Aalborg Universitet

Algorithmic Approach to Abstracting Linear Systems by Timed Automata

Sloth, Christoffer; Wisniewski, Rafael

Published in:
I F A C Workshop Series

DOI (link to publication from Publisher):
10.3182/20110828-6-IT-1002.02568

Publication date:
2011

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

Link to publication from Aalborg University

Citation for published version (APA):
Sloth, C., & Wisniewski, R. (2011). Algorithmic Approach to Abstracting Linear Systems by Timed Automata. *I F A C Workshop Series*, 4546-4551. https://doi.org/10.3182/20110828-6-IT-1002.02568

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: December 06, 2025

Algorithmic Approach to Abstracting Linear Systems by Timed Automata [⋆]

Christoffer Sloth [*] Rafael Wisniewski [**]

[*] *Department of Computer Science, Aalborg University, 9220 Aalborg East, Denmark (e-mail: csloth@cs.aau.dk).*
[**] *Section of Automation & Control, Aalborg University, 9220 Aalborg East, Denmark (e-mail: raf@es.aau.dk)*

Abstract: This paper proposes an LMI-based algorithm for abstracting dynamical systems by timed automata, which enables automatic formal verification of linear systems.
The proposed abstraction is based on partitioning the state space of the system using positive invariant sets, generated by Lyapunov functions. This partitioning ensures that the vector field of the dynamical system is transversal to all facets of the cells, which induces some desirable properties of the abstraction.
The algorithm is based on identifying intersections of level sets of quadratic Lyapunov functions, and determining the minimum and maximum time that a trajectory of the system can stay in a set, defined as the set-difference of sub-level sets of Lyapunov functions. The proposed algorithm applies for linear systems and can therefore be efficiently implemented using LMI-based tools.

1. INTRODUCTION

The verification of system properties such as safety is based on reachability calculations or approximations. Since the exact reachable sets of continuous and hybrid systems in general are incomputable Asarin et al. [2006], a lot of attention has been paid to their approximations. Yet, reachability is decidable for system models such as automata and timed automata; consequently, there exists a rich set of tools aimed at verifying properties of such systems, e.g., UPPAAL, see Behrmann et al. [2004]. Therefore, abstracting a dynamical system by this type of system model would enable its verification.

There exist different methods for verifying continuous and hybrid systems. One of these methods is to over-approximate the reachable states of a system by convex sets as in Kurzhanski and Vlyi [1997], where the sets are ellipsoids and in Girard [2005] using zonotopes. Other methods abstract systems with models of reduced complexity, while preserving certain properties of the original systems. This is accomplished for hybrid systems in Tiwari [2008] and for continuous systems in Maler and Batt [2008]. The class of verification methods presented in Tiwari [2008], Prajna [2006] is close to the presented method. The core of these methods is to generate a positive invariant set that includes the initial set and excludes the unsafe sets. If such a set exists, then no solution trajectory initialized in the initial set reaches the unsafe sets.

Among the tools for the verification of continuous and hybrid systems is PHAVer. This tool is capable of verifying safety properties of systems with piecewise affine dynamics, via over-approximation of the reachable set by polyhedra Frehse [2005]. The disadvantage of this tools is scalability, as it needs to generate a lot of polyhedra to approximate the reachable set of a system. Furthermore, the verification relies on simulations of trajectories of the system; this is also computationally expensive.

In this paper, linear systems are abstracted by timed automata using the method presented in Sloth and Wisniewski [2010]. This method is based on partitioning the state space using level sets of Lyapunov functions; hence, the partitioning is conducted according to the dynamics of the system. In Sloth and Wisniewski [2010] no constructive method for generating a timed automaton was provided; hence, this paper presents an LMI-based approach for generating a timed automaton via quadratic Lyapunov functions for linear systems. In contrast to Frehse [2005], the proposed abstraction procedure does not use solutions to the system equations, which makes the method less computationally demanding, and in contrast to Tiwari [2008], Prajna [2006], we generate timed models.

The generated abstraction makes it possible to verify requirements in terms of timed temporal logic specifications Alur et al. [1990]; hence, requirements to reachability and timing can be added to the usual stability requirement.

This paper is organized as follows. Section 2 contains preliminary definitions utilized throughout the paper, Section 3 explains the partitioning of the state space, and Section 4 describes how a timed automaton can be generated from the partition. Section 5 presents algorithms for synthesizing the abstraction, and an example is provided in Section 6. Finally, Section 7 comprises conclusions.

1.1 Notation

The set $\{1, \dots, k\}$ is denoted $\boldsymbol{k}$. B^A is the set of maps $A \to B$. The power set of A is denoted 2^A. The cardinality of the set A is denoted $|A|$. We consider the Euclidean space $(\mathbb{R}^n, \langle , \rangle)$, where $\langle , \rangle$ is the scalar product.

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

Copyright by the
International Federation of Automatic Control (IFAC)

2. PRELIMINARIES

The purpose of this section is to provide definitions related to dynamical systems and timed automata.

A dynamical system $\Gamma = (X, f)$ has state space $X \subseteq \mathbb{R}^n$ and dynamics described by ordinary differential equations $f : X \to \mathbb{R}^n$

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

In this paper, we assume that (1) is linear; hence,

$$\dot{x} = Ax \tag{2}$$

where $A \in \mathbb{R}^{n \times n}$ is a non-singular matrix.

The solution of (2), from an initial state $x_0 \in X_0 \subseteq X$ at time $t \geq 0$ is described by the flow function $\phi_\Gamma : [0, \epsilon] \times X \to X$, $\epsilon > 0$ satisfying

$$\frac{d\phi_\Gamma(t, x_0)}{dt} = f\left(\phi_\Gamma(t, x_0)\right) \tag{3}$$

for all $t \geq 0$.

Lyapunov functions are utilized in stability theory and are defined in the following Meyer [1968].

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

p is a critical point of $f \Leftrightarrow p$ is a critical point of φ

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

$$\dot{\varphi}(x) = 0 \;\; \forall x \in \mathrm{Cr}(f) \tag{4b}$$

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

and there exists $\alpha > 0$ and an open neighborhood of each critical point $p \in \mathrm{Cr}(f)$, 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 φ, i.e., $\dot{\varphi}(x) = \langle \nabla\varphi(x), f(x) \rangle < 0$ for all $x \in X \backslash \mathrm{Cr}(f)$, 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 Meyer [1968]. This makes it possible to abstract unstable systems.

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

$$\begin{aligned} \mathrm{Reach}_{[t_1,t_2]}(\Gamma, X_0) \equiv \{x \in X | \exists t \in [t_1, t_2],\, \exists x_0 \in X_0 \\ \text{such that } x = \phi_\Gamma(t, x_0)\}. \end{aligned} \tag{6}$$

The dynamical system will be abstracted by a timed automaton Alur and Dill [1994]. In the definition of a timed automaton, a set of clock constraints $\Psi(C)$ is used for the set C of clocks. $\Psi(C)$ is defined as the set of constraints ψ described by the following grammar:

$$\begin{aligned} &\psi ::= c_1 \bowtie k | c_1 - c_2 \bowtie k | \psi_1 \wedge \psi_2, \text{ where} \\ &c_1, c_2 \in C,\ k \in \mathbb{R}_{\geq 0}, \text{ and } \bowtie \in \{\boldsymbol{\leq}, \boldsymbol{<}, \boldsymbol{=}, \boldsymbol{>}, \boldsymbol{\geq}\}. \end{aligned} \tag{7}$$

Note that the clock constraint k should usually be an integer, but in this paper no effort is done to convert the clock constraints into integers. Furthermore, the elements of $\bowtie$ are bold to indicate that they are syntactic operations.

Definition 3. (Timed Automaton). A timed automaton, $\mathcal{A}$, is a tuple $(E, E_0, C, \Sigma, I, \Delta)$, where

- E is a finite set of locations, and $E_0 \subseteq E$ is the set of initial locations.
- C is a finite set of clocks.
- Σ is the set of actions.
- $I : E \to \Psi(C)$ assigns invariants to locations.
- $\Delta \subseteq E \times \Psi(C) \times \Sigma \times 2^C \times E$ is a finite set of transition relations. The transition relations provide edges between locations as tuples $(e, G_{e \to e'}, \sigma, R_{e \to e'}, e')$, where e is the source location, e' is the destination location, $G_{e \to e'} \in \Psi(C)$ is the guard set, σ is an action in Σ, and $R_{e \to e'} \in 2^C$ is the set of clocks to be reset.

The semantics of a timed automaton is defined in the following, adopting the notion of Fahrenberg et al. [2009].

Definition 4. (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, $t \in \mathbb{R}_{\geq 0}$, and $R \subseteq C$, the valuations $v + t$ and $v[R]$ are defined as follows

$$(v + t)(c) = v(c) + t, \tag{8a}$$

$$v[R](c) = \begin{cases} 0 & \text{for } c \in R, \\ v(c) & \text{otherwise.} \end{cases} \tag{8b}$$

We see that (8a) is used to progress time and that (8b) is used to reset the clocks in the set R to zero.

Definition 5. (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

$$\llbracket c \boldsymbol{\bowtie} k \rrbracket = \{v : C \to \mathbb{R}_{\geq 0} | v(c) \bowtie k\} \tag{9a}$$

$$\llbracket \psi_1 \wedge \psi_2 \rrbracket = \llbracket \psi_1 \rrbracket \cap \llbracket \psi_2 \rrbracket. \tag{9b}$$

For convenience we denote $v \in \llbracket \psi \rrbracket$ by $v \models \psi$.

Definition 6. (Semantics of Timed Automaton). The semantics of a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ is a transition system $\llbracket \mathcal{A} \rrbracket = (S, S_0, \Sigma \cup \mathbb{R}_{\geq 0}, T_\sigma \cup T_\mathrm{t})$, where

$$\begin{aligned} S &= \{(e, v) \in E \times \mathbb{R}^C_{\geq 0} | v \models I(e)\} \\ S_0 &= \{(e, v) \in E_0 \times v_0\} \\ T_\sigma &= \{(e, v) \xrightarrow{\sigma} (e', v') | \exists (e, G_{e \to e'}, \sigma, R_{e \to e'}, e') \in \Delta : \\ &\qquad v \models G_{e \to e'},\ v' = v[R_{e \to e'}]\} \\ T_\mathrm{t} &= \{(e, v) \xrightarrow{t} (e, v + t) | \forall t' \in [0, t] : v + t' \models I(e)\}. \end{aligned}$$

Analog to the solution of (2) is a run of a timed automaton.

Definition 7. (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

$$\varrho_\mathcal{A} : (e_0, v_0) \xrightarrow{t_1} (e_0, v_1) \xrightarrow{\sigma_1} (e_1, v_2) \xrightarrow{t_2} \ldots \tag{10}$$

which is a path in $\llbracket \mathcal{A} \rrbracket$, where $t_i \in \mathbb{R}_{\geq 0}$ and $\sigma_i \in \Sigma$. The multifunction describing the runs of a timed automaton $\phi_\mathcal{A} : \mathbb{R}_{\geq 0} \times E_0 \to 2^E$, is defined by $e \in \phi_\mathcal{A}(t, e_0)$ if and only if there exists a path in $\llbracket \mathcal{A} \rrbracket$ initialized in (e_0, v_0) that reaches the location e at time $t = \sum_i t_i$.

Definition 8. (Reachable set of Timed Automaton). The reachable set of a timed automaton $\mathcal{A}$, with initial locations E_0, in the time interval $[t_1, t_2]$ is defined as

$$\begin{aligned} \mathrm{Reach}_{[t_1,t_2]}(\mathcal{A}, E_0) \equiv \{e \in E | \exists t \in [t_1, t_2], \exists e_0 \in E_0 \\ \text{such that } e \in \phi_\mathcal{A}(t, e_0)\}. \end{aligned} \tag{11}$$

3. GENERATION OF FINITE PARTITION

The utilized abstraction is based on partitioning the state space of Γ into a finite number of cells, generated by intersecting slices defined as the set-difference of positive invariant sets Sloth and Wisniewski [2010].

Definition 9. (Slice). A nonempty set S is a slice if there exist two sets A and B such that

(1) B is a proper subset of A, i.e., $B \subset A$.
(2) A and B are positively invariant,
(3) $S = \text{cl}(A \backslash B)$.

The slices are defined to be set-differences of positive invariant sets, to ensure that the flow of the system is transversal to the boundaries of the slices. This also ensures a nonzero minimum time for staying in each slice.

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

Definition 10. (Slice-Family). A slice-family $\mathcal{S}$ is a collection of slices generated by the positive invariant sets $A_0 \subset A_1 \subset \cdots \subset A_k$ covering the entire state space of Γ. By convention the first positive invariant set is $A_0 = \emptyset$. Thereby $S_1 = \text{cl}(A_1 \backslash A_0), \ldots, S_k = \text{cl}(A_k \backslash A_{k-1})$ and $X \subseteq A_k$. For convenience $|\mathcal{S}|$ is defined to be the cardinality of the slice-family $\mathcal{S}$, thus $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$.

We say that $\mathcal{S}$ is generated by the sets $\{A_i | i \in \boldsymbol{k}\}$. A function is associated to each slice-family $\mathcal{S}$, to provide an easy way of describing the boundaries of a slice.

Definition 11. (Partitioning Function). Let $\mathcal{S}$ be a slice-family, then a continuous function $\varphi : \mathbb{R}^n \to \mathbb{R}$ smooth on $\mathbb{R}^n \backslash \text{Cr}(f)$ is a partitioning function associated to $\mathcal{S}$ if for any positive invariant set A_i generating $\mathcal{S}$ there exists $a_i, a'_i \in \mathbb{R} \cup \{-\infty, \infty\}$ such that

$$\varphi^{-1}([a'_i, a_i]) = A_i \tag{12}$$

and a_i, a'_i are regular values of φ. By regular level set theorem, the boundary $\varphi^{-1}(a_i)$ of A_i is a smooth manifold Tu [2008].

Definition 12. (Transversal Intersection of Slices). We say that the slices S_1 and S_2 intersect transversally and write

$$S_1 \pitchfork S_2 = S_1 \cap S_2 \tag{13}$$

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

Definition 13. (Extended Cell). Let $\{\mathcal{S}^i | i \in \boldsymbol{k}\}$ be a collection of k slice-families and define $\mathcal{G} = \{1, \ldots, |\mathcal{S}^1|\} \times \cdots \times \{1, \ldots, |\mathcal{S}^k|\}$. Denote the j^{th} slice in $\mathcal{S}^i$ by S^i_j and let $g \in \mathcal{G}$. Then

$$e_{\text{ex},g} = \pitchfork_{i=1}^{k} S^i_{g_i}. \tag{14}$$

Any nonempty set $e_{\text{ex},g}$ will be called an extended cell.

The cells in Definition 13 are denoted extended cells, since the intersection of slices may form multiple disjoint sets.

Proposition 1. (Sloth and Wisniewski [2010]).
If $S_1 \pitchfork S_2 \neq \emptyset$ then

$$\text{int}(S_1 \cap S_2) \neq \emptyset. \tag{15}$$

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

$$e_{\text{ex},[9,5,27]} = S^1_9 \pitchfork S^2_5 \pitchfork S^3_{27}. \tag{16}$$

Notice that the vector g from Definition 13 equals $[9, 5, 27]$.

Definition 14. (Cell). A cell is a connected component of an extended cell

$$\bigcup_h e_{(g,h)} = e_{\text{ex},g}, \text{ where} \tag{17a}$$

$$e_{(g,h)} \cap e_{(g,k)} = \emptyset \quad \forall h \neq k. \tag{17b}$$

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

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

Definition 15. (Finite Partition). Let $\mathcal{S}$ be a collection of slice-families, $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then the finite partition $K(\mathcal{S})$ is defined to be the collection of all cells generated by $\mathcal{S}$ according to Definition 14.

4. GENERATION OF TIMED AUTOMATON FROM FINITE PARTITION

To obtain a timed automaton $\mathcal{A}$ from a finite partition $K(\mathcal{S})$, the following abstraction procedure is used.

Procedure 1. Given a dynamical system $\Gamma = (X, f)$ and a partition $K(\mathcal{S})$, the timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ abstracting Γ is generated as follows.

- **Locations:** Let the locations of $\mathcal{A}$ be given by
$$E = K(\mathcal{S}). \tag{18}$$
This means that a location $e_{(g,h)}$ is associated to all states within the cell $e_{(g,h)}$ of the partition $K(\mathcal{S})$.
- **Clocks:** Given k slice-families, the number of clocks is k, i.e., $C = \{c^i | i \in \boldsymbol{k}\}$. The clock c^i monitors the time for being in the slices of the slice-family $\mathcal{S}^i$.
- **Invariants:** In each location $e_{(g,h)}$, there are up to k invariants, providing upper bounds on the time for staying in the k slices generating the cell $e_{(g,h)}$. We impose an invariant whenever there is an upper bound for the time for staying in a slice generating the cell
$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \bar{t}_{S^i_{g_i}} \tag{19}$$
where $\bar{t}_{S^i_{g_i}} \in \mathbb{R}_{\geq 0}$ is an upper bound on the time for staying in the slice $S^i_{g_i}$.
- **Input Alphabet:** The input alphabet Σ consists of symbols $\sigma^1, \ldots, \sigma^k$, where σ^i is associated with transitions between pairs of slices in $\mathcal{S}^i = \{S^i_1, \ldots, S^i_{|\mathcal{S}_i|}\}$.
- **Transition relations:** For every pair of locations, $e_{(g,h)}$ and $e_{(g',h')}$, satisfying the following conditions
(1) $e_{(g,h)}$ and $e_{(g',h')}$ are adjacent cells in the state space, i.e., $e_{(g,h)} \cap e_{(g',h')} \neq \emptyset$, and
(2) $g'_i \leq g_i$ for $i \in \boldsymbol{k}$
there is a transition relation
$$\delta_{(g,h)\to(g',h')} = (e_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma, R_{(g,h)\to(g',h')}, e_{(g',h')}), \tag{20a}$$
where
$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}_{S^i_{g_i}} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise} \end{cases} \tag{20b}$$
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 σ^i is taken.

Let $i = 1, \dots, k$. We define $R_{(g,h)\to(g',h')}$ by

$$c_i \in R_{(g,h)\to(g',h')} \tag{20c}$$

iff $g_i - g_i' = 1$.

For convenience the following notion is introduced.

Definition 16. Let $\mathcal{S}$ be a collection of slice-families, i.e., $\mathcal{S} = \{\mathcal{S}^i | i \in \boldsymbol{k}\}$. Then $\mathcal{A}(\mathcal{S})$ is the timed automaton generated by $\mathcal{S}$ according to (18)-(20c).

Remark 1. Nonetheless, the locations of $\mathcal{A}(\mathcal{S})$ are associated with cells of $K(\mathcal{S})$, we will also utilize the timed automaton $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$ with locations associated to extended cells, i.e.,

$$E = \{e_{\mathrm{ex},g} | g \in \mathcal{G}\}. \tag{21}$$

The other steps of the procedure are identical for the two timed automata $\mathcal{A}(\mathcal{S})$ and $\mathcal{A}_{\mathrm{ex}}(\mathcal{S})$.

5. METHOD

The purpose of this section is to set up an algorithm for synthesizing a timed automaton from a linear system and a family of quadratic Lyapunov functions, according to Procedure 1. First, an overall algorithm is presented and then an algorithm is provided for each of the steps in the algorithm.

Algorithm 1. Suppose $\Gamma = (X, f)$ is a linear system, $\{\mathcal{S}^i | i \in \boldsymbol{k}\}$ is a collection of slice-families, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i(x) = x^{\mathrm{T}} P^i x$ is associated with slice-family $\mathcal{S}^i$, and $\mathcal{S}^i$ is generated using the regular values, i.e., for all i, j $a^i_j \neq 0$, $\{a^i_j | j = 1, \dots, |\mathcal{S}^i|\}$. Then the timed automaton $\mathcal{A}(\mathcal{S})$ abstracting Γ can be synthesized as follows:

1. Define the set of clocks $C = \{c^i | i \in \boldsymbol{k}\}$ and the set of actions $\Sigma = \{\sigma^i | i \in \boldsymbol{k}\}$.
2. Generate the finite set of locations E, using Algorithm 2.
3. Determine the invariants of each location $e \in E$, using Algorithm 3.
4. Use Algorithm 4 to define transition relations between pairs of locations in E.

Steps 2-4 in Algorithm 1 need separate algorithms to be synthesized. The algorithms are based on intersections of level sets of quadratic forms Boyd et al. [1994].

Proposition 2. 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 optimization problem

$$\min \overline{\gamma} \text{ subject to} \tag{22a}$$
$$P^2 - \overline{\gamma} P^1 \leq 0 \tag{22b}$$
$$\overline{\gamma} > 0 \tag{22c}$$

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

$$\max \underline{\gamma} \text{ subject to} \tag{23a}$$
$$P^1 \underline{\gamma} - P^2 \leq 0 \tag{23b}$$
$$\underline{\gamma} > 0 \tag{23c}$$

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 $a_2 \in [a_1 \underline{\gamma}, a_1 \overline{\gamma}]$.

This is illustrated in Fig. 1, 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.

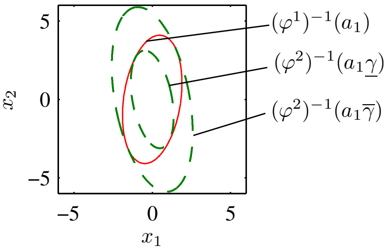

Fig. 1. Illustration of a level set $(\varphi^1)^{-1}(a_1)$ (red) and the level sets $(\varphi^2)^{-1}(a_1 \underline{\gamma})$, and $(\varphi^2)^{-1}(a_1 \overline{\gamma})$ (green and dashed) intersecting $(\varphi^1)^{-1}(a_1)$.

5.1 Generation of Locations

In this subsection, an algorithm is presented that generates the finite set of locations for the abstraction $\mathcal{A}(\mathcal{S})$. To generate the set of locations it follows from Definition 14 that we should check emptiness of (14) repeated below

$$e_{\mathrm{ex},g} = \pitchfork_{i=1}^{k} S^i_{g_i}. \tag{24}$$

This emptiness checking can be accomplished using Proposition 2, as shown in the following lemma. Note that we use strict inequalities to ensure transversal intersections.

Lemma 1. Suppose $P^i = (P^i)^{\mathrm{T}} > 0$, $P^{i'} = (P^{i'})^{\mathrm{T}} > 0$, and $\varphi^i(x) = x^{\mathrm{T}} P^i x$ and $\varphi^{i'}(x) = x^{\mathrm{T}} P^{i'} x$. Let $S^i_j = (\varphi^i)^{-1}([a^i_{j-1}, a^i_j])$, $S^{i'}_{j'} = (\varphi^{i'})^{-1}([a^{i'}_{j'-1}, a^{i'}_{j'}])$, and define $\overline{\gamma}^{ii'}$ and $\underline{\gamma}^{ii'}$ as the solutions to the optimization problems

$$\min \overline{\gamma}^{ii'} \text{ subject to} \tag{25a}$$
$$P^{i'} - \overline{\gamma}^{ii'} P^i < 0 \tag{25b}$$
$$\overline{\gamma}^{ii'} > 0. \tag{25c}$$

$$\max \underline{\gamma}^{ii'} \text{ subject to} \tag{26a}$$
$$P^i \underline{\gamma}^{ii'} - P^{i'} < 0 \tag{26b}$$
$$\underline{\gamma}^{ii'} > 0 \tag{26c}$$

Then $S^i_j \cap S^{i'}_{j'} \neq \emptyset$ if and only if $[a^i_{j-1} \underline{\gamma}^{ii'}, a^i_j \overline{\gamma}^{ii'}] \cap [a^{i'}_{j'-1}, a^{i'}_{j'}] \neq \emptyset$.

From Lemma 1, it follows that to check emptiness of all slices in two slice-families, we only need to solve two optimization problems and relate the regular values generating the slices, which is very simple. This should be used in the following algorithm.

Algorithm 2. Suppose $\Gamma = (X, f)$ is a linear system, $\{\varphi^i | i \in \boldsymbol{k}\}$ is a family of quadratic Lyapunov functions, where $\varphi^i(x) = x^{\mathrm{T}} P^i x$ is associated with $\mathcal{S}^i$, and $\mathcal{S}^i$ is generated using the regular values $\{a^i_j | j = 1, \dots, |\mathcal{S}^i|\}$. Then the set of locations E can be generated as follows

```
$E \equiv \emptyset$
for $j^1 = 1, \dots, |S^1|$
    $\ddots$
        for $j^k = 1, \dots, |S^k|$
            if $\bigcap_{i=1}^{k} (\varphi^i)^{-1}([a^i_{j^i-1}, a^i_{j^i}]) \neq \emptyset$
                $E \equiv E \cup e_{\mathrm{ex},(j^1,\dots,j^k)}$
            end if
        end for
    $\iddots$
end for
```

5.2 Generation of Invariants

To determine the guard and invariant conditions for the abstraction, a sufficient condition for soundness from Sloth and Wisniewski [2010] is used. However, first we define sound abstraction.

Definition 17. (Sound Abstraction). Let $\Gamma = (X, f)$ be a dynamical system and suppose its state space X is partitioned by $K(\mathcal{S}) = \{e_i | i \in \boldsymbol{m}\}$. Let the initial states $X_0 = \bigcup_{i \in \mathcal{I}} e_i$, with $\mathcal{I} \subseteq \boldsymbol{m}$. Then a timed automaton $\mathcal{A} = (E, E_0, C, \Sigma, I, \Delta)$ with $E_0 = \{e_i | i \in \mathcal{I}\}$ is said to be a sound abstraction of Γ on $[t_1, t_2]$ if $\forall\ t \in [t_1, t_2]$

$$e_i \cap \text{Reach}_{[t,t]}(\Gamma, X_0) \neq \emptyset \text{ implies} \tag{27a}$$
$$\exists e_0 \in E_0 \text{ such that}$$
$$e_i \in \phi_{\mathcal{A}}(t, e_0). \tag{27b}$$

If a sound abstraction $\mathcal{A}$ is safe then Γ is also safe, as the abstraction reaches all locations reached by $\Gamma = (X, f)$. Soundness is close to the notion of simulation; however, by soundness we relate different categories of models.

We partition the state space using quadratic Lyapunov functions, where $\dot{\varphi}^i(x) < 0$; hence, Proposition 7 in Sloth and Wisniewski [2010] can be reformulated in terms of the minimum and maximum decay rates ($\underline{\gamma}^i$, $\overline{\gamma}^i$) of the Lyapunov functions, making the original constraints less conservative. Recall that $a^i_{g_i-1} < a^i_{g_i}$ per definition, and that a^i_j is a regular value if it is different from zero.

Proposition 3. 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 $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$, and let $\dot{\varphi}^i = -x^{\mathrm{T}} Q^i x$. Then a timed automaton $\mathcal{A}(\mathcal{S})$ is a sound abstraction of Γ if its invariants and guards in Procedure 1 satisfy the following inequalities

$$\underline{t}_{S^i_{g_i}} \leq -\log\left(\frac{a^i_{g_i-1}}{a^i_{g_i}}\right)\frac{1}{\overline{\gamma}^i} \tag{28a}$$
$$\overline{t}_{S^i_{g_i}} \geq -\log\left(\frac{a^i_{g_i-1}}{a^i_{g_i}}\right)\frac{1}{\underline{\gamma}^i} \tag{28b}$$

where $\overline{\gamma}^i$ is the solution to the following optimization problem

$$\min \overline{\gamma}^i \text{ subject to} \tag{29a}$$
$$Q^i - \overline{\gamma}^i P^i \leq 0 \tag{29b}$$
$$\overline{\gamma}^i > 0 \tag{29c}$$

and $\underline{\gamma}^i$ is the solution to the following optimization problem

$$\max \underline{\gamma}^i \text{ subject to} \tag{30a}$$
$$\underline{\gamma}^i P^i - Q^i \leq 0 \tag{30b}$$
$$\underline{\gamma}^i > 0. \tag{30c}$$

The proof of Proposition 3 is omitted for brevity, but can be derived from Boyd et al. [1994].

From Proposition 3, it follows that the guard and invariant conditions can be determined by considering only the boundaries of the slices. Therefore, the minimum value of $\dot{\varphi}^i$ can be determined by finding the smallest value a, where $(\dot{\varphi}^i)^{-1}(a)$ intersects a boundary of a slice. This is illustrated in the left subplot of Fig. 2. The following proposition provides a method for obtaining $\overline{t}_{S^i_{g_i}}$ used in the invariant conditions.

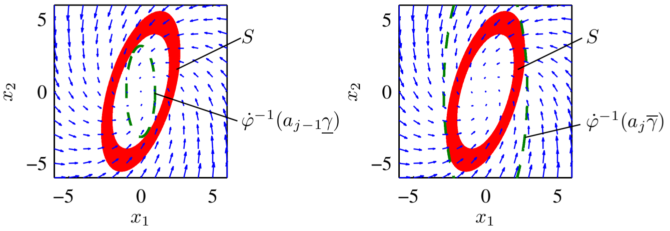

Fig. 2. Illustration of a slice $S = \varphi^{-1}([a_{j-1}, a_j])$ (red) and the level sets $\dot{\varphi}^{-1}(a_{j-1}\underline{\gamma})$ and $\dot{\varphi}^{-1}(a_j\overline{\gamma})$ (green and dashed) intersecting the slice S.

Algorithm 3. 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 $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$. Then for any location $e_{(g,h)} \in E$, the invariant is

$$I(e_{(g,h)}) = \bigwedge_{i=1}^{k} c^i \leq \overline{t}_{S^i_{g_i}} \tag{31}$$

where $\overline{t}_{S^i_{g_i}}$ satisfies (28b).

To find the guard conditions, the maximum value a where $(\dot{\varphi}^i)^{-1}(a)$ intersects a boundary of a slice should be found. This is illustrated in the right subplot of Fig. 2.

5.3 Generation of Transition Relations

The following algorithm provides a method for generating the transition relations, and for obtaining $\underline{t}_{S^i_{g_i}}$ used in the guard conditions.

Algorithm 4. 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 $\{a^i_j | j = 1, \ldots, |\mathcal{S}^i|\}$. Then for every pair of locations, $e_{(g,h)}, e_{(g',h')} \in E$, where $g'_i - 1 = g_i$ there is a transition relation

$$\delta_{(g,h)\to(g',h')} =$$
$$(e_{(g,h)}, G_{(g,h)\to(g',h')}, \sigma, R_{(g,h)\to(g',h')}, e_{(g',h')}), \tag{32a}$$

where

$$G_{(g,h)\to(g',h')} = \bigwedge_{i=1}^{k} \begin{cases} c^i \geq \underline{t}_{S^i_{g_i}} & \text{if } g_i - g'_i = 1 \\ c^i \geq 0 & \text{otherwise} \end{cases} \tag{32b}$$

where $\underline{t}_{S^i_{g_i}}$ satisfies (28a) and $R_{(g,h)\to(g',h')}$ is given by

$$c_i \in R_{(g,h)\to(g',h')} \tag{32c}$$

iff $g_i - g'_i = 1$.

We conclude that the main complexity of this method origin from generation of locations, which requires solving 2^k LMIs, where k is the number of slice families. The calculation of the time information only requires solving $2k$ LMIs.

6. ILLUSTRATIVE EXAMPLE

In this section, a system is abstracted by a timed automaton using the proposed algorithm. It is chosen to abstract a two-dimensional system, to enable visualization of the partitioning; however, the method applies for systems of arbitrary dimension.

Consider the following two-dimensional linear system

$$\dot{x} = \begin{bmatrix} -1 & -0.5 \\ 0.75 & -1.6 \end{bmatrix} x. \tag{33}$$

We have randomly chosen two quadratic Lyapunov functions $\varphi^i = x^{\mathrm{T}} P^i x$, $i = 1, 2$, with

$$P^1 = \begin{bmatrix} 0.4438 & -0.0750 \\ -0.0750 & 0.0359 \end{bmatrix}, P^2 = \begin{bmatrix} 0.0901 & 0.0668 \\ 0.0668 & 0.2916 \end{bmatrix}, \tag{34}$$

and regular values $\{a_1^1, a_2^1, a_3^1, a_4^1\} = \{0.1, 0.3, 0.6, 1\}$, $\{a_1^2, a_2^2, a_3^2, a_4^2\} = \{0.18, 0.35, 0.6, 0.8\}$. The partition of the state space is illustrated in Fig. 3.

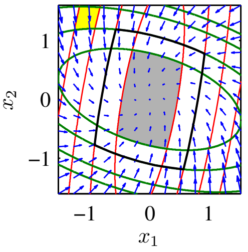

Fig. 3. Illustration of a partition of a state space using two Lyapunov functions $\varphi^1(x)$ (red) and $\varphi^2(x)$ (green).

A small part of the timed automaton abstracting the system, corresponding to the cells within the black line in Fig. 3, is illustrated in Fig. 4.

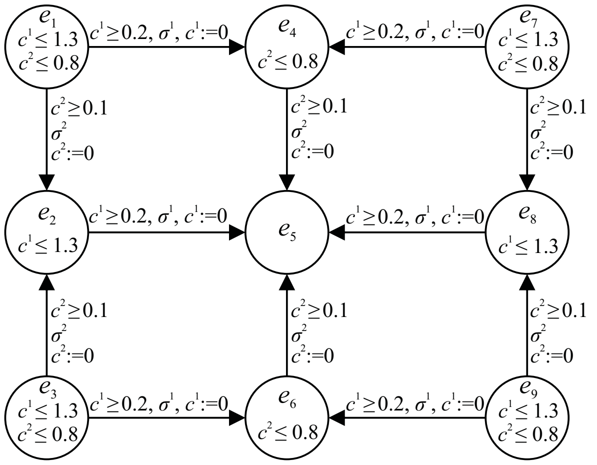

Fig. 4. Illustration of the timed automata abstracting the dynamical system.

It follows from Fig. 4 that the timed automaton has minimum and maximum times for staying in a location. Therefore, it is clear that an over-approximation of trajectories of the system can be calculated using the abstraction.

To show the type of temporal requirements that can be verified using this abstraction, the following question is asked: *Do all trajectories initialized in the yellow cell in Fig. 3 reach the gray cell within 5 s? (TCTL specification: $A\Diamond_{\leq 5} e_5$)* The answer is *Yes*. The analysis took less than 0.5 s in the prototype implementation in MATLAB.

7. CONCLUSION

In this paper, an algorithm is proposed for abstracting linear systems by timed automata. The abstraction is based on partitioning the state space of the dynamical systems by set-differences of positive invariant sets, generated by level sets of quadratic forms.

The proposed algorithm makes it possible to automatically abstract linear system models by timed automata, which can be verified in a model checker. As the abstraction is sound, it is possible to verify temporal properties of dynamical systems automatically. This is illustrated in an example, where a dynamical system is abstracted by a timed automaton using the proposed algorithm. Future research directions are in relation to finding optimal partitioning functions, and a refinement procedure for the choice of regular values.

REFERENCES

R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.

R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 414–425, June 1990.

E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler. Recent progress in continuous and hybrid reachability analysis. In *Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design*, pages 1582–1587, 2006.

G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *International School on Formal Methods for the Design of Real-Time Systems*, pages 200–237, 2004.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *SIAM studies in applied mathematics*. SIAM, 1994. ISBN 0-89871-485.

U. Fahrenberg, K. G. Larsen, and C. R. Thrane. Verification, performance analysis and controller synthesis for real-time systems. In *FSEN*, pages 34–61, 2009.

G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *HSCC*, pages 258–273, 2005.

A. Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC 2005*, 2005.

A. B. Kurzhanski and I. Vlyi. *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser Boston, 1997.

O. Maler and G. Batt. Approximating continuous systems by timed automata. In *Proceedings of the 1st international workshop on Formal Methods in Systems Biology*, pages 77–89, 2008.

K. R. Meyer. Energy functions for Morse Smale systems. *American Journal of Mathematics*, 90(4):1031–1040, 1968.

Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117 – 126, 2006.

C. Sloth and R. Wisniewski. Abstraction of continuous dynamical systems utilizing Lyapunov functions. In *Proceedings of the 49th IEEE Conference on Decision and Control*, Atlanta, Georgia, USA, December 2010.

A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, pages 57–83, 2008.

L. W. Tu. *An Introduction to Manifolds*. Springer, 2008.