to be the solution to the following optimization problem*

$$\textit{minimize} \quad \overline{\gamma}^{ij} \tag{28a}$$

$$\textit{subject to} \quad P^j - \overline{\gamma}^{ij} P^i \leq 0, \quad \overline{\gamma}^{ij} > 0 \tag{28b}$$

*and define* $\underline{\gamma}^{ij}$ *to be the solution to the following optimization problem*

$$\textit{maximize} \quad \underline{\gamma}^{ij} \tag{29a}$$

$$\textit{subject to} \quad P^i \underline{\gamma}^{ij} - P^j \leq 0, \quad \underline{\gamma}^{ij} > 0. \tag{29b}$$

*Then* $\mathcal{A}(\mathcal{S})$ *is deterministic if and only if for all* $i \neq j \in \boldsymbol{k}$ *and for all* $a^j \in \{a_k^j | k \in \{1, \ldots, |\mathcal{S}^j|\}\}$

$$a^j \in (a^i \underline{\gamma}^{ij}, a^i \overline{\gamma}^{ij}) \quad \forall a^i \in \{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\} \quad \textit{or} \tag{30a}$$

$$a^j \notin (a^i \underline{\gamma}^{ij}, a^i \overline{\gamma}^{ij}) \quad \forall a^i \in \{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\}. \tag{30b}$$

Remark that the optimization problems used in Proposition 3 can be solved using standard tools for solving linear optimization problems.

### 5.2 Compositionality of abstractions

Under certain conditions it is possible to generate the timed automaton as a parallel composition of multiple timed automata.

**Definition 23** (Parallel composition of timed automata) The parallel composition of two timed automata, $\mathcal{A}_i = (L_i, L_{0,i}, C_i, \Sigma_i, I_i, \Delta_i)$ for $i = 1, 2$ with transition relations $(l_i, G_{i,l_i \to l_i'}, \sigma^i, R_{i,l_i \to l_i'}, l_i')$, is denoted $\mathcal{A} = \mathcal{A}_1 \| \mathcal{A}_2$ and is a timed automaton $(L, L_0, C, \Sigma, I, \Delta)$, where:

- $L = L_1 \times L_2$.
- $L_0 = L_{0,1} \times L_{0,2}$.
- $C = C_1 \cup C_2$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$.
- $I : L \to \Psi(C)$, where $I(l_1, l_2) = I_1(l_1) \wedge I_2(l_2)$.
- $\Delta \subseteq L \times \Psi(C) \times \Sigma \times 2^C \times L$ is a finite set of transition relations, where $(l, G_{l \to l'}, \sigma, R_{l \to l'}, l')$ is defined by the following
  1. If $\sigma \in \Sigma_1 \cap \Sigma_2$ then $l = (l_1, l_2)$, $G_{l \to l'} = G_{1,l_1 \to l_1'} \wedge G_{2,l_2 \to l_2'}$, $R_{l \to l'} = R_{1,l_1 \to l_1'} \cup R_{2,l_2 \to l_2'}$, and $l' = (l_1', l_2')$.
  2. If $\sigma \in \Sigma_1$ and $\sigma \notin \Sigma_2$ then $l = (l_1, l_2)$, $G_{l \to l'} = G_{1,l_1 \to l_1'}$, $R_{l \to l'} = R_{1,l_1 \to l_1'}$, and $l' = (l_1', l_2)$.
  3. If $\sigma \notin \Sigma_1$ and $\sigma \in \Sigma_2$ then $l = (l_1, l_2)$, $G_{l \to l'} = G_{2,l_2 \to l_2'}$, $R_{l \to l'} = R_{2,l_2 \to l_2'}$, and $l' = (l_1, l_2')$.

**Proposition 4** *Let* $\mathcal{A}_{\text{ex}}(\mathcal{S})$ *be a timed automaton and let the slices of* $\mathcal{S}$ *be generated such that for each pair* $(S_{g_i}^i, S_{g_j}^j)$, *with* $i, j \in \boldsymbol{k}$, $g_i \in \{1, \ldots, |\mathcal{S}^i|\}$, $g_j \in \{1, \ldots, |\mathcal{S}^j|\}$, *we have*

$$S_{g_i}^i \pitchfork S_{g_j}^j \neq \emptyset \quad \forall i \neq j. \tag{31}$$

*Then,* $\mathcal{A}_{\text{ex}}(\mathcal{S})$ *is isomorphic to the parallel composition of k timed automata each generated by one slice-family* $\mathcal{S}^i$ *having an alphabet* $\Sigma_i = \{\sigma^i\}$.

Note that case 1 in Definition 23, where $\sigma \in \Sigma_i \cap \Sigma_j$ is never satisfied in this work, as $\Sigma_i \cap \Sigma_j = \emptyset$ for all $i \neq j$.

*Remark 3* A parallel composition of timed automata $\mathcal{A}_i(\mathcal{S}^i)$ for $i \in \boldsymbol{k}$ is similar to the intersection of slices in the slice-families $\mathcal{S}^i$. Therefore, the intersection of slices should be nonempty to let the locations of the timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ be such a parallel composition, as stated in Proposition 4.

The satisfaction of Proposition 4 can be checked using the following proposition.

**Proposition 5** *Let $\mathcal{A}(\mathcal{S})$ be an abstraction of a linear system $\Gamma = (X, f)$. Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families. Associate to each slice-family $\mathcal{S}^i$ a quadratic partitioning function $\varphi^i(x) = x^{\mathrm{T}} P^i x$, and let $\mathcal{S}^i$ be generated using the regular values $\{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\}$. For all $i, j \in \boldsymbol{k}$, define $\overline{\gamma}^{ij}$ to be the solution to the following optimization problem*

$$\text{minimize} \quad \overline{\gamma}^{ij} \tag{32a}$$

$$\text{subject to} \quad P^j - \overline{\gamma}^{ij} P^i \leq 0, \quad \overline{\gamma}^{ij} > 0 \tag{32b}$$

*and define $\underline{\gamma}^{ij}$ to be the solution to the following optimization problem*

$$\text{maximize} \quad \underline{\gamma}^{ij} \tag{33a}$$

$$\text{subject to} \quad P^i \underline{\gamma}^{ij} - P^j \leq 0, \quad \underline{\gamma}^{ij} > 0. \tag{33b}$$

*Then Proposition 4 is satisfied if and only if for all $i \neq j \in \boldsymbol{k}$ and for all $a^j \in \{a_k^j | k \in \{1, \ldots, |\mathcal{S}^j|\}\}$*

$$a^j \in (a^i \underline{\gamma}^{ij}, a^i \overline{\gamma}^{ij}) \quad \forall a^i \in \{a_k^i | k \in \{1, \ldots, |\mathcal{S}^i|\}\}. \tag{34a}$$

The property that $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$ is isomorphic to the parallel composition of $k$ timed automata is very important for computations, since it allows parallel verification of the $k$ timed automata each with only one clock. Furthermore, it makes it possible to sequentially add slice-families to the abstraction, to replace, and to refine slice-families to improve the accuracy of the abstraction.

The parallel composition of timed automata also allows the sequential verification of the abstraction. We show this in terms of safety in the following.

**Definition 24** (Safety) Given a timed automaton $\mathcal{A}(\mathcal{S})$ and a set of unsafe locations $L_{\mathrm{US}}$. The timed automaton $\mathcal{A}(\mathcal{S})$ is said to be safe if

$$\mathrm{Reach}_{[0,\infty)}(\mathcal{A}(\mathcal{S}), L_0) \cap L_{\mathrm{US}} = \emptyset. \tag{35}$$

**Proposition 6** *Let $\mathcal{A}_{\mathrm{ex}}(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1) \| \ldots \| \mathcal{A}_k(\mathcal{S}^k)$ be a timed automaton and let the timed automaton $\mathcal{A}_1(\mathcal{S}^1) \| \ldots \| \mathcal{A}_j(\mathcal{S}^j)$ be safe, for some $j \in \boldsymbol{k}$. Then, $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$ is also safe.*

This proposition is quite intuitive, when considering how the timed automaton is generated from positive invariant sets as shown in the following example.

*Example 3* Consider a timed automaton $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$ generated by two slice-families abstracting a two-dimensional system as shown in Fig. 9 (left subplot). Its initial states are illustrated as gray areas and the unsafe states are illustrated as black areas.

We compose $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$ into $\mathcal{A}(\mathcal{S}^1)$ (middle subplot) and $\mathcal{A}(\mathcal{S}^2)$ (right subplot). We observe that $\mathcal{A}(\mathcal{S}^2)$ is safe; hence, $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$ is also safe, which is seen in the left subplot.

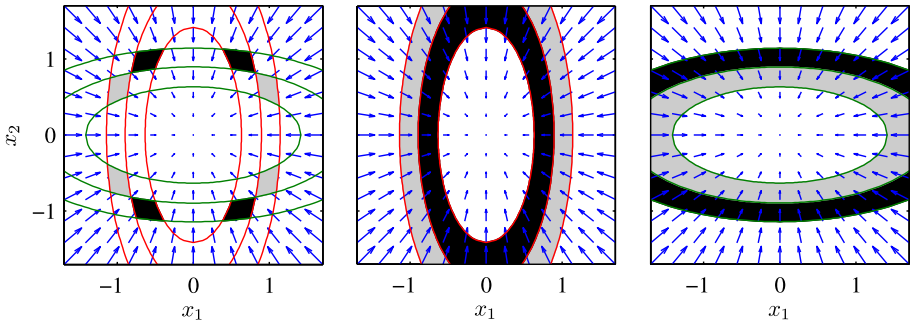


**Fig. 9** (Color online) Illustration of partition associated with $\mathcal{A}_{\text{ex}}(\mathcal{S})$ (*left subplot*), $\mathcal{A}(\mathcal{S}^1)$ (*middle subplot*), and $\mathcal{A}(\mathcal{S}^2)$ (*right subplot*). The *gray areas* illustrate initial states and the *black* are illustrates unsafe states

### 5.3 Bisimilarity of abstractions generated from cells and extended cells

Under certain conditions, the timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ is bisimilar to the timed automaton $\mathcal{A}(\mathcal{S})$, bisimilarity is defined in Definition 35. In the next proposition we say that the timed automata $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are related by bisimulation.

**Proposition 7** *Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families, and $\varphi^i$ be a partitioning function for $\mathcal{S}^i$. A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ generated by extended cells is bisimilar to a timed automaton $\mathcal{A}(\mathcal{S})$ generated by cells if for each cell $e_{(g,h)}$ and each $i \in \boldsymbol{k}$*

$$e_{(g,h)} \cap (\varphi^i)^{-1}(a^i_{g_i-1}) \neq \emptyset \quad \forall h \quad \textit{or} \tag{36a}$$

$$e_{(g,h)} \cap (\varphi^i)^{-1}(a^i_{g_i-1}) = \emptyset \quad \forall h. \tag{36b}$$

If (36) holds, then all cells in each extended cell have the same symbols on their outgoing transitions; hence, $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are bisimilar. The following example clarifies this proposition.

*Example 4* To illustrate the use of Proposition 7 three different partitions of a two-dimensional state space are shown in Fig. 10.

In the partitions shown in the left and right side of Fig. 10, the conditions shown in (36) are satisfied for all $g$ and $h$. In the partition shown in the middle side of Fig. 10, the constraints shown in (36) are not satisfied for e.g. $g = (2, 2, 2)$ (shaded region), as $(\varphi^3)^{-1}(a^3_1)$ (inner black line) does not intersect all cells in $e_{\text{ex},(2,2,2)}$.

### 5.4 Convergence check via partial order

We can define a partial order for the locations of a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$ generated by Procedure 1 as follows. Recall that a partial order on a set $X$, denoted $\sqsubseteq$, is a relation on $X$ that is reflexive, transitive, and anti-symmetric.

We define the partial order on the set of locations abstracting extended cells as follows.

**Definition 25** (Partial order of locations) Given a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S}) = (L, L_0, C, \Sigma, I, \Delta)$. Let $R_{\text{par}} \subseteq L \times L$ be a relation defined by

$$(l_{\text{ex},g}, l_{\text{ex},g'}) \in R_{\text{par}} \quad \text{iff} \tag{37a}$$

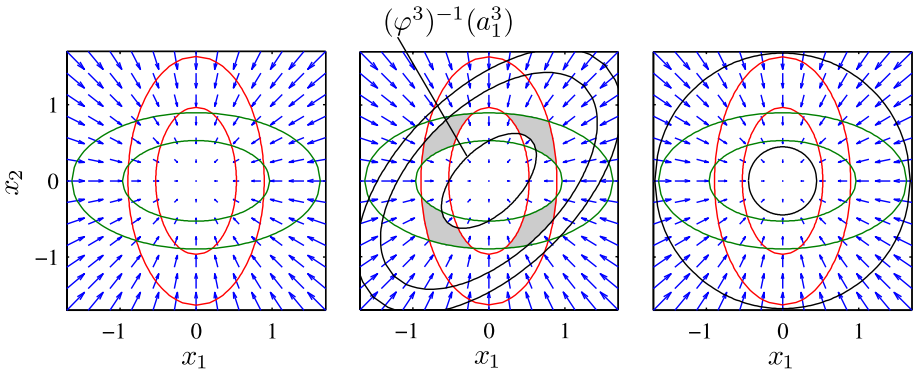


**Fig. 10** (Color online) Illustration of three different partitions of a two-dimensional state space. The *left* and *right partitions* satisfy Proposition 7. The *middle partition* does not satisfy Proposition 7, as, e.g., the extended cell *shaded with gray* consists of cells, which have either two or three reachable cells

$$g_i \leq g_i' \quad \forall i \in \boldsymbol{k}. \tag{37b}$$

We write $l_{\mathrm{ex},g} \sqsubseteq l_{\mathrm{ex},g'}$ when $(l_{\mathrm{ex},g}, l_{\mathrm{ex},g'}) \in R_{\mathrm{par}}$.

The relation $R_{\mathrm{par}}$ is a partial order relation, and $(L, \sqsubseteq)$ is a partially ordered set, which justifies its name.

From the partial order, we can provide a rough estimate of whether a location is reachable or not according to the following.

**Proposition 8** *Given a timed automaton $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$, a partially ordered set $(L, \sqsubseteq)$ of its locations, and an initial location $l_{\mathrm{ex},g}$. Let $l_{\mathrm{ex},g} \sqsubseteq l_{\mathrm{ex},g'}$, then*

$$\mathrm{Reach}_{[0,\infty)}(\mathcal{A}, l_{\mathrm{ex},g}) \cap l_{\mathrm{ex},g'} = \emptyset. \tag{38}$$

We define a lattice according to the following.

**Definition 26** (Lattice) A partially ordered set $(X, \sqsubseteq)$ is said to be a lattice if for any finite set $X' \subseteq X$, the supremum and infimum of $X'$ exist and belong to $X$.

**Proposition 9** *Let $\mathcal{A}_{\mathrm{ex}}(\mathcal{S}) = (L, L_0, C, \Sigma, I, \Delta)$ be a timed automaton generated from Procedure 1, and let $\sqsubseteq$ be the partial order from Definition 25. Then the partially ordered set $(L, \sqsubseteq)$ is a lattice.*

We see that $(L, \sqsubseteq)$ is a lattice by construction, as whenever, e.g., $l_{\mathrm{ex},(a,b)}, l_{\mathrm{ex},(b,a)} \in L$, then also $l_{\mathrm{ex},(a,a)}, l_{\mathrm{ex},(b,b)} \in L$, and $l_{\mathrm{ex},(a,a)} < l_{\mathrm{ex},(a,b)}$, $l_{\mathrm{ex},(a,a)} < l_{\mathrm{ex},(b,a)}$ and $l_{\mathrm{ex},(a,b)} < l_{\mathrm{ex},(b,b)}$, $l_{\mathrm{ex},(b,a)} < l_{\mathrm{ex},(b,b)}$. Therefore, both supremum and infimum are defined for all $L' \subseteq L$.

Additionally, it is seen that almost all solutions of the system converge towards the location $\inf(L)$. By this we mean that the solutions end in the location $\inf(L)$. This can happen when a system with saddle points is abstracted. Consider a two-dimensional linear system with a saddle point. For this system all solutions initialized on one of the eigenaxis converge to the saddle point, which will not be abstracted by the location $\inf(L)$. However, in $\mathbb{R}^n$ for $n > 1$ this eigenaxis has Lebesgue measure zero. Therefore, we say that almost all solutions

diverge from the saddle point. This also means that the analysis of convergence using $\inf(L)$ is "blind" to solutions of measure zero.

## 6 Conditions for the partitioning

The purpose of this section is to set up necessary and sufficient conditions for the partition of the state space to generate sound, complete, and refinable abstractions.

### 6.1 Sound and complete abstractions

A useful abstraction shall preserve safety. Therefore, we introduce sound and complete abstractions in the following [3]. A sound abstraction can verify safety, and a complete abstraction can both verify and falsify safety.

**Definition 27** (Sound abstraction) Let $\Gamma = (X, f)$ be a dynamical system and suppose its state space $X$ is partitioned by $K(\mathcal{S}) = \{e_i | i \in \boldsymbol{k}\}$. Let the initial states $X_0 = \bigcup_{i\in\mathcal{I}} e_i$ with $\mathcal{I} \subseteq \boldsymbol{k}$. Then a timed automaton $\mathcal{A} = (L, L_0, C, \Sigma, I, \Delta)$ with $L_0 = \{l_i \mid i = \mathcal{I}\}$ is said to be a sound abstraction of $\Gamma$ on $[t_1, t_2]$ if for all $t \in [t_1, t_2]$

$$e_i \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \quad \text{implies} \tag{39a}$$

$$\exists l_0 \in L_0 \quad \text{such that}$$

$$\alpha_K(e_i) \in \phi_{\mathcal{A}}(t, l_0). \tag{39b}$$

If a sound abstraction $\mathcal{A}$ is safe then $\Gamma$ is also safe, as the abstraction reaches all locations reached by $\Gamma = (X, f)$.

**Definition 28** (Complete abstraction) Let $\Gamma$ be a dynamical system and suppose its state space $X$ is partitioned by $K(\mathcal{S}) = \{e_i | i \in \boldsymbol{k}\}$ and let the initial states be $X_0 = \bigcup_{i\in\mathcal{I}} e_i$ with $\mathcal{I} \subseteq \boldsymbol{k}$. Then a timed automaton $\mathcal{A} = (L, L_0, C, \Sigma, I, \Delta)$ with $L_0 = \{l_i \mid i = \mathcal{I}\}$ is said to be a complete abstraction of $\Gamma$ on $[t_1, t_2]$ if it is a sound abstraction and for all $t \in [t_1, t_2]$ and

$$\text{for each } l_i \in \text{Reach}_{[t,t]}(\mathcal{A}, L_0) \tag{40a}$$

$$\exists x_0 \in X_0 \quad \text{such that}$$

$$\phi_\Gamma(t, x_0) \in \alpha_K^{-1}(l_i). \tag{40b}$$

If a complete abstraction $\mathcal{A}$ is safe (unsafe) then $\Gamma$ is also safe (unsafe).

A sound and a complete abstraction of a dynamical system is illustrated in Fig. 11.

*Remark 4* It is not sufficient to demand that an abstraction is complete, since an abstraction with only one location abstracting the entire state space is always complete.

**Proposition 10** *A timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S}) = \mathcal{A}_1 \| \ldots \| \mathcal{A}_k$ is a sound (complete) abstraction of the system $\Gamma$, if and only if $\mathcal{A}_1, \ldots, \mathcal{A}_k$ are sound (complete) abstractions of $\Gamma$.*

Sufficient conditions for soundness and completeness of an abstraction are formulated in the following.

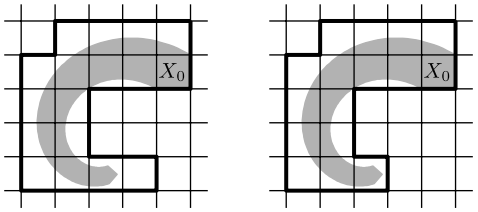


**Fig. 11** Illustration of the reachable set of a dynamical system (*gray*) from initial set $X_0$ and a sound approximation of this (*cells within bold black lines*) on the *left* and a complete abstraction on the *right*. Note that the lowest right cell is not reached by the dynamical system, but is reached by the sound abstraction

**Proposition 11** (Sufficient condition for soundness) *A timed automaton $\mathcal{A}(\mathcal{S})$ is a sound abstraction of the system $\Gamma$, if its invariants and guards satisfy*

$$\underline{t}_{S^i_{g_i}} \leq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{\sup\{|\dot{\varphi}^i(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{g_i}\}}, \tag{41a}$$

$$\overline{t}_{S^i_{g_i}} \geq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{\inf\{|\dot{\varphi}^i(x)| \in \mathbb{R}_{\geq 0} | x \in S^i_{g_i}\}} \tag{41b}$$

*where $\dot{\varphi}^i(x)$ is defined as shown in* (4a).

The values of guards of $\underline{t}_{S^i_{g_i}}$ and $\overline{t}_{S^i_{g_i}}$ can be algorithmically generated. The invariant $\overline{t}_{S^i_{g_i}}$ can be generated from Algorithm 1 and the guard can be generated from Algorithm 2.

**Algorithm 1** Suppose $\Gamma = (X, f)$ is a linear system, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i = x^{\mathrm{T}} P^i x$ is associated with $\mathcal{S}^i$, and $\mathcal{S}^i$ is generated using the regular values $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$. Let $\dot{\varphi}^i = -x^{\mathrm{T}} Q^i x$, and define $\underline{\gamma}^i$ as the solution to the following optimization problem

$$\text{maximize} \quad \underline{\gamma}^i \tag{42a}$$

$$\text{subject to} \quad \underline{\gamma}^i P^i - Q^i \leq 0, \quad \underline{\gamma}^i > 0. \tag{42b}$$

Then for any location $l_{(g,h)} \in L$, the invariant is

$$I(l_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \overline{t}_{S^i_{g_i}}, \text{ where} \tag{43a}$$

$$\overline{t}_{S^i_{g_i}} \geq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{a^i_{g_i-1}\underline{\gamma}^i}. \tag{43b}$$

**Algorithm 2** Suppose $\Gamma = (X, f)$ is a linear system, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i = x^{\mathrm{T}} P^i x$ is associated with $\mathcal{S}^i$, and $\mathcal{S}^i$ is generated using the regular values $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$. Let $\dot{\varphi} = -x^{\mathrm{T}} Q^i x$ and define $\overline{\gamma}^i$ to be the solution to the following optimization problem

$$\min \quad \overline{\gamma}^i \tag{44a}$$

$$\text{subject to} \quad Q^i - \overline{\gamma}^i P^i \leq 0 \tag{44b}$$

$$\overline{\gamma}^i > 0. \tag{44c}$$

Then for every pair of locations, $l_{(g,h)}, l_{(g',h')} \in L$, where $g'_i - 1 = g_i$ there is a transition relation

$$\delta_{(g,h)\to(g',h')} = (l_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma, R_{(g,h)\to(g',h')}, l_{(g',h')}), \tag{45a}$$

where

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}_{S^i_{g_i}} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise} \end{cases} \tag{45b}$$

and

$$\underline{t}_{S^i_{g_i}} \leq \frac{|a^i_{g_i} - a^i_{g_i-1}|}{a^i_{g_i}\overline{\gamma}^i}. \tag{45c}$$

The sufficient condition states that the abstraction is sound if $\underline{t}_{S^i_{g_i}}$ is less than or equal to the time it takes to traverse $S^i_{g_i}$ maintaining a constant speed equal to the largest possible speed within $S^i_{g_i}$. Similarly, $\overline{t}_{S^i_{g_i}}$ should be greater than or equal to the time it takes to traverse $S^i_{g_i}$ maintaining a constant speed equal to the smallest possible speed within $S^i_{g_i}$. In the next example of a one-dimensional system, we calculate these sufficient conditions.

*Example 5* Consider the system $\Gamma = (X, f)$, where $f$ is

$$\dot{x} = x \tag{46}$$

and $X = [0, 3]$. Let $\varphi = \frac{1}{2}x^2$ be a partitioning function for the system. Then $\dot{\varphi} = -x^2$ according to (4a). An illustration of the partitioning function $\varphi$ and its derivative $\dot{\varphi}$ is shown in Fig. 12. We partition $X$ into three slices utilizing the set of values $(a^1_1, a^1_2, a^1_3) = (1/2, 2, 9/2)$ such that $S^1_1 = [0, 1]$, $S^1_2 = [1, 2]$, and $S^1_3 = [2, 3]$, as shown in Fig. 13.

Now, it is possible to calculate the sufficient conditions for soundness shown in (41). These are

$$\underline{t}_{S^1_2} \leq \frac{|a^1_2 - a^1_1|}{|-(a^1_2)^2|} = \frac{3/2}{4}\ \text{s}, \qquad \overline{t}_{S^1_2} \geq \frac{|a^1_2 - a^1_1|}{|-(a^1_1)^2|} = 3/2\ \text{s}, \tag{47a}$$

$$\underline{t}_{S^1_3} \leq \frac{|a^1_3 - a^1_2|}{|-(a^1_3)^2|} = \frac{5/2}{9}\ \text{s}, \qquad \overline{t}_{S^1_3} \geq \frac{|a^1_3 - a^1_2|}{|-(a^1_2)^2|} = \frac{5/2}{4}\ \text{s}. \tag{47b}$$

Note that neither guards nor invariants for $S^1_1$ are calculated, as the location associated with this slice has no outgoing transitions and we can stay in this location forever.

**Proposition 12** (Sufficient condition for completeness) *Let $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of slice-families, and let*

$$S^i_{g_i} = (\varphi^i)^{-1}([a^i_{g_i-1}, a^i_{g_i}]). \tag{48}$$

*A deterministic timed automaton $\mathcal{A}(\mathcal{S})$ is a complete abstraction of $\Gamma$ if*

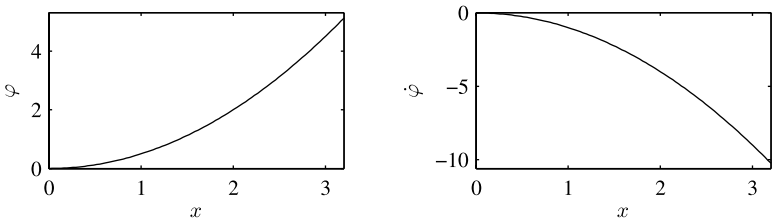


**Fig. 12** Plot of the partitioning function and its derivative

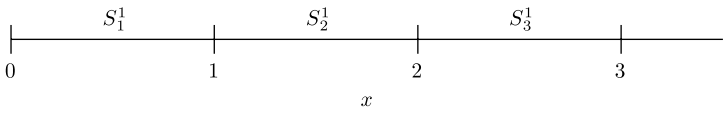


**Fig. 13** Illustration of a one-dimensional state space divided into three slices

1. *for any* $g \in \mathcal{G}(\mathcal{S})$, *recall the definition of* $\mathcal{G}(\mathcal{S})$ *from Definition* 17, *with* $g_i \geq 2$ *there exists a time* $t^i_{g_i}$ *such that for all* $x_0 \in (\varphi^i)^{-1}(a^i_{g_i})$

$$\phi_\Gamma(t^i_{g_i}, x_0) \in (\varphi^i)^{-1}(a^i_{g_i-1}) \tag{49}$$

   *and*
2. $\overline{t}_{S^i_{g_i}} = \underline{t}_{S^i_{g_i}} = t^i_{g_i}$.

Equation (49) states that it takes the time $t^i_{g_i}$ for all trajectories of $\Gamma$ to propagate from $(\varphi^i)^{-1}(a^i_{g_i})$ to $(\varphi^i)^{-1}(a^i_{g_i-1})$ (i.e., $t^i_{g_i}$ is the time to travel through slice $S^i_{g_i}$). Utilizing this time for both the invariant and guard conditions (i.e., $\overline{t}_{S^i_{g_i}} = \underline{t}_{S^i_{g_i}} = t^i_{g_i}$) implies that the abstraction is complete. Recall that $\overline{t}$ is used for invariants, while $\underline{t}$ is used for guard conditions.

*Example 6* Consider the system defined in Example 5 and partition it into the same slices $S^1_1$, $S^1_2$, and $S^1_3$.

The condition shown in (49) is automatically satisfied, as there is only one solution from each level surface of $\varphi$. Therefore, $\underline{t} = \overline{t}$, i.e.,

$$\underline{t}_{S^1_2} = \overline{t}_{S^1_2} = \ln(4)\ \text{s}, \tag{50a}$$

$$\underline{t}_{S^1_3} = \overline{t}_{S^1_3} = \ln(9/4)\ \text{s}. \tag{50b}$$

This finalizes the explanation of sound and complete abstractions, but as stared in Remark 4 another condition, called refinability, should also be considered.

### 6.2 Refinable abstraction

To ensure that an abstraction can get any desired accuracy, it should be refinable according to the following definitions.

**Definition 29** (Refinement of partition) Let the partition $K(\mathcal{S})$ be generated by the slice-families $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then the partition $K(\tilde{\mathcal{S}})$ is a refinement of $K(\mathcal{S})$ if and only if for any $e \in K(\mathcal{S})$ there is a cell $\tilde{e} \in K(\tilde{\mathcal{S}})$ such that

$$\tilde{e} \subseteq e. \tag{51}$$

**Definition 30** (Refinable abstraction) An abstraction $\mathcal{A}(\mathcal{S})$ of a system $\Gamma$ is said to be refinable if for all $\epsilon > 0$ there exists an abstraction $\mathcal{A}(\tilde{\mathcal{S}})$, where $K(\tilde{\mathcal{S}})$ is a refinement of $K(\mathcal{S})$, such that for all $\tilde{e} \in K(\tilde{\mathcal{S}})$

$$\tilde{e} \subseteq B(\epsilon), \tag{52}$$

where $B(\epsilon)$ is a ball with radius $\epsilon$.

The least $\epsilon$ that satisfies (52) for all $\tilde{e} \in K(\tilde{\mathcal{S}})$ is called the radius of the partition. We see that combining Definition 28 and Definition 30 yields the following corollary.

**Corollary 1** *If the system $\Gamma$ is abstracted with a complete and refinable abstraction $\mathcal{A}(\mathcal{S})$, then for all $\epsilon > 0$ there exists a partition $\mathcal{A}(\tilde{\mathcal{S}})$ such that for all $t \in [t_1, t_2]$*

$$\alpha_K^{-1}(\text{Reach}_{[t,t]}(\mathcal{A}(\tilde{\mathcal{S}}), L_0)) \subseteq \text{Reach}_{[t,t]}(\Gamma, X_0) + B(\epsilon). \tag{53}$$

We say that the accuracy of the abstraction is the smallest $\epsilon$ such that (53) is satisfied. Corollary 1 states that a complete and refinable abstraction can approximate the reachable states of a system $\Gamma$ arbitrarily close; hence, this type of abstraction solves Problem 1 in Sect. 1. In conclusion, to get any desired radius of the partition, all cells of its partition $K(\mathcal{S})$ should converge towards points. In the next proposition, we answer the question of minimal number of slice-families necessary to construct a refinable abstraction.

**Proposition 13** (Necessary condition for refinable abstraction) *If $\mathcal{A}(\mathcal{S})$ is a refinable abstraction of a system $\Gamma$, then $\mathcal{S}$ is a collection of $n$ slice-families.*

To clarify this proposition the following example is provided.

*Example 7* Consider a partition of the state space of a two-dimensional system generated utilizing one slice-family shown in Fig. 14.

Imagine that level curves are added to improve the accuracy of the partition. It is seen that the smallest cells that can be created utilizing only one slice-family are level curves of

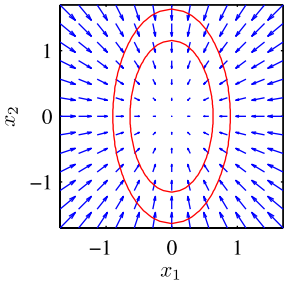


**Fig. 14** Partition of a two-dimensional state space utilizing one slice-family

the partitioning function. Therefore, the radius of the partition cannot converge to zero, as required by Definition 30. However, partitioning the same state space utilizing two slice-families would allow the radius of the partition to go to zero.

Now, the conditions for obtaining sound, complete, and refinable abstractions have been set up. In the next section, a method for synthesizing such abstractions using Lyapunov functions is provided.

## 7 Partitioning the state space using Lyapunov functions

To synthesize the partitioning presented in this paper, we need to generate partitioning functions. As partitioning functions, we use Lyapunov functions since their sub-level sets are positive invariant sets. It is beneficial to use Lyapunov functions as partitioning functions, as this enables the use of existing methods for generating Lyapunov functions used in controller design. This is possible even though Lyapunov functions are usually associated to systems with stable equilibria, but recall from Definition 3 that in this context they are associated to dynamical systems with hyperbolic equilibria. This is done to allow Lyapunov functions to be partitioning functions for both stable and unstable systems.

In the following, we provide existence results for refinable, sound, and complete abstractions.

### 7.1 Refinable abstraction

The use of continuous partitioning functions, see Definition 14, for generating the partition gives a natural and easy way to describe the boundaries of slices and cells. This also automatically makes partitions generated by $n$ slice-families refinable.

A partition generated by continuous partitioning functions can be refined, since for any slice $\varphi^{-1}([a_1, a_2])$ the regular values $a_1$ and $a_2$ can be chosen arbitrarily close to each other. Therefore, if $n$ such functions exist, the abstraction generated from the partition is refinable. The existence of $n$ Lyapunov functions for a Morse-Smale system is provided in Proposition 14.

### 7.2 Sound abstraction

In this subsection we show that there exists a sound abstraction of any Morse-Smale system. We do this by showing the existence of Lyapunov functions for Morse-Smale systems, actually we show that there exists $n$ Lyapunov functions for any Morse-Smale system; hence, they are sound and refinable abstractions.

**Definition 31** Let $a \in \mathrm{CrV}(f)$ if and only if there is $p \in Cr(f)$ such that $f(p) = a$ and suppose $\Gamma = (X, f)$. Then two Lyapunov functions $\varphi^1, \varphi^2 : \mathbb{R}^n \to \mathbb{R}$ are transversal if the level sets $(\varphi^1)^{-1}(a)$ and $(\varphi^2)^{-1}(a)$ are transversal for any $a \in \mathbb{R} \backslash \{\mathrm{CrV}(f)\}$.

**Proposition 14** *Let $n > 1$. For any Morse-Smale system on $\mathbb{R}^n$, there exists $n$ transversal Lyapunov functions.*

From this, the following theorem follows easily.

**Theorem 3** *Let $\Gamma = (X, f)$ be a Morse-Smale system and let $n > 1$. Then there exists a sound and refinable abstraction $\mathcal{A}(\mathcal{S})$ of $\Gamma$ generated by $n$ transversal Lyapunov functions.*

### 7.3 Complete abstraction

To generate complete abstractions, we set up a proposition that gives an easy testable condition for completeness. A complete abstraction of (1) can be obtained by constructing a partition generated by Lyapunov functions, which satisfies Proposition 12.

**Proposition 15** *Let each slice-family of* $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$ *be associated with a Lyapunov function* $\varphi^i(x)$ *for the system* $\Gamma$, *such that* $S^i_j = (\varphi^i)^{-1}([a^i_{j-1}, a^i_j])$ *and let*

$$\varphi^i(x) = \alpha \dot{\varphi}^i(x) \quad \forall x \in \mathbb{R}^n. \tag{54}$$

*Then* $\mathcal{A}(\mathcal{S})$ *is a complete abstraction of* $\Gamma$.

The existence of such Lyapunov functions is addressed in the following proposition.

**Proposition 16** *For any hyperbolic linear system* $\Gamma$ *there exists n transversal Lyapunov functions* $\varphi_i(x)$ *each satisfying*

$$\varphi^i(x) = \alpha \dot{\varphi}^i(x) \quad \forall x \in \mathbb{R}^n. \tag{55}$$

**Theorem 4** *For any hyperbolic linear system* $\Gamma$ *there exists a complete abstraction given by n transversal Lyapunov functions.*

## 8 Examples

To illustrate the use of the proposed abstraction method, three examples are provided in this section. In the first example, we provide the reachable sets of a sound and a complete abstractions of a simple dynamical system are provided. In the second example, we demonstrate what type of questions we can answer using the proposed abstraction. Note that these requirements cannot be verified using conventional technics from control theory; however, using some existing simulation-based verification tools the verification is possible. In the third example, we demonstrate that the method is applicable for high-dimensional systems, by verifying a 100 dimensional system.

In the following, we compare the reachable sets of a sound and a complete abstraction and see how their different partitions look. It is chosen to utilize a two-dimensional linear system in the example, since it is possible to visualize its state space, but the method applies for systems of arbitrary dimension. The considered linear system is specified in the following

$$\dot{x} = Ax \tag{56}$$

with $X = [-10, 10] \times [-10, 10]$

$$A = \begin{bmatrix} -3 & -1 \\ -2 & -5 \end{bmatrix} \quad \text{and} \quad X_0 = [1.5, 2] \times [-10, -9.5].$$

In both cases of the sound and the complete abstraction, the state space is partitioned utilizing two different Lyapunov functions, i.e., two slice-families. Furthermore, each slice-family consists of 10 sub-level sets, which are equally distributed on the considered state

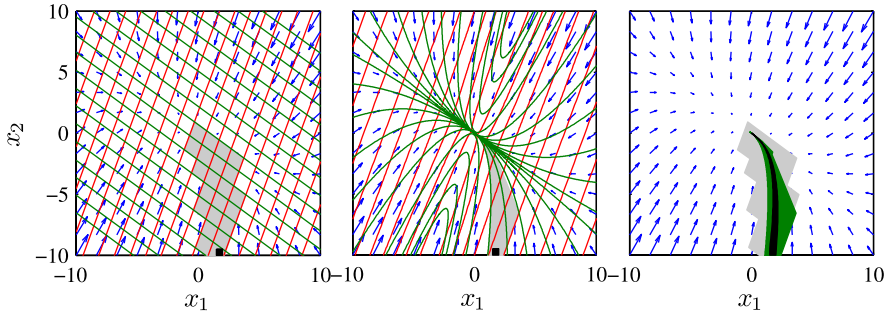


**Fig. 15** (Color online) Illustration of a sound abstraction (*left subplot*), a complete abstraction (*middle subplot*), and a comparison of their reachable sets (*right subplot*). The *black area* in the *right subplot* is the reachable state space of (56), the *green area* is the reachable state space of the complete abstraction, and the *gray area* is the reachable states of the sound abstraction

space Fig. 15. Both abstractions are timed automata with 361 locations (100 extended locations).

To simplify Fig. 15, we have chosen to only depict the partitions of the state space, not the timed automata.

In the figure, the reachable states of the two abstractions are shaded gray and are compared with the reachable space of the considered system in the right subplot. Notice that no effort is done to make the initial states (black box) fit with the cells of the partitions; hence, this introduces some over-approximation of the abstractions. In conclusion, the reachable state space for the sound abstraction is larger than the reachable state space for the complete abstraction, as expected.

When we look at the shape of the cells of the two partitions, it is seen that the cells of the complete abstraction look more complicated than the shape of the sound partition. Additionally, more computations are necessary for generating the complete abstraction, as it is generated via conjugate transformation, as explained in the proof of Proposition 16. Hence, the increased accuracy comes at a price—increased computational complexity.

The next example shows what type of specifications we are capable of checking using the proposed abstraction. In the example, we consider a slightly more complicated example, and a very complicated specification. The system is given by the following three-dimensional system (again the dimensionality is kept low to allow visualization of the example)

$$\dot{x} = \begin{bmatrix} -0.1 & -1 & 0 \\ 1 & -0.1 & 0 \\ 0 & 0 & -0.15 \end{bmatrix} x. \tag{57}$$

Now, we check if the system satisfies the following specification, which is illustrated in Fig. 16: Does all trajectories of the system (57) initialized in $X_0 = [-0.1,\ 0.1] \times [0.8,\ 1] \times [0.9,\ 1]$ (blue box)

- avoid the unsafe region (red box),
- reach the bottom half of the state space (below the gray surface) within 10 s,
- and reach the goal set (green box) within 20 s and stay there.

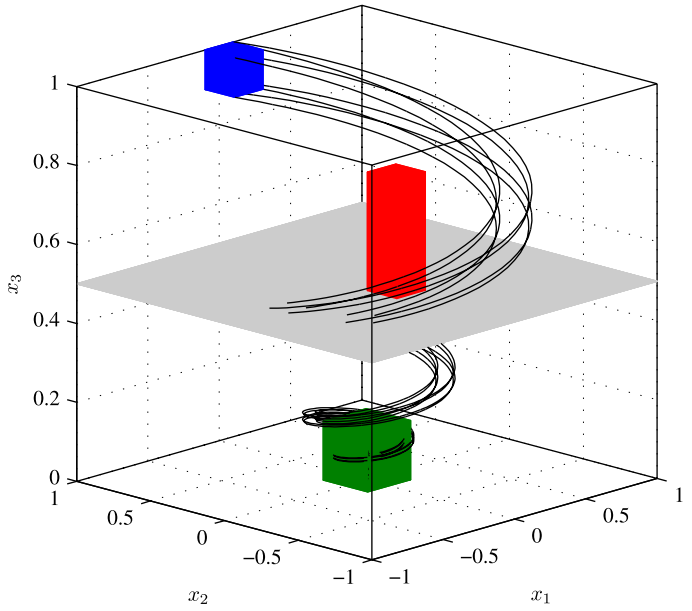


**Fig. 16** (Color online) State space of the considered system. The *blue box* illustrates the initial states of the system, the *red box* illustrates the unsafe states, and the *green box* illustrated the goal states. The plane given by $x_3 = 0.5$ is illustrated with *gray* and finally a set of system trajectories are drawn with *black lines*

To verify this specification, we partition the state space using three quadratic Lyapunov functions $\varphi^i(x) = x^{\mathrm{T}} P^i x$, for $i \in \{1, 2, 3\}$ and

$$P^1 = \begin{bmatrix} 0.0050 & 0 & 0 \\ 0 & 0.0050 & 0 \\ 0 & 0 & 3.3333 \end{bmatrix}, \tag{58a}$$

$$P^2 = \begin{bmatrix} 4.3069 & 0.0693 & 0 \\ 0.0693 & 4.6931 & 0 \\ 0 & 0 & 0.0033 \end{bmatrix}, \tag{58b}$$

$$P^3 = \begin{bmatrix} 5.2475 & -0.0248 & 0 \\ -0.0248 & 4.7525 & 0 \\ 0 & 0 & 0.0033 \end{bmatrix}. \tag{58c}$$

The partition is not shown in the figure; whereas, both the requirements and a set of trajectories of (57) are illustrated.

The analysis of the system says that the requirements are satisfied, as no trajectories reach the red box, all trajectories go below $x_3 = 0.5$ within 7 s and stay inside the green box after 12.7 s. This also complies with the simulated trajectories shown in Fig. 16.

Now, we verify a 100-dimensional system, by checking if all trajectories initialized in the unit hypercube centered at $x_{\text{init}}$ avoid the unit hypercube centered at $x_{\text{avoid}}$. We do not want to write all the 10,000 elements of the system matrix and the 100 coordinates of $x_{\text{init}}$ and $x_{\text{avoid}}$. Therefore, we provide the MATLAB code necessary to reproduce the setup in the following:

```
rand('state',1); \% Set state of random generator
randn('seed',1); \% Set seed of random generator
SYS = rss(100,0); \% Generate 100 dimensional random state space model
rand('state',2); \% Set state of random generator
randn('seed',2); \% Set seed of random generator
x_init = 10*rand(1,100);\% Pick 100 random numbers from uniform dist.
rand('state',3); \% Set state of random generator
randn('seed',3); \% Set seed of random generator
x_avoid = 8*rand(1,100); \% Pick 100 random numbers from uniform dist.
```

Before verifying a system of this size, it would be appropriate to apply some model order reduction techniques [5] to obtain a lower order model. However, to demonstrate that it is possible to verify such systems by the proposed method, the dimension is kept at 100; hence, we use 100 Lyapunov functions.

It is not reasonable to just generate one automaton with a location for each cell of the partition, as the automaton would have billions of locations. Therefore, we generate one timed automaton per Lyapunov function (i.e. 100 timed automata). The automata are analyzed separately, giving a time interval in which a solution may be in $x_{\text{avoid}}$. If the conjunction of all these time intervals is empty, then no trajectory reaches $x_{\text{avoid}}$. In the considered example, this time interval is empty; hence, all trajectories avoid $x_{\text{avoid}}$.

## 9 Conclusion

In this paper, a method for abstracting dynamical systems by timed automata is proposed. The abstraction is based on partitioning the state space of the dynamical systems by set-differences of positively invariant sets.

To enable both verification and falsification of safety of the considered system based on the abstraction, conditions for soundness, completeness, and refinability are derived. Furthermore, it has been demonstrated that the abstraction can be obtained as a parallel composition of multiple timed automata under certain conditions.

Via algorithms, based on LMI-based optimization problems, it is shown how the conditions for the abstraction can be checked and how time information for the timed automaton can be generated for linear systems.

It is shown that there exist sound and refinable abstractions for hyperbolic Morse-Smale systems and complete and refinable abstractions for all hyperbolic linear systems. Finally, an example of a sound and a complete abstraction is provided and their reachable sets are compared.

In the presented work, it is seen that abstractions generated by partitioning the state space of a system with the use of Lyapunov functions can be designed to be sound, complete, and refinable. Furthermore, a priori an upper bound on the over-approximation of the reachable set introduced by the abstraction can be calculated. Finally, an example shows that the method is applicable for high-dimensional systems, due to its compositionality.

## Appendix A: Definitions

**Definition 32** (Morse-Smale system [29]) A smooth vector field $X \in \mathfrak{X}^r(M)$ will be called a Morse-Smale system (or field) provided it satisfies the following conditions:

1. $X$ has a finite number of singular points, say $\beta_1, \ldots, \beta_k$, each hyperbolic. A hyperbolic singular point is a singular point such that in local coordinates the matrix of partial derivatives of $X$ has eigenvalues with nonzero real parts.

2. $X$ has a finite number of closed orbits (periodic solutions), say $\beta_{k+1}, \ldots, \beta_n$, each hyperbolic.
3. For any $p \in M$, $\alpha(p) = \beta_i$ and $\omega(p) = \beta_j$ for some $i$ and $j$.
4. Let $\Omega(X)$ be the nonwandering[1] points for $X$, then $\Omega(X) = \{\beta_1, \ldots, \beta_N\}$.
5. The stable and unstable manifolds associated with the $\beta_i$ have transversal intersection.

**Definition 33** (Topologically equivalent vector fields [30]) Two vector fields $\xi, \eta \in \mathfrak{X}^r(M)$ are topologically equivalent if there exists a homeomorphism $h : M \to N$ ($h$ is continuous and has continuous inverse) such that

1. $h \circ \phi_\xi(\mathbb{R}, x_0) = \phi_\eta(\mathbb{R}, h(x_0))$ for each $x_0 \in M$,
2. $h$ preserves the orientation, that is if $x_0 \in M$ and $\delta > 0$ there exists $\epsilon > 0$ such that, for $0 < t < \delta$, $h \circ \phi_\xi(t, x_0) = \phi_\eta(\tau, h(x_0))$ for some $0 < \tau < \epsilon$.

*Remark 5* Two vector fields are topologically conjugate if $t = \tau$ in the previous definition.

From Definition 33, we see that the solution of a vector field $\eta$ from some initial state can be described by a continuous deformation (the homeomorphism $h$) of the solution to a topologically equivalent vector field $\xi$. For a more formal explanation, see Proposition 4.1 in [19].

**Definition 34** (Regular value [28]) Let $f : N \to M$ be a smooth map. A point $p$ in $N$ is a regular point if the differential

$$f_{*,p} : T_p N \to T_{f(p)} M \tag{59}$$

is surjective. A point in $M$ is a regular value if it is the image of a regular point.

**Definition 35** (Timed-abstracted bisimulation [12]) Let $\mathcal{A} = (L, L_0, C, \Sigma, I, \Delta)$ be a timed automaton. A reflexive and symmetric relation $R \subseteq L \times \mathbb{R}^C \times L \times \mathbb{R}^C$ is a time-abstracted bisimulation if for all $(l_1, v_1)R(l_2, v_2)$, (Note that we denote $(l_1, v_1, l_2, v_2) \in R$ by $(l_1, v_1)R(l_2, v_2)$)

- for all $(l_1, v_1) \xrightarrow{d} (l_1', v_1')$ there exists $(l_1', v_1')R(l_2', v_2')$ for which $(l_2, v_2) \xrightarrow{d} (l_2', v_2')$, and
- for all $(l_1, v_1) \xrightarrow{\sigma} (l_1', v_1')$, $\sigma \in \Sigma$, there exists $(l_1', v_1')R(l_2', v_2')$ for which $(l_2, v_2) \xrightarrow{\sigma} (l_2', v_2')$.

We say $\mathcal{A}_1$ and $\mathcal{A}_2$ are bisimilar if there exists a time-abstracted bisimulation $R$ for $(\mathcal{A}_1, \mathcal{A}_2)$.

## Appendix B: Proofs

*Proof of Proposition 1* If $e_{(g,h)} \cap (\varphi^i)^{-1}(a^i_{g_i - 1})$ is not connected for some $i$, then $\sigma^i$ is the label of multiple outgoing transitions from the location $e_{(g,h)}$, i.e., there exist multiple transitions in $\Delta$, where $e_{(g,h)}$ is the source location and $\sigma^i$ is the label. Therefore, the timed automaton $\mathcal{A}(\mathcal{S})$ is nondeterministic. □

[1] We say that $p \in M$ is a wandering point for $X$ if there exists a neighborhood $V$ of $p$ and a number $t_0$ such that $\phi_X(t, V) \cap V = \emptyset$ for $|t| > t_0$.

*Proof of Proposition 4* Consider the timed automaton $\mathcal{A}_\parallel(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1)\| \ldots \|\mathcal{A}_k(\mathcal{S}^k)$ where $\mathcal{A}_i(\mathcal{S}^i) = (L_i, L_{0,i}, C_i, \Sigma_i, I_i, \Delta_i)$ and $L_i = \{l_1^i, \ldots, l_{|\mathcal{S}_i|}^i\}$, abstracting the slices $S_1^i, \ldots, S_{|\mathcal{S}_i|}^i$. Then the timed automaton $\mathcal{A}_\parallel(\mathcal{S})$ is given by

**Locations:** $L = L_1 \times \cdots \times L_k$, which according to Definition 17 represents extended cells, if the transversal intersection of all slices is nonempty i.e. (31) is satisfied.
**Clocks:** $C = \{c^i, \ldots, c^k\}$, where $c^i$ monitors the time for being in a slice of $\mathcal{S}^i$.
**Invariants:** The invariant for location $l_{\text{ex},g} = (l_{g_1}^1, \ldots, l_{g_k}^k)$ is identical to (22) and is

$$I(l_{\text{ex},g}) = \bigwedge_{i=1}^{k} I_i(l_{g_i}^i). \tag{60}$$

**Input Alphabet:** $\Sigma = \{\sigma^1, \ldots, \sigma^k\}$.
**Transition relations:** $\Sigma_i$ is disjoint from $\Sigma_j$ for all $i \neq j$; hence, item (1) in Definition 23 never happens.

This implies that $\mathcal{A}_\parallel(\mathcal{S}) = \mathcal{A}_1(\mathcal{S}^1)\| \ldots \|\mathcal{A}_k(\mathcal{S}^k)$ and $\mathcal{A}_{\text{ex}}(\mathcal{S})$ are isomorph. □

*Proof of Proposition 6* If the locations of $\mathcal{A}_{\text{ex}}$ are extended cells, then soundness of $\mathcal{A}_{\text{ex}}$ can be reformulated to the following.

A timed automaton $\mathcal{A}_{\text{ex}}$ with $L_0 = \{e_{\text{ex},g} | g \in \mathcal{G}_0 \subseteq \mathcal{G}\}$ is said to be a sound abstraction of $\Gamma$ with $X_0 = \bigcup_{g\in\mathcal{G}_0} e_{\text{ex},g}$ on $[t_1, t_2]$ if for all $t \in [t_1, t_2]$ and for all $g \in \mathcal{G}$

$$\bigcap_{i=1}^{k} S_{g_i}^i \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \quad \text{implies} \tag{61a}$$

$$\exists l_0 \in L_0 \quad \text{such that}$$

$$\bigcap_{i=1}^{k} S_{g_i}^i \in \alpha_K^{-1}(\phi_{\mathcal{A}_{\text{ex}}}(t, l_0)) \tag{61b}$$

which is equivalent to: For all $i \in \boldsymbol{k}$, all $g \in \mathcal{G}$, and for all $t \in [t_1, t_2]$

$$S_{g_i}^i \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \quad \text{implies} \tag{62a}$$

$$\exists l_{0,i} \in L_{0,i} \quad \text{such that}$$

$$\alpha_K^{-1}(\phi_{\mathcal{A}_i}(t, l_{0,i})) = S_{g_i}^i. \tag{62b}$$

From (62) it is seen that $\mathcal{A}_{\text{ex}} = \mathcal{A}_1\| \ldots \|\mathcal{A}_k$ is sound if and only if $\mathcal{A}_i$ is sound for $i \in \boldsymbol{k}$. Similar arguments can be used to prove the completeness part of Proposition 10. □

*Proof of Proposition 7* Let $e_{(g,h)}$ with $h \in \boldsymbol{m}$ be the cells which union is the extended cell $e_{\text{ex},g}$. Then

$$I(e_{(g,h)}) = I(e_{(g,k)}) \quad \forall h, k \in \boldsymbol{m} \tag{63}$$

as the invariants are calculated based on slices (22).

If the partition satisfies (36), then the same outgoing transitions exist for all cells within the same extended cell. Furthermore,

$$G_{(g,h)\to(g',h')} = G_{(g,k)\to(g',k')} \quad \forall h, k \in \boldsymbol{m} \tag{64}$$

since the guards are also calculated based on slices (19b) in [26]. This implies that all possible behaviors from each cell in an extended cell are the same; hence, $\mathcal{A}(\mathcal{S})$ is bisimilar to a timed automaton $\mathcal{A}_{\text{ex}}(\mathcal{S})$. □

*Proof of Proposition 11* Let $\mathcal{A}(\mathcal{S})$ be a timed automaton with $L_0 = \{e_i | i \in \mathcal{I}\}$, be an abstraction of $\Gamma$ with initial set $X_0 = \bigcup_{i\in\mathcal{I}} e_i$. If guards and invariants of $\mathcal{A}(\mathcal{S})$ satisfy (41), then

$$\text{Reach}_{[t_1,t_2]}(\Gamma, X_0) \subseteq \alpha_K^{-1}(\text{Reach}_{[t_1,t_2]}(\mathcal{A}, L_0)) \tag{65}$$

since for all $x_0 \in (\varphi^i)^{-1}(a^i_{g_i})$ there exists $t \in [\underline{t}_{S^i_{g_i}}, \overline{t}_{S^i_{g_i}}]$ such that

$$\phi_\Gamma(t, x_0) \in (\varphi^i)^{-1}(a^i_{g_i-1}). \tag{66}$$

□

*Proof of Proposition 12* The proposition states that it takes the same time for all trajectories of $\Gamma$ to propagate between any two level sets of $\varphi^i$. From this it follows that $\mathcal{A}(\mathcal{S})$ is complete if $\overline{t}_{S^i_{g_i}}$ and $\underline{t}_{S^i_{g_i}}$ are equal to $t^i_{g_i}$. □

*Proof of Proposition 13* If $\mathcal{A}(\mathcal{S})$ is a refinable abstraction, then for any $\epsilon > 0$ there exists a partitioning $K(\mathcal{S})$ such that (30) in [26] holds for cells in $K(\mathcal{S})$. Therefore,

$$S^i_{g_i} \subset (\varphi^i)^{-1}(a^i_{g_i}) + B(\epsilon) \tag{67}$$

where $\epsilon > 0$. Note that $a^i_{g_i}$ is a regular value of $\varphi^i$, i.e., the dimension of the level set $(\varphi^i)^{-1}(a^i_{g_i})$ is $n-1$. The locations of $\mathcal{A}(\mathcal{S})$ are cells for which

$$\bigcup_h e_{(g,h)} = \pitchfork_{i=1}^k S^i_{g_i} \tag{68a}$$

$$\subset \pitchfork_{i=1}^k \left(\varphi_i^{-1}(a^i_{g_i}) + B(\epsilon)\right) \tag{68b}$$

$$\subset \pitchfork_{i=1}^k \varphi_i^{-1}(a^i_{g_i}) + B(2\epsilon). \tag{68c}$$

But (68c) is true for any $\epsilon$, thus it is enough to prove that

$$\dim\left(\pitchfork_{i=1}^k (\varphi^i)^{-1}(a^i_{g_i})\right) = 0. \tag{69}$$

Using Theorem 7.7 in [9] the dimension of an extended cell is given by

$$\dim\left(\pitchfork_{i=1}^k (\varphi^i)^{-1}(a^i_{g_i})\right) = [[(n-1) + (n-1) - n] + (n-1) - n \cdots \tag{70a}$$

$$= k(n-1) - (k-1)n. \tag{70b}$$

We see that if $k \neq n$ then $\dim(\pitchfork_{i=1}^k (\varphi^i)^{-1}(a^i_{g_i})) \neq 0$, thus we have contradiction. We conclude that $k = n$. □

*Proof of Proposition 14* Let $S(n, \mathbb{R})$ be a set of $n \times n$ symmetric matrices. $S(n, \mathbb{R})$ is a subspace of $M(n, \mathbb{R})$ of $\dim(S(n, \mathbb{R})) = n(n+1)/2$. Consider the map $\varphi_A : S(n, \mathbb{R}) \to S(n, \mathbb{R})$ and let

$$P \mapsto A^{\mathrm{T}} P + PA. \tag{71}$$

Now consider the map $\det : M(n, \mathbb{R}) \to \mathbb{R}$ and let

$$A \mapsto \det(A). \tag{72}$$

Then $(\det \circ \varphi_A)^{-1}(\{0\})$ is a closed set. Therefore,

$$U_A \equiv \{P \in S(n, \mathbb{R}) | \det \circ \varphi_A(P) \neq 0\} \tag{73}$$

is an open set. $V_A \equiv V \cap U_A$ is open, where

$$V = \{P \in S(n, \mathbb{R}) | \det(P) \neq 0\} \quad (V \text{ is open}). \tag{74}$$

Let $\Theta = \{Q \in S(n, \mathbb{R}) | Q > 0\}$ by Proposition 2.18 in [19] the map

$$M(n, \mathbb{R}) \to C^n / S^n \quad \text{defined by} \tag{75}$$

$$L \mapsto \operatorname{diag}([\lambda_1, \ldots, \lambda_n]) \quad \text{is continuous.} \tag{76}$$

Thus $\Theta$ is an open set in $S(n, \mathbb{R})$.

We pick an open neighborhood around $Q = A^{\mathrm{T}}P + PA$ and denote it $U$. Then for every $Q' \in U$ there exists a (unique) $P$, thus $\varphi_A^{-1}(U)$ is a nonempty open set in $S(n, \mathbb{R})$.

We can pick $n$ linear independent matrices $P_1, \ldots, P_n \in \varphi_A^{-1}(U)$. This is possible because $\varphi_A^{-1}(U)$ is open in $S(n, \mathbb{R})$ and $\dim(S(n, \mathbb{R}))$ is $n(n+1)/2$. Then for any $a \in \mathbb{R} \backslash \{0\}$ and $i \neq j$

$$\{x \in \mathbb{R}^n | x^{\mathrm{T}} P_i x = a\} \pitchfork \{x \in \mathbb{R}^n | x^{\mathrm{T}} P_j x = a\}. \tag{77}$$

Extending this to Morse-Smale systems follows directly from Theorem 1 in [23]. □

*Proof of Proposition 15* Let $\varphi(x)$ be a Lyapunov function for the system $\Gamma$ and let $x, x' \in \varphi^{-1}(a_m)$. According to Proposition 12 the abstraction is complete if there exists a $t_m$, for $m = 2, \ldots, k$ such that

$$\phi_\Gamma(t_m, x), \phi_\Gamma(t_m, x') \in \varphi^{-1}(a_{m-1}). \tag{78}$$

This is true if

$$\dot{\varphi}(\phi_\Gamma(t, x)) - \dot{\varphi}(\phi_\Gamma(t, x')) = 0 \quad \forall t. \tag{79}$$

The combination of (78) and (79) implies that for all $c > 0$ there exists an $\alpha$ such that

$$\varphi^{-1}(c) = \dot{\varphi}^{-1}\left(\frac{c}{\alpha}\right) \tag{80}$$

hence for all $x$ there exists an $\alpha$ such that

$$\varphi(x) = \alpha \dot{\varphi}(x). \tag{81}$$

□

*Proof of Proposition 16* This is proved for linear systems, by constructing the complete abstraction.

Consider a linear differential equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 I_1 & 0 \\ 0 & \lambda_2 I_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{82}$$

where $I_1$, $I_2$ are identity matrices and $\lambda_1 < 0$ and $\lambda_2 > 0$.

The stable and unstable subspaces of (82) are orthogonal and can be treated separately. This system is divided into a stable space described by $x_1$ and an unstable space described by $x_2$. For $i \in \{1, 2\}$ let $\varphi_i(x_i) = x_i^{\mathrm{T}} P_i x_i$ be a quadratic Lyapunov function. Then its derivative is $\dot{\varphi}(x_i) = x_i^{\mathrm{T}} Q_i x_i$, where

$$2\lambda_i P_i = Q_i \quad \text{for } i = 1, 2. \tag{83}$$

This implies that any quadratic Lyapunov function satisfies Proposition 15 and hence generates a complete abstraction.

Since hyperbolic linear systems are topologically conjugate if and only if they have the same index [18]. There is a homeomorphism $h : \mathbb{R}^n \to \mathbb{R}^n$ such that any hyperbolic linear system is topologically conjugate of (82), by choosing $I_1$ and $I_2$ appropriately. Note that $h$ is a diffeomorphism on $\mathbb{R}^n \backslash \{0\}$.

This implies that there exists a complete abstraction of every hyperbolic linear system. □

## References


1. Alfaro LD, Henzinger TA, Majumdar R (2001) Symbolic algorithms for infinite-state games. In: Proceedings of the 12th international conference on concurrency theory, Aalborg, Denmark, pp 536–550. doi:10.1007/3-540-44685-0_36
2. Alur R, Courcoubetis C, Dill D (1990) Model-checking for real-time systems. In: Proceedings of the fifth annual IEEE symposium on logic in computer science, Philadelphia, PA, USA, pp 414–425. doi:10.1109/LICS.1990.113766
3. Alur R, Dang T, Ivančic F (2003) Progress on reachability analysis of hybrid systems using predicate abstraction. In: Proceedings of the 6th international conference on hybrid systems: computation and control, Prague, Czech Republic, pp 4–19
4. Alur R, Dill DL (1994) A theory of timed automata. Theor Comput Sci 126(2):183–235. doi:10.1016/0304-3975(94)90010-8
5. Antoulas AC (2005) Approximation of large-scale dynamical systems. Advances in design and control. SIAM, Philadelphia
6. Asarin E, Dang T, Frehse G, Girard A, Guernic CL, Maler O (2006) Recent progress in continuous and hybrid reachability analysis. In: Proceedings of the 2006 IEEE conference on computer aided control systems design, Munich, Germany, pp 1582–1587. doi:10.1109/CACSD-CCA-ISIC.2006.4776877
7. Behrmann G, David A, Larsen KG (2004) A tutorial on Uppaal. In: Formal methods for the design of real-time systems. Lecture notes in computer science, vol 3185. Springer, Berlin, pp 200–236. doi:10.1007/978-3-540-30080-9_7
8. Boyd S, Ghaoui LE, Feron E, Balakrishnan V (1994) Linear matrix inequalities in system and control theory. SIAM studies in applied mathematics, vol 15. SIAM, Philadelphia
9. Bredon GE (1993) Topology and geometry. Springer, Berlin
10. Broucke M (1998) A geometric approach to bisimulation and verification of hybrid systems. In: Proceedings of the 37th IEEE conference on decision and control, Tampa, FL, USA, pp 4277–4282. doi:10.1109/CDC.1998.761977
11. Clarke F, Ledyaev Y, Stern R, Wolenski P (1998) Nonsmooth analysis and control theory. Springer, Berlin
12. Fahrenberg U, Larsen K, Thrane C (2010) Verification, performance analysis and controller synthesis for real-time systems. In: Fundamentals of software engineering. Lecture notes in computer science, vol 5961. Springer, Berlin, pp 34–61. doi:10.1007/978-3-642-11623-0_2
13. Ghosh R, Tomlin C (2004) Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-Notch protein signalling. Syst Biol 1(1):170–183

14. Girard A (2005) Reachability of uncertain linear systems using zonotopes. In: Proceedings of the 8th international conference on hybrid systems: computation and control, Zurich, Switzerland, pp 291–305
15. Girard A, Pola G, Tabuada P (2010) Approximately bisimilar symbolic models for incrementally stable switched systems. IEEE Trans Autom Control 55(1):116–126. doi:10.1109/TAC.2009.2034922
16. Guéguen H, Lefebvre MA, Zaytoon J, Nasri O (2009) Safety verification and reachability analysis for hybrid systems. Annu Rev Control 33(1):25–36. doi:10.1016/j.arcontrol.2009.03.002
17. Hirsch MW (1976) Differential topology. Springer, Heidelberg
18. Hirsch MW, Smale S, Devaney RL (2004) Differential equations, dynamical systems & an introduction to chaos, 2nd edn. Elsevier, Amsterdam
19. Junior JP, de Melo W (1980) Geometric theory of dynamical systems: an introduction. Springer, Berlin
20. Kurzhanski AB, Vályi I (1997) Ellipsoidal calculus for estimation and control. Birkhäuser, Boston
21. Maler O, Batt G (2008) Approximating continuous systems by timed automata. In: Proceedings of the 1st international workshop on formal methods in systems biology, Cambridge, UK, pp 77–89
22. Matsumoto Y (2002) An introduction to Morse theory. American Mathematical Society, Providence
23. Meyer KR (1968) Energy functions for Morse Smale systems. Am J Math 90(4):1031–1040
24. Morse M, Hedlund GA (1938) Symbolic dynamics. Am J Math 60(4):815–866
25. Peixoto MM (1962) Structural stability on two-dimensional manifolds. Topology 1(2):101–120
26. Sloth C, Wisniewski R (2010) Abstraction of continuous dynamical systems utilizing Lyapunov functions. In: Proceedings of the 49th IEEE conference on decision & control, Atlanta, Georgia USA, pp 3760–3765
27. Tiwari A (2008) Abstractions for hybrid systems. Form Methods Syst Des 32(1):57–83. doi:10.1007/s10703-007-0044-3
28. Tu LW (2008) An introduction to manifolds. Springer, Berlin
29. Wisniewski R (2005) Flow lines under perturbations within section cones. PhD thesis, Department of Mathematical Sciences, Aalborg University
30. Wisniewski R, Raussen M (2007) Geometric analysis of nondeterminacy in dynamical systems. Acta Inform 43(7):501–519. doi:10.1007/s00236-006-0037-5
31. Yazarel H, Pappas GJ (2004) Geometric programming relaxations for linear system reachability. In: Proceedings of the 2004 American control conference, Boston, MA, USA, pp 553–559. doi:10.1109/ACC.2004.182304