Timed-automata abstraction of switched dynamical systems using control funnels*

Patricia Bouyer[1], Nicolas Markey[1],
Nicolas Perrin[2,3], Philipp Schlehuber-Caissier[2]

[1]Laboratoire Spécification & Vérification – CNRS & ENS Cachan – France
[2]Sorbonne Universités, UPMC Univ Paris 06, UMR 7222, ISIR, F-75005, Paris, France
[3]CNRS, UMR 7222, ISIR, F-75005, Paris, France

Abstract. The development of formal methods for control design is an important challenge with potential applications in a wide range of safety-critical cyber-physical systems. Focusing on switched dynamical systems, we propose a new abstraction, based on time-varying regions of invariance (the *control funnels*), that models behaviors of systems as timed automata. The main advantage of this method is that it allows automated verification of formal specifications and reactive controller synthesis without discretizing the evolution of the state of the system. Efficient constructions are possible in the case of linear dynamics. We demonstrate the potential of our approach with two examples.

1 Introduction

Verification and synthesis are notoriously difficult for hybrid dynamical systems, i.e. systems that allow abrupt changes in continuous dynamics. For instance, reachability is already undecidable for 2-dimensional piecewise-affine maps [14], or for 3-dimensional dynamical systems with piecewise-constant derivatives [2].

To enable automated logical reasoning on switched dynamical systems, most methods tend to entirely discretize the dynamics, for example by approximating the behavior of the system with a finite-state machine. Alternatively, early work pointed out links between hybrid and timed systems [20], and several methods have been designed to create formal abstractions of dynamical systems that do not rely on a discretization of time. In [11], a finite maneuver automaton is constructed from a library of motion primitives, and motion plans correspond to timed words. In [16, 12], switched controller synthesis and stochastic optimal control are realized via metric temporal logic (MTL) or metric-interval temporal logic (MITL) specifications. In [22, 19], grid-based abstractions and timed automata are used for motion planning or to check timed properties of dynamical systems. In [24], a subdivision of the state space created from sublevel sets of Lyapunov functions leads to an abstraction of dynamical systems by timed automata that enables verification and falsification of safety properties. The same kind of abstraction is

* This work has been partly supported by ERC Starting grant EQualIS (FP7-308087) and by European FET project Cassting (FP7-601148)

used in [23] for controller design via timed games, but the update map of the timed games obtained is such that synthesis cannot be realized using existing tools. In [8], the state space of each mode of a piecewise-affine hybrid system is portioned into polytopes, and thanks to control laws that prevent the system from exiting through a given facet, or that force the system to exit through a facet in finite time, reactive control problems can be solved as timed games on timed automata.

Our contribution is a novel timed-automata abstraction of switched dynamical systems based on a particular kind of time-varying regions of invariance: control funnels. Recent results have shown that these invariants are very useful for robust motion planning and control [26, 18, 17], and that funnels or similar concepts related to the notion of Lyapunov stability can be used for formal verification of hybrid systems [13, 10], and for reactive controller synthesis [9].

The paper is organized as follows: Section 2, describes how control funnels, especially for trajectory tracking controllers, can be used to create timed transition systems that abstract the behavior of a given switched dynamical system, and as a result can potentially allow the use of verification tools for motion planning. In Section 3, we show how these timed transition systems can be encoded as timed automata. In Section 4, we consider the case of linear dynamics and introduce the notion of fixed size LQR funnel. In Section 5, we present two examples of application and efficient algorithms that manipulate these LQR funnels. In the first one, a timed game is solved by the tool Uppaal-Tiga [5] for the synthesis of a controller that can reactively adjust the phase of a sine wave controlled in acceleration. In the second example, we show that, using our timed-automata abstraction with LQR funnels along constant velocity trajectories, a non-trivial solution to a pick-and-place problem can be computed by the model checker Uppaal [6]. Section 6 concludes and presents avenues for future work.

2 Graphs of control funnels

2.1 Control funnels

Consider a controlled dynamical system governed by the following equation:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t, u(\boldsymbol{x}, t)), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the state of the system (which can contain velocities[1]), $t \in \mathbb{R}^+$ is a real (clock) value corresponding to time (we restrict ourselves to nonnegative time values), $u\colon \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^k$ is the control input function, and f is a continuously differentiable function from $\mathbb{R}^d \times \mathbb{R}^+ \times \mathbb{R}^k$ to $\mathbb{R}^d$ (which ensures the uniqueness of the solution for given initial conditions). Assuming that the function u is fixed, we also use the following notation for Equation (1):

$$\dot{\boldsymbol{x}} = f_u(\boldsymbol{x}, t). \tag{2}$$

[1] In this paper, we mostly consider state spaces that describe the position and velocity of systems controlled in acceleration. The continuity of trajectories in the state space ensures that the position is always a continuously differentiable function of time.

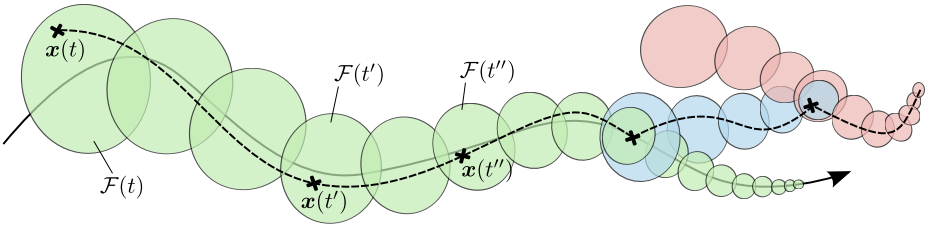

Fig. 1. An example of control funnel for a controller tracking a reference trajectory. The dashed line is a trajectory of the controlled system in the state space. On the right side, switching transitions between control funnels are depicted.

A *control funnel* for the above dynamical system is a function $\mathcal{F}\colon I \to 2^{\mathbb{R}^d}$ such that $I \subseteq \mathbb{R}^+$ and for any solution $\boldsymbol{x}(t)$ of (2), the following property holds:

$$\forall t_1 \in I.\ \forall t_2 \in I.\ (t_2 > t_1 \text{ and } \boldsymbol{x}(t_1) \in \mathcal{F}(t_1)) \Rightarrow \boldsymbol{x}(t_2) \in \mathcal{F}(t_2). \tag{3}$$

It corresponds to time-varying regions of invariance.

Example 1. A typical example of a control funnel based on a trajectory tracking controller (that is, a control funnel asymptotically converging towards a reference trajectory in the state space) is shown in Fig. 1.

Example 2. For a concrete example, consider the simple system whose trajectories are of the form $e^{-t} \cdot \boldsymbol{x}_0$. Then any set $W \subseteq \mathbb{R}^d$ defines a control funnel $\mathcal{F}_W\colon t \mapsto \{e^{-t} \cdot \boldsymbol{w} \mid \boldsymbol{w} \in W\}$.

The notion of funnel was popularized by Mason [21], and it usually specifically refers to operations that eliminate uncertainty (as is the case in the example of Fig. 1) by collapsing a large set of initial conditions into a smaller set of final conditions (see for instance [26]). In our case, the control funnel may or may not reduce uncertainty, and it is important to note that the set $\mathcal{F}(t)$ does not have to decrease in size over time. This more general concept is closer to the definition of *viability tubes* [4], but we nevertheless use the term control funnel as some reduction of uncertainty is often essential to the usefulness of our abstractions. We address the computation of control funnels in Section 4, and leave them as relatively abstract objects for now.

2.2 Motion planning

Let us suppose that we have a finite set U of control laws $u_1(\boldsymbol{x}, t)$, $u_2(\boldsymbol{x}, t)$, …, $u_n(\boldsymbol{x}, t)$ that respectively set the dynamics of a given system to $\dot{\boldsymbol{x}} = f_{u_1}(\boldsymbol{x}, t)$, $\dot{\boldsymbol{x}} = f_{u_2}(\boldsymbol{x}, t)$, …, $\dot{\boldsymbol{x}} = f_{u_n}(\boldsymbol{x}, t)$.

We say that the system can switch to the control law $u_i(\boldsymbol{x}, t)$ at some state $\widetilde{\boldsymbol{x}}$ whenever there is $t_0 \in \mathbb{R}^+$ and a solution $\boldsymbol{x}(t)$ of $\dot{\boldsymbol{x}} = f_{u_i}(\boldsymbol{x}, t)$ with initial condition $\widetilde{\boldsymbol{x}} = \boldsymbol{x}(t_0)$. Typically, if $u_i(\boldsymbol{x}, t)$ corresponds to a trajectory tracking controller, t_0 identifies the point of the trajectory where the tracking is triggered.

Informally, the *motion planning* problem asks, given a finite set of control laws as above, an initial point $\boldsymbol{x}_0$, a target zone $T_f \subseteq \mathbb{R}^d$, and an obstacle $\Omega \subseteq \mathbb{R}^d$, whether there exists a sequence of control law switches that generates a trajectory from $\boldsymbol{x}_0$ to T_f while avoiding the obstacle Ω. Several variants of this problem can be considered, that vary on the objective (for instance some tasks can be expressed as ω-regular objectives), but we focus here on a reachability with avoidance objective.

More formally, motion planning asks for a finite sequence of time values $t_0^1 < t_1^1, t_0^2 < t_1^2, \ldots, t_0^P < t_1^P$, a finite sequence of control laws indices $k_1, \ldots, k_P$, and a finite sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_P \in T_f$ of points in $\mathbb{R}^d$, such that:

(a) for every $1 \leq p \leq P$, if $\boldsymbol{x}^p$ is the unique solution to $\dot{\boldsymbol{x}} = f_{u_{k_p}}(\boldsymbol{x}, t)$ with initial condition $\boldsymbol{x}^p(t_0^p) = \boldsymbol{x}_{p-1}$, then $\boldsymbol{x}^p(t_1^p) = \boldsymbol{x}_p$.
(b) for every $1 \leq p \leq P$, for every $t_0^p \leq t \leq t_1^p$, $\boldsymbol{x}^p(t) \notin \Omega$.

Intuitively, this means that we can switch conveniently between all the control laws, causing discrete changes in the system dynamics, and ensure the global (reachability with avoidance) objective. The continuous trajectory generated by the solution above is the concatenation of the trajectory portions $\{x^p(t) \mid t_0^p \leq t \leq t_1^p\}$ for $1 \leq p \leq P$.

2.3 Motion planning with graphs of control funnels

We now explain how the motion planning problem can be abstracted using timed transition systems based on control funnels.

For each control law $u_i(\boldsymbol{x}, t)$, we assume that we have a finite set of control funnels $\mathcal{F}_i^1, \mathcal{F}_i^2, \ldots, \mathcal{F}_i^{m_i}$, respectively defined over $I_i^1 \subseteq \mathbb{R}^+$, $I_i^2 \subseteq \mathbb{R}^+$, $\ldots$, $I_i^{m_i} \subseteq \mathbb{R}^+$. We assume that for every $1 \leq i \leq n$, for every $1 \leq j \leq m_i$, for every $t \in I_i^j$, it holds $\mathcal{F}_i^j(t) \cap \Omega = \varnothing$, which means that trajectories contained in these funnels always avoid the obstacle Ω.

Consider a control law switch at position $\widetilde{\boldsymbol{x}}$ to law $u_i(\boldsymbol{x}, t)$ with clock value t_0. If there exists a control funnel $\mathcal{F}_i^j$ such that $t_0 \in I_i^j$, and $\widetilde{\boldsymbol{x}} \in \mathcal{F}_i^j(t_0)$, then we know that the state of the system will remain inside $\mathcal{F}_i^j(t)$ for any $t > t_0$ in I_i^j (as long as control law $u_i(\boldsymbol{x}, t)$ is used). To always keep the system inside one of the control funnels, we can impose sufficient conditions on the switches. For instance, if the state is inside $\mathcal{F}_i^j(t_0)$, and if for some future clock value t_1, there exists a control funnel $\mathcal{F}_k^l$ and $t_2 \in I_k^l$ such that $\mathcal{F}_i^j(t_1) \subseteq \mathcal{F}_k^l(t_2)$, then when the clock value is t_1 we can safely switch to the control law $u_k(\boldsymbol{x}, t)$ while setting the clock to t_2. Indeed, we know that the state of the system at the switch instant will be inside $\mathcal{F}_k^l(t_2)$, and therefore it will remain inside $\mathcal{F}_k^l(t)$ after the switch. Such transitions from a funnel to another are illustrated on the right side of Fig. 1. It is worth noting that similar transitions could be achieved with, instead of control funnels, controller specifications as introduced in [15]. For some control funnels $\mathcal{F}_i^j$ and $\mathcal{F}_i^k$ associated to the same control law, it is the case (see Section 4) that when funnel $\mathcal{F}_i^j$ is entered at time t, then at any time

$t' \geq t + h_i^{j \to k}$ (where $h_i^{j \to k}$ is a constant), the state of the system is inside $\mathcal{F}_i^k(t')$. In that case, we say that the funnel $\mathcal{F}_i^k$ $h_i^{j \to k}$-absorbs the funnel $\mathcal{F}_i^j$.

These rules for navigating between control funnels give to the set of control funnels the structure of an infinite graph, or, more precisely, of a timed transition system with real-valued clocks. One of the clocks of this timed transition system is c_t, the *controller clock*. We add two other clocks: a global clock c_g, and a local clock c_h.

The *funnel timed transition system* $\mathcal{T}_{U,F}$ associated with the family of laws $U = (u_i(\boldsymbol{x}, t))_{1 \leq i \leq n}$ and the family of funnels $F = ((\mathcal{F}_i^j, I_i^j))_{1 \leq i \leq n, 1 \leq j \leq m_i}$ is defined as follows. The configurations are pairs $(\mathcal{F}_i^j, v)$ where v assigns a non-negative real value to each of the clocks c_t, c_g and c_h, with $v(c_t) \in I_i^j$, and its transition set contains three types of elements:

- the *time-elapsing transitions*: $(\mathcal{F}_i^j, v) \to (\mathcal{F}_i^j, v + \Delta)$ whenever $[v(c_t), v(c_t) + \Delta] \subseteq I_i^j$ (where $v + \Delta$ denotes the valuation that maps each clock c to $v(c) + \Delta$);
- the *switching transitions*: $(\mathcal{F}_i^j, v) \to (\mathcal{F}_k^l, v')$ whenever $v'(c_g) = v(c_g)$, $v'(c_h) = 0$, $v(c_t) \in I_i^j$, $v'(c_t) \in I_k^l$, and $\mathcal{F}_i^j(v(c_t)) \subseteq \mathcal{F}_k^l(v'(c_t))$;
- the *absorbing transitions:* $(\mathcal{F}_i^j, v) \to (\mathcal{F}_i^k, v')$ whenever $\mathcal{F}_i^k$ $h_i^{j \to k}$-absorbs $\mathcal{F}_i^j$, $v(c_h) \geq h_i^{j \to k}$, $v'(c_h) = 0$, $v'(c_g) = v(c_g)$ and $v'(c_t) = v(c_t)$.

A run in this timed transition system is a finite sequence of configurations $\big((\mathcal{F}_{i_0}^{j_0}, v_0), (\mathcal{F}_{i_1}^{j_1}, v_1), \ldots, (\mathcal{F}_{i_P}^{j_P}, v_P)\big)$ such that $v_0(c_h) = v_0(c_g) = 0$, $v_0(c_t) \in I_{i_0}^{j_0}$, and all the transitions $(\mathcal{F}_{i_p}^{j_p}, v_p) \to (\mathcal{F}_{i_{p+1}}^{j_{p+1}}, v_{p+1})$ for $0 \leq p < P$ are valid transitions that belong to $\mathcal{T}_{U,F}$.

Such a run is of total duration $v_P(c_g)$, and it corresponds to a schedule of control law switches that results from the following rules: initially, the control law is set to $u_{i_0}(\boldsymbol{x}, t)$, and the controller clock c_t is set to $v_0(c_t)$. For every time-elapsing transition $(\mathcal{F}_i^j, v) \to (\mathcal{F}_i^j, v + \Delta)$, the same control law $u_i(\boldsymbol{x}, t)$ is kept for a duration of Δ time units, and for every switching transition $(\mathcal{F}_i^j, v) \to (\mathcal{F}_k^l, v')$, the control law is switched from $u_i(\boldsymbol{x}, t)$ to $u_k(\boldsymbol{x}, t)$, with an initialization of the controller clock to $v'(c_t)$. Absorbing transitions are discarded, as they just correspond to an instantaneous change of funnels for the same control law. Let us denote this sequence of switches by r. Then, it is fundamental to notice that for every $\boldsymbol{x} \in \mathcal{F}_{i_0}^{j_0}(v_0(c_t))$, if we follow the schedule of control law switches just described, then the system remains inside control funnels and reaches at the end of the run a unique point of $\mathbb{R}^d$, that we denote $r(\boldsymbol{x})$. The trajectory going from $\boldsymbol{x}$ to $r(\boldsymbol{x})$ is also uniquely defined.

The funnel timed transition system $\mathcal{T}_{U,F}$ satisfies the following property:

Theorem 1. *Let* $r = \big((\mathcal{F}_{i_0}^{j_0}, v_0), (\mathcal{F}_{i_1}^{j_1}, v_1), \ldots, (\mathcal{F}_{i_P}^{j_P}, v_P)\big)$ *be a run in* $\mathcal{T}_{U,F}$. *If* $\boldsymbol{x} \in \mathcal{F}_{i_0}^{j_0}(v_0(c_t))$, *then* $r(\boldsymbol{x}) \in \mathcal{F}_{i_P}^{j_P}(v_P(c_t))$.

In some sense, the funnel timed transition system $\mathcal{T}_{U,F}$ is a correct abstraction of trajectories that can be generated by the set of control laws: if $\boldsymbol{x}_0 \in \mathcal{F}_{i_0}^{j_0}(v_0(c_t))$

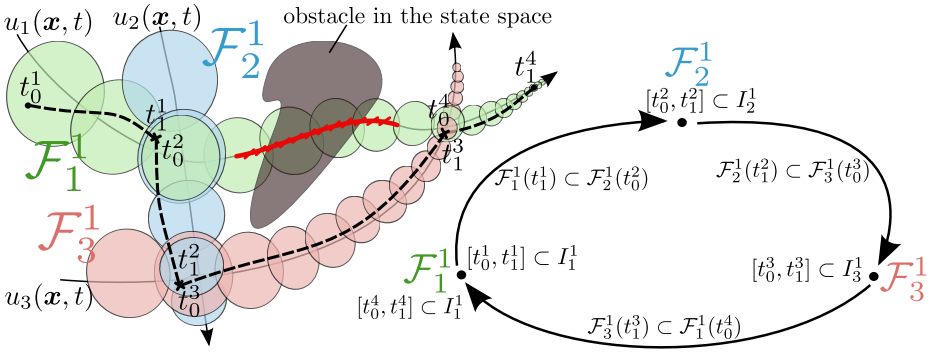

Fig. 2. Run of a funnel timed transition system with three control funnels: $r = \big((\mathcal{F}_1^1, v_0^1), (\mathcal{F}_1^1, v_1^1), (\mathcal{F}_2^1, v_0^2), (\mathcal{F}_2^1, v_1^2), (\mathcal{F}_3^1, v_0^3), (\mathcal{F}_3^1, v_1^3), (\mathcal{F}_4^1, v_0^4), (\mathcal{F}_4^1, v_1^4)\big)$, with: $\forall 1 \leq i \leq 4$, $v_0^i(c_t) = t_0^i$, $v_1^i(c_t) = t_1^i$, $v_0^i(c_h) = 0$, $v_1^i(c_h) = t_1^i - t_0^i$, $v_1^i(c_g) = v_0^i(c_g) + v_1^i(c_h)$, and $v_0^1(c_g) = 0$, and $\forall 2 \leq i \leq 4$, $v_0^i(c_g) = v_1^{i-1}(c_g)$.

and $\mathcal{F}_{i_P}^{j_P}(v_P(c_t)) \subseteq T_f$, then such a run witnesses a solution to the motion planning problem. However, this abstraction is obviously not complete.

Example 3 (An example with obstacles). The example in Fig. 2 shows a run with three control laws $u_1(\boldsymbol{x},t)$, $u_2(\boldsymbol{x},t)$ and $u_3(\boldsymbol{x},t)$, three control funnels $\mathcal{F}_1^1$, $\mathcal{F}_2^1$ and $\mathcal{F}_3^1$, and an obstacle in the state space. The domains of definition of the control funnels I_1^1, I_2^1 and I_3^1 are such that for all $\alpha \in \{1,2,3\}$ and all $t \in I_\alpha^1$, $\mathcal{F}_\alpha^1(t)$ does not intersect the obstacles.

With the previous property, any run in the corresponding funnel timed transition system leads to a trajectory that avoids the obstacles. The example of Fig. 2, where reaching $\mathcal{F}_1^1(t_1^4)$ from $\mathcal{F}_1^1(t_0^1)$ requires a series of switches between the different control funnels, shows the potential interest of automated verification in timed transition systems, as it can result in the generation of obstacle-free dynamic trajectories via appropriate control law switches.

3 Reduction to timed automata

Timed automata [1] are a timed extension of finite-state automata, with a well-understood theory. They provide an expressive formalism for modelling and reasoning about real-time systems, and enjoy decidable reachability properties; much efforts have been invested over the last 20 years for the development of efficient algorithms and tools for their automatic verification (such as the tool Uppaal [6], which we use in this work).

Let C be a finite set of real-valued variables called *clocks*. A *clock valuation* over a finite set of clocks C is a mapping $v\colon C \to \mathbb{R}^+$. We write $\mathbb{R}^C$ for the set of clock valuations over C. If $\Delta \in \mathbb{R}^+$, we write $v + \Delta$ for the clock valuation defined by $(v+\Delta)(c) = v(c) + \Delta$ for every $c \in C$. A *clock constraint* over C is a boolean combination of expressions of the form $c \sim \alpha$ where $\alpha \in \mathbb{Q}$, and

$\sim \in \{\leq, <, =, >, \geq\}$. We denote by $\mathcal{C}(C)$ the set of clock constraints over C. We write $v \models g$ if v satisfies g (defined in a natural way). A reset of the clocks is an element res of $(\mathbb{Q} \cup \{\bot\})^C$ (which we may write $R(C)$), and if v is a valuation, its image by res, denoted $\mathsf{res}(v)$, is the valuation mapping c to $v(c)$ whenever $\mathsf{res}(c) = \bot$, and to $\mathsf{res}(c) \in \mathbb{Q}$ otherwise.

We define a slight extension of timed automata with rational constants, general boolean combinations of clock constraints and extended clock resets; those timed automata are as expressive as standard timed automata [7], but they will be useful for expressing funnel timed transition systems. A *timed automaton* is a tuple $\mathcal{A} = (L, L_0, L_F, C, E, \mathsf{Inv})$ where L is a finite set of locations, $L_0 \subseteq L$ is a set of initial locations, $L_F \subseteq L$ is a set of final locations, C is a finite set of clocks, $E \subseteq L \times \mathcal{C}(C) \times R(C) \times L$ is a finite set of edges, and $\mathsf{Inv} \colon L \to \mathcal{C}(C)$ is an invariant labelling function.

A configuration of $\mathcal{A}$ is a pair $(\ell, v) \in L \times \mathbb{R}^C$ such that $v \models \mathsf{Inv}(\ell)$, and the timed transition system generated by $\mathcal{A}$ is given by the following two rules:

- *time-elapsing transition:* $(\ell, v) \to (\ell, v + \Delta)$ whenever $v + \delta \in \mathsf{Inv}(\ell)$ for every $0 \leq \delta \leq \Delta$;
- *switching or absorbing transition:* $(\ell, v) \to (\ell', v')$ whenever there exists $(\ell, g, \mathsf{res}, \ell') \in E$ such that $v \models g \wedge \mathsf{Inv}(\ell)$, $v' = \mathsf{res}(v)$, and $v' \in \mathsf{Inv}(\ell')$.

A run in $\mathcal{A}$ is a sequence of consecutive transitions. The most fundamental result about timed automata is the following:

Theorem 2 ([1]). *Reachability in timed automata is PSPACE-complete.*

We consider again the family of control laws $U = (u_i(\boldsymbol{x}, t))_{1 \leq i \leq n}$, and the family of funnels $F = ((\mathcal{F}_i^j, I_i^j))_{1 \leq i \leq n, 1 \leq j \leq m_i}$, as in the previous section. For every pair $1 \leq i, k \leq n$, and every $1 \leq j \leq m_i$ and $1 \leq l \leq m_k$, we select finitely many tuples $(\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (k, l))$ with $\alpha, \beta, \gamma \in \mathbb{Q}$ such that (i) $[\alpha, \beta] \subseteq I_i^j$, (ii) $\gamma \in I_k^l$, and (iii) for every $t \in [\alpha, \beta]$, $\mathcal{F}_i^j(t) \subseteq \mathcal{F}_k^l(\gamma)$. This allows us to under-approximate the possible switches between funnels. For every $1 \leq i \leq n$, for every pair $1 \leq j, k \leq m_i$, we select at most one tuple $(\mathsf{absorb}, \nu, (i, j, k))$ such that $\nu \in \mathbb{Q}$ and $\mathcal{F}_i^k(t)$ ν-absorbs $\mathcal{F}_i^j(t)$. This allows us to under-approximate the possible absorbing transitions. For every $1 \leq i \leq n$ and every $1 \leq j \leq m_i$, we fix a finite set of tuples $(\mathsf{initial}, \alpha, (i, j))$ such that $\alpha \in \mathbb{Q}$ and $\boldsymbol{x}_0 \in \mathcal{F}_i^j(\alpha)$. This allows us to under-approximate the possible initialization to a control funnel containing the initial point $\boldsymbol{x}_0$. For every $1 \leq i \leq n$ and $1 \leq j \leq m_i$, we fix finitely many tuples $(\mathsf{invariant}, S_{i,j}, (i, j))$, where $S_{i,j} \subseteq I_i^j$ is a finite set of closed intervals with rational bounds. This allows us to under-approximate the definition set of the funnels. Finally, for every $1 \leq i \leq n$ and $1 \leq j \leq m_i$, we fix finitely many tuples $(\mathsf{target}, [\alpha, \beta], (i, j))$, where $\alpha, \beta \in \mathbb{Q}$ and $[\alpha, \beta] \subseteq I_i^j \cap \{t \mid \mathcal{F}_i^j(t) \subseteq T_f\}$. This allows us to under-approximate the target zone. We denote by K the set of all tuples we just defined.

We can now define a timed automaton that conservatively computes the runs generated by the funnel timed transition system. It is defined by $\mathcal{A}_{U,F,K} = (L, L_0, L_F, C, E, \mathsf{Inv})$ with:

– $L = \{\mathcal{F}_i^j \mid 1 \leq i \leq n,\ 1 \leq j \leq m_i\} \cup \{\mathsf{init}, \mathsf{stop}\}$; $L_0 = \{\mathsf{init}\}$; $L_F = \{\mathsf{stop}\}$;
– $C = \{c_t, c_g, c_h\}$;
– E is composed of the following edges:
  • for every $(\mathsf{initial}, \alpha, (i,j)) \in K$, we have an edge $(\mathsf{init}, \mathtt{true}, \mathsf{res}, \mathcal{F}_i^j)$ in E, with $\mathsf{res}(c_t) = \alpha$ and $\mathsf{res}(c_g) = \mathsf{res}(c_h) = 0$;
  • for every $(\mathsf{switch}, [\alpha, \beta], (i,j), \gamma, (k,l)) \in K$, we have an edge $(\mathcal{F}_i^j, \alpha \leq c_t \leq \beta, \mathsf{res}, \mathcal{F}_k^l)$ with $\mathsf{res}(c_t) = \gamma$, $\mathsf{res}(c_h) = 0$ and $\mathsf{res}(c_g) = \bot$;
  • for every $(\mathsf{target}, [\alpha, \beta], (i,j)) \in K$, we have an edge $(\mathcal{F}_i^j, \alpha \leq c_t \leq \beta, \mathsf{res}, \mathsf{stop})$ in E, with $\mathsf{res}(c_t) = \mathsf{res}(c_g) = \mathsf{res}(c_h) = \bot$;
  • for every $(\mathsf{absorb}, \nu, (i,j,k)) \in K$, we have an edge $(\mathcal{F}_i^j, c_h \geq \nu, \mathsf{res}, \mathcal{F}_i^k)$ with $\mathsf{res}(c_h) = 0$ and $\mathsf{res}(c_t) = \mathsf{res}(c_g) = \bot$;
– for every $(\mathsf{invariant}, S_{i,j}, (i,j)) \in K$, we let $\mathsf{Inv}(\mathcal{F}_i^j) \triangleq \bigvee_{[\alpha,\beta] \in S_{i,j}} (\alpha \leq c_t \leq \beta)$.

We easily get the following result:

Theorem 3. *Let* $(\mathsf{init}, v_0) \to (\ell_1, v_1) \to \cdots \to (\ell_P, v_P) \to (\mathsf{stop}, v_P)$ *be a run in* $\mathcal{A}_{U,F,K}$ *such that* v_0 *assigns* 0 *to every clock. Then* $r = ((\ell_1, v_1),\ \ldots,\ (\ell_P, v_P))$ *is a run of the funnel timed transition system* $\mathcal{T}_{U,F}$ *that brings* $\boldsymbol{x}_0$ *to* $r(\boldsymbol{x}_0) \in T_f$ *while avoiding the obstacle* Ω.

This shows that the reachability of stop in $\mathcal{A}_{U,F,K}$ implies that there exists an appropriate schedule of control law switches that safely brings the system to the target zone. Of course, the method is not complete, not all schedules can be obtained using the timed automaton $\mathcal{A}_{U,F,K}$. But if $\mathcal{A}_{U,F,K}$ is precise enough, it will be possible to use automatic verification techniques for dynamic trajectory generation.

Remark 1. We could be more precise in the modelling as a timed automaton, if we could use non-deterministic clock resets [7]; but we should then be careful with decidability issues. Additionally, non-deterministic resets are not implemented in Uppaal, which is why we have chosen timed automata with deterministic resets only.

Remark 2. As we show with some examples in Section 5, our timed-automata abstraction can be used for other types of objectives than just reachability with avoidance. In particular, the approach can be extended to *timed games* [3], where special uncontrollable transitions model uncertainty in the environment. In that case, the aim is not to synthesize one single run in the system, but rather a *strategy* that dictates how the system should be controlled, depending on how the environment evolves. It is worth knowing that winning strategies can be computed in exponential time in timed games, and that the tool Uppaal-Tiga [5] computes winning strategies. In Section 5.1, we give an example of application where timed games and Uppaal-Tiga are used.

4 LQR funnels

In this section we consider the particular case of linear time-invariant stabilizable systems whose dynamics are described by the following equation:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}, \tag{4}$$

where $A \in \mathbb{R}^{d\times d}$ and $B \in \mathbb{R}^{d\times k}$ are two constant matrices, and $\boldsymbol{u} \in \mathbb{R}^k$ is the control input. We also consider reference trajectories that can be realized with controlled systems described by Eq. (4), i.e. trajectories $\boldsymbol{x}_{\text{ref}}(t)$ for which there exists $\boldsymbol{u}_{\text{ref}}(\boldsymbol{x}, t)$ such that $\dot{\boldsymbol{x}}_{\text{ref}} = A\boldsymbol{x}_{\text{ref}} + B\boldsymbol{u}_{\text{ref}}$. We can combine this equation with (4) and get $\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\text{ref}} = A(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}) + B(\boldsymbol{u} - \boldsymbol{u}_{\text{ref}})$, which rewrites

$$\dot{\boldsymbol{x}}_{\Delta} = A\boldsymbol{x}_{\Delta} + B\boldsymbol{u}_{\Delta}. \tag{5}$$

To track $\dot{\boldsymbol{x}}_{\text{ref}}$, we compute $\boldsymbol{u}_{\Delta}$ as an infinite-time linear quadratic regulator (LQR, see [25]), i.e. a minimization of the cost: $J = \int_0^{\infty} \left(\boldsymbol{x}_{\Delta}^{\mathsf{T}} Q \boldsymbol{x}_{\Delta} + \boldsymbol{u}_{\Delta}^{\mathsf{T}} R \boldsymbol{u}_{\Delta}\right) \mathrm{d}t$, where Q and R respectively are positive-semidefinite and positive-definite matrices. The solution is $\boldsymbol{u}_{\Delta} = -K\boldsymbol{x}_{\Delta}$, with $K = R^{-1}B^{\mathsf{T}}P$ and P being the unique positive-definite matrix solution of the continuous time algebraic Riccati equation: $PA + A^{\mathsf{T}}P - PBR^{-1}B^{\mathsf{T}}P + Q = 0$.

The dynamics can be rewritten $\dot{\boldsymbol{x}}_{\Delta} = (A - BK)\boldsymbol{x}_{\Delta} = \bar{A}\boldsymbol{x}_{\Delta}$, i.e.:

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_{\text{ref}} + \bar{A}(\boldsymbol{x} - \boldsymbol{x}_{\text{ref}}), \tag{6}$$

and the matrix $\bar{A}$ is Hurwitz, i.e. all its eigenvalues have negative real parts. Additionally, $V\colon \boldsymbol{x}_{\Delta} \mapsto \boldsymbol{x}_{\Delta}^{\mathsf{T}} P \boldsymbol{x}_{\Delta}$ is a Lyapunov function ($V(\boldsymbol{0}) = 0$ and for all $\boldsymbol{x}_{\Delta} \neq 0$, it holds $V(\boldsymbol{x}_{\Delta}) > 0$ and $\dot{V}(\boldsymbol{x}_{\Delta}) < 0$). The solutions of Eq. (6) can be written: $\boldsymbol{x}(t) = \boldsymbol{x}_{\text{ref}}(t) + e^{\bar{A}(t-t_0)}\boldsymbol{x}_{\Delta}(t_0)$. Since $\bar{A}$ is Hurwitz, the term $e^{\bar{A}(t-t_0)}$ tends to 0 exponentially fast, and the tracking asymptotically converges towards the reference trajectory $\boldsymbol{x}_{\text{ref}}(t)$. The Lyapunov function V can be used to define control funnels as follows. For $\alpha > 0$, we let:

$$\mathcal{F}_{\alpha}(t) = \{\boldsymbol{x}_{\text{ref}}(t) + \boldsymbol{x}_{\Delta} \mid V(\boldsymbol{x}_{\Delta}) \leq \alpha\} \tag{7}$$

$\mathcal{F}$ is a control funnel defined over $\mathbb{R}$: if $\boldsymbol{x}_{\Delta}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t)$ is a solution of Eq. (5) such that $\boldsymbol{x}(t_1) \in \mathcal{F}_{\alpha}(t_1)$, then for any $t_2 > t_1$, since $V(\boldsymbol{x}_{\Delta})$ only decreases, $V(\boldsymbol{x}_{\Delta}(t_2)) \leq V(\boldsymbol{x}_{\Delta}(t_1)) \leq \alpha$, and thus $\boldsymbol{x}(t_2) = \boldsymbol{x}_{\text{ref}}(t_2) + \boldsymbol{x}_{\Delta}(t_2) \in \mathcal{F}_{\alpha}(t_2)$.

$\mathcal{F}_{\alpha}(t)$ is a fixed d-dimensional ellipsoid centered at $\boldsymbol{x}_{ref}(t)$. Without going into details, it is possible to get lower bounds on the rate of decay of $V(\boldsymbol{x}_{\Delta})$, and effectively compute $\beta > 0$ such that, for any solution $\boldsymbol{x}_{\Delta}(t)$ of Eq. (5):

$$\forall t \in \mathbb{R}, \forall \delta t \in \mathbb{R}^+,\ V(\boldsymbol{x}_{\Delta}(t + \delta t)) \leq e^{-\beta.\delta t} V(\boldsymbol{x}_{\Delta}(t)) \tag{8}$$

This proves that if the system is inside the control funnel $\mathcal{F}_{\alpha}(t)$ at a given instant, then after letting time elapse for a duration of δt, the system will be inside the control funnel $\mathcal{F}_{\alpha e^{-\beta.\delta t}}(t)$. Using the terminology of Section 2.3, this can be equivalently stated as follows: for $0 < \alpha' < \alpha$, the control funnel $\mathcal{F}_{\alpha'}(t)$ $\left[\frac{1}{\beta}\log(\frac{\alpha}{\alpha'})\right]$-absorbs the control funnel $\mathcal{F}_{\alpha}(t)$. Thanks to this property, for a given LQR controller and a reference trajectory $\boldsymbol{x}_{\text{ref}}(t)$, we can define a finite set of fixed-size control funnels $\mathcal{F}_{\alpha_0}(t), \mathcal{F}_{\alpha_1}(t), \ldots, \mathcal{F}_{\alpha_q}(t)$, with $\alpha_0 > \alpha_1 > \cdots > \alpha_q > 0$, and absorbing transitions between them in the corresponding timed automaton.

In the remainder of the article, we will only use this kind of fixed size control funnels, which we call "LQR funnels". They are convenient because the larger

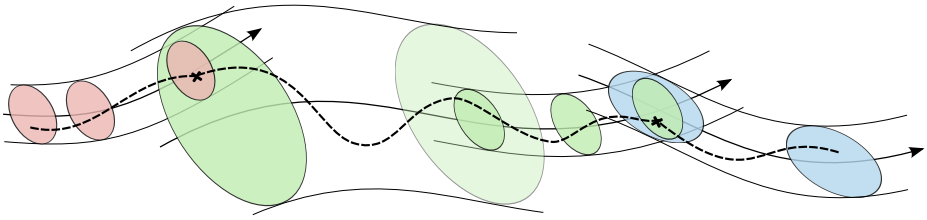

Fig. 3. An absorbing transition (in green) between two switching transitions.

ones can be used to "catch" other control funnels, and the smaller ones can easily be caught by other control funnels. Figure 3 depicts a typical sequence, where first a large control funnel (in green) catches the system, then after some time, an absorbing transition can be triggered, and finally, a new transition brings the system to a larger control funnel (in blue) on another trajectory. Besides that, testing for inclusion between fixed-size ellipsoids is easy, and therefore LQR funnels allow relatively efficient algorithms for the computation of the tuples needed for the timed-automaton reduction ($(\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (k, l))$, $(\mathsf{invariant}, S_{i,j}, (i, j))$, ..., see Section 3).

It should be noted that the concepts of fixed size control funnels and absorbing transitions, introduced here for linear systems, are also suitable for general nonlinear systems. Lyapunov functions in general, and quadratic ones in particular, can be computed via optimization, for example with Sum-of-Squares techniques as shown in [17]. By imposing specific constraints on the optimization, fixed size control funnels with exponential convergence can be obtained inducing the same kind of absorbing transitions as introduced in the last paragraph.

5 Examples of application

5.1 Synchronization of sine waves

In this example, there is a unique reference trajectory: $x_{\text{ref}}(t) = \sin(\frac{2\pi}{\tau}t)$, for $t \in [0, \tau]$ and $\tau \in \mathbb{Q}$, and a unique LQR controller tracking this trajectory. We define two fixed size LQR funnels $\mathcal{F}^1$ (the large one) and $\mathcal{F}^2$ (the small one) defined over $[0, \tau]$ such that $\mathcal{F}^2$ γ-absorbs $\mathcal{F}^1$ for some $\gamma \in \mathbb{Q}$. The size of $\mathcal{F}^1$ is computed so that an upper bound on the acceleration is always ensured, as long as the state of the system remains inside the control funnel.

The set $\mathcal{F}^1(\tau/2)$ contains the smaller control funnel $\mathcal{F}^2(t)$ for a range of time values $[\alpha, \beta]$ for some $\alpha < \frac{\tau}{2} \in \mathbb{Q}$ and $\beta > \frac{\tau}{2} \in \mathbb{Q}$. This allows switching transitions from $\mathcal{F}^2$ to $\mathcal{F}^1$ with abrupt modifications of the controller clock c_t. Together with the absorbing transition and "cyclic transitions" that come from the equalities $\mathcal{F}^1(0) = \mathcal{F}^1(\tau)$ and $\mathcal{F}^2(0) = \mathcal{F}^2(\tau)$, it results in an abstraction by the timed automaton shown on the left side of Fig. 4. The goal is to synchronize the controlled signal to a fixed signal $\sin(\frac{2\pi}{\tau}t + \varphi_0)$. The phase φ_0 is initially

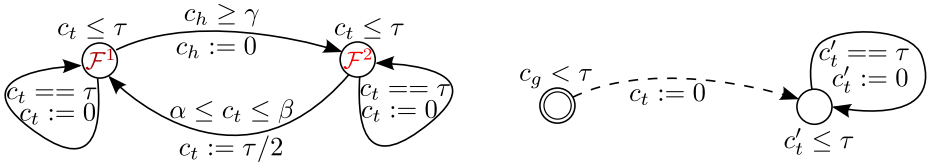

Fig. 4. *On the left:* the timed automaton for the controlled signal (the system). *On the right:* the timed automaton used to model the target signal with an initially unknown phase φ_0. The opponent transition (dashed) is the one used to set φ_0.

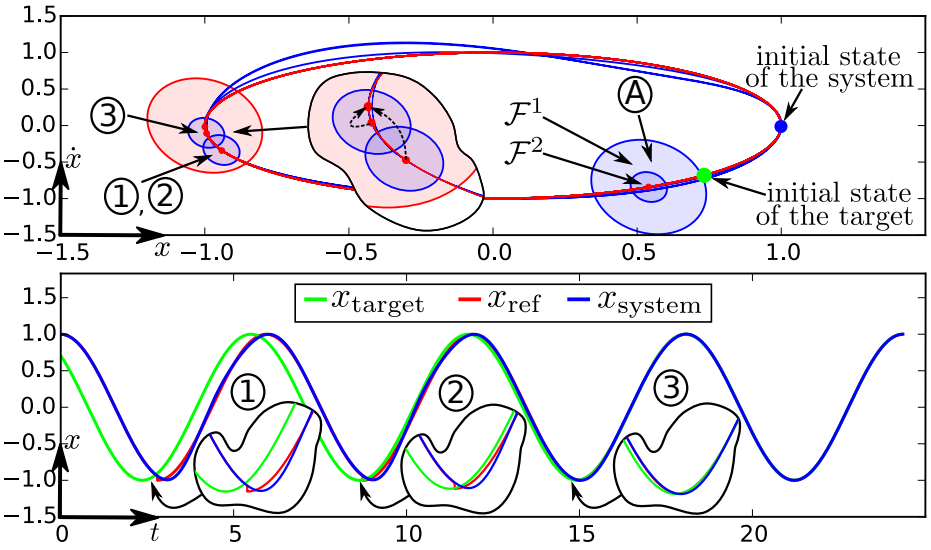

Fig. 5. The reactive controller performs three switching transitions to exactly adjust its phase to that of the target signal.

unknown, which we model using an adversary: we use a new clock c'_t, and an opponent transition as in the timed automaton on the right side of Fig. 4.

With these two timed automata, we can use the tool Uppaal-Tiga to synthesize a controller that reacts to the choice of the adversary, and performs adequate switching transitions until $c_t = c'_t$. It is even possible to generate a strategy that guarantees that the synchronization can always be performed in a bounded amount of time. We show in Fig. 5 a trajectory generated by the synthesized reactive controller. In this example, the phase chosen by the adversary is such that it is best to accelerate the controlled signal. Therefore, the controller uses twice the switching transition from $\mathcal{F}^2$ to $\mathcal{F}^1$ with a reset of the controller clock from α to $\tau/2$ (① and ② in Fig. 5). Between these switching transitions, an absorbing transition is taken to go back to the control funnel $\mathcal{F}^2$ (Ⓐ in Fig. 5). After the first two switching transitions, the remaining gap ε between c_t and c'_t is smaller

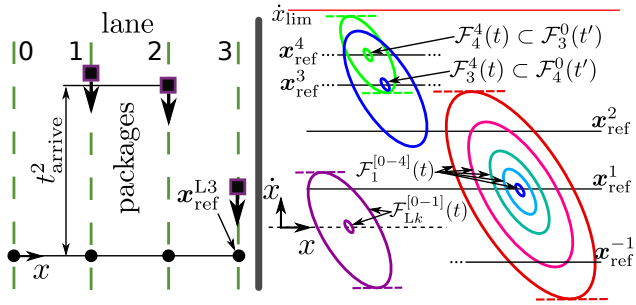

Fig. 6. The figure on the left shows the set-up. The black dots correspond to the position of the lanes. On the right are shown some LQR funnels along the constant velocity reference trajectories in the state space.

than $\frac{\tau}{2} - \alpha$, and therefore the controller waits a bit longer (until $\frac{\tau}{2} - \varepsilon$) to perform the switching transition that exactly synchronizes the two signals (③ in Fig. 5).

This example shows that our abstraction can be used for reactive controller synthesis via timed games. The main advantage of our approach over methods based on full discretization is that, since a continuous notion of time is kept in our abstraction, the reactive strategy is theoretically able to *exactly* synchronize the controlled signal to any real value of φ_0. One of our hopes is that extensions of this result can lead to a general formal approach for signal processing.

5.2 A 1D pick-and-place problem

In this second example, we show that timed-automata abstractions based on control funnels can be used to perform non-trivial planning. We propose a one-dimensional pick-and-place scenario. The set-up consists of a linear system controlled in acceleration moving along a straight line. On this line, four positions are defined as lanes (see Fig. 6). On three of these lanes (1, 2 and 3), packages arrive that have to be caught at the right time by the system and later delivered to lane 0. The system has limited acceleration and velocity, and can carry at most two packages at a time.

The LQR funnels in this example are constructed based on 12 reference trajectories. The first four have different constant positive velocities ($\boldsymbol{x}^i_{\text{ref}}$ with $i \in \{1, \ldots, 4\}$, the fastest one being $\boldsymbol{x}^4_{\text{ref}}$, and the slowest one $\boldsymbol{x}^1_{\text{ref}}$). The next four are the same trajectories with negative velocities. On each of these reference trajectories, five different control funnels of constant size are defined ($\mathcal{F}^j_i$ for $j \in \{0, \ldots, 4\}$, the largest one being $\mathcal{F}^0_i$). The control funnels with negative constant velocity are the mirror image of those with positive velocity. Additionally, four stationary trajectories $\boldsymbol{x}^{\text{L}k}_{ref}$ (with $k \in \{0, \ldots, 3\}$) at the positions of the lanes are defined. The controllers associated to these trajectories simply stabilize the system at lane positions. For each of these trajectories a small ($j = 0$) and a large ($j = 1$) control funnel are constructed. They are denoted by $\mathcal{F}^j_{\text{L}k}$. By construction, neighboring trajectories (e.g. $\boldsymbol{x}^3_{\text{ref}}$ and $\boldsymbol{x}^2_{\text{ref}}$ or $\boldsymbol{x}^1_{\text{ref}}$ and $\boldsymbol{x}^{-1}_{\text{ref}}$) are connected, meaning that for two neighboring trajectories $\boldsymbol{x}^i_{\text{ref}}$ and $\boldsymbol{x}^k_{\text{ref}}$, $\forall t \in I_i$, $\exists t' \in I_k$ s.t. $\mathcal{F}^4_i(t) \subset \mathcal{F}^0_k(t')$ (see Fig. 6). This allows the system to reach a higher or lower velocity without the need of an explicitly defined

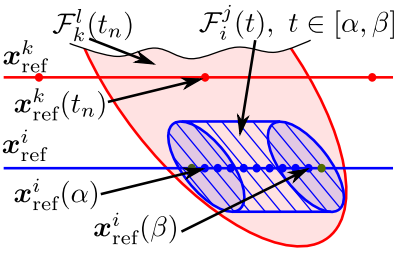

Fig. 7. In order to define the tuples $(\mathsf{switch}, [\alpha, \beta], (i, j), \gamma, (k, l))$ (see Section 3), N regularly spaced points are chosen in $\boldsymbol{x}^k_{ref}$ (defining the ellipses $\mathcal{F}_k^l(t_n)$ for $n \in \{1, \ldots, N\}$), and for each n, we set $\gamma = t_n$, and if a point $\boldsymbol{x}^i_{ref}(t)$ such that $\mathcal{F}_i^j(t) \subset \mathcal{F}_k^l(\gamma)$ is found, an incremental search is performed to define a range $[\alpha, \beta]$ such that $\forall t \in [\alpha, \beta],\ \mathcal{F}_i^j(t) \subset \mathcal{F}_k^l(\gamma)$.

acceleration trajectory. While the abstraction based on these control funnels does not represent all the possible behaviors of the system (it is not complete), switching between different velocity references allows the system to perform a great variety of trajectories with continuous and bounded velocity and bounded acceleration.

To fully specify the timed-automata abstraction, the tuples defining the transition guards must be computed (see Section 3). Here, the regions of invariance defining the funnels are identically-shaped ellipses (only translated along a reference trajectory and scaled), thus the test for inclusion is computationally very cheap. Therefore, many points can be tested for inclusion on each trajectory, as depicted in Fig. 7, which leads to precise ranges for the switching transitions. Since the funnels are fixed sets translated along reference trajectories, knowing velocity or acceleration bounds on these references, and using offsets in the inclusion tests, we can ensure inclusion on the whole range of a switching transition with only a finite number of inclusion tests.

We consider an example where three packages respectively arrive on lanes 3, 2 and 1 at times $t^1_{\text{arrive}} = 40$, $t^2_{\text{arrive}} = 111$ and $t^3_{\text{arrive}} = 122$ (corresponding equality tests on c_g can be used to refer to these moments in the timed automaton abstraction). The goal is to find a trajectory that catches all the packages and delivers them to lane 0. At the moment of the catch ($c_g = t^p_{\text{arrive}}$), the reference $\boldsymbol{x}^i_{\text{ref}}$ tracked by the system must be exactly at the correct position (i.e. on the lane of the arriving package). Depending on the reference trajectory, this corresponds to a particular value of c_t. We add the following constraints on the catches: an upper bound on velocity such that the system cannot be tracking $\boldsymbol{x}^4_{\text{ref}}$, $\boldsymbol{x}^3_{\text{ref}}$, $\boldsymbol{x}^{-3}_{\text{ref}}$ or $\boldsymbol{x}^{-4}_{\text{ref}}$ when it catches a package, and a bound on uncertainty such that the system must be in a small control funnel to catch a package. Using additional constructions in our timed automaton abstraction (for example a bounded counter that keeps track of the number of packages being carried by the system), it is easy to specify these constraints and the objective as a reachability specification that can be checked by Uppaal. Uppaal outputs a timed word that corresponds to the schedule of control law switches and the trajectory shown on Fig. 8, which successfully catches the packages and delivers them to lane 0.

The two upper graphs of Fig. 8 show the evolution of the system in its state space and some of the regions of invariance when taking a *switching transition* (colored ellipses). The green dots mark positions at which *absorbing transitions* take place ($\mathcal{F}_i^j \to \mathcal{F}_i^{j+1}$). Purple crosses represent a package. The lower graph

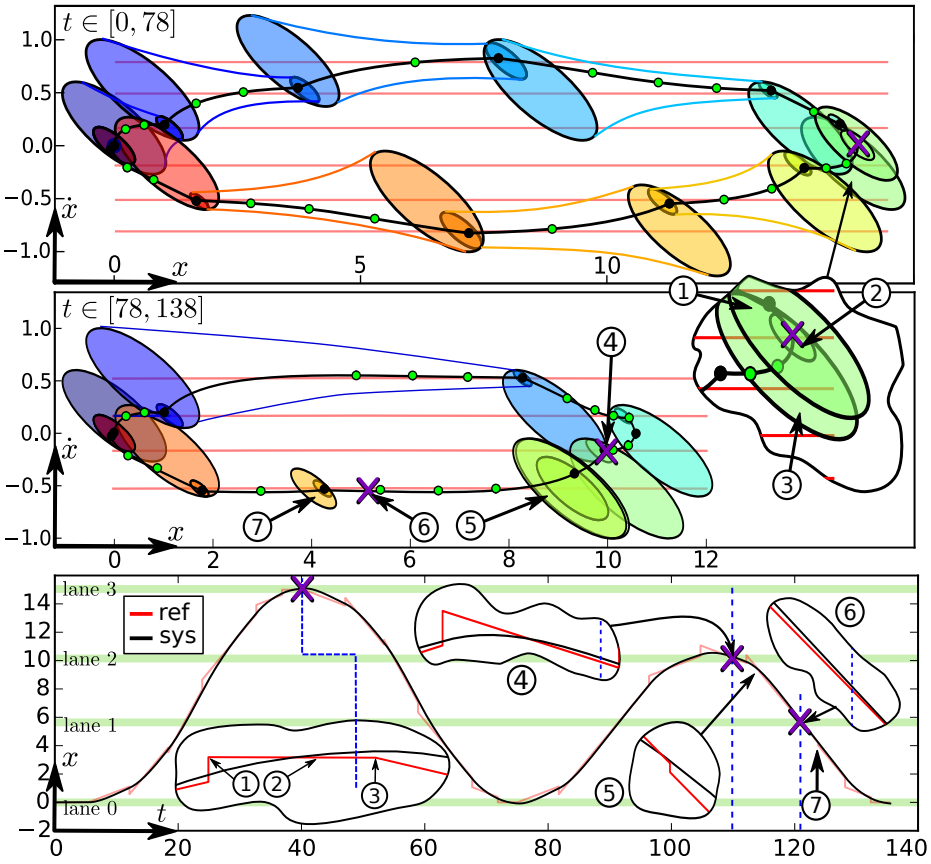

Fig. 8. Execution of a succeeding control strategy given as a timed word.

compares the evolution of the position of the real system with the reference. One can see that even though the reference velocity can only take seven different values, a relatively smooth trajectory is realized. Before catching the first package, the system switches from $\mathcal{F}_1^4$ to $\mathcal{F}_{\text{L3}}^0$ ①. It then converges to $\mathcal{F}_{\text{L3}}^1$ ② just before the catch. The difference between the real system position and the reference is very small at that point in time. The system then switches to $\mathcal{F}_{-1}^0$ ③ in order to return to lane 0. It is interesting to notice that the system chooses to return to lane 0 after having picked only one package, therefore adopting a non-greedy strategy. This is because it wouldn't have time to perform a delivery to lane 0 between the arrival of the second and third packages.

When the second package arrives on lane 2, the system catches it while being in $\mathcal{F}_{-1}^4$ ④ . This is again a non-trivial behavior: in order to get both the second and the third packages, the system has to first go a little bit further than lane 2 so as to be able to catch the two packages without violating the limit on acceleration. A slight adjustment of the reference position ⑤ has to be done to

catch the third package exactly on time ⑥. After that, the system performs a local acceleration ⑦ to reach lane 0 as soon as possible, and delivers the two packages.

6 Conclusion and future work

We have presented a timed-automata abstraction of switched dynamical systems based on control funnels, i.e. time-varying regions of invariance. Applying verification tools (such as Uppaal) on this abstraction, one can solve motion planning or more complicated problems with timing requirements. In the example of Section 5.2, we are able to generate a non-trivial solution for a pick-and-place problem. Synthesis of controllers that react to the environment can be done by solving timed games, and in the example of Section 5.1 we use Uppaal-Tiga to generate a controller that can reactively adjust the phase of a signal controlled in acceleration.

To go further and improve our abstraction, as mentioned in Section 3 (Remark 1), we could use non-deterministic clock updates and study the related decidability issues. We could also exploit the specific structure of the timed automata used in our abstraction and design dedicated verification and synthesis algorithms. Indeed, the timed automata of our model have three clocks, and there is non-determinism for only one of them (c_t). This makes us believe that we could potentially outperform the general algorithms of Uppaal and Uppaal-Tiga and solve more complex problems. Finally, in this quest to scale our approach up to larger models and more advanced specifications, we also plan to combine it to numerical and optimization methods.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Computer Science 126(2), 183–235 (Apr 1994)
2. Asarin, E., Maler, O., Pnueli, A.: Reachability analysis of dynamical systems having piecewise-constant derivatives. Theor. Computer Science 138(1), 35–65 (Feb 1995)
3. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller synthesis for timed automata. In: SSSC'98. pp. 469–474. Elsevier (1998)
4. Aubin, J.P.: Viability tubes. In: Modelling and Adaptive Control, pp. 27–47. Springer (1988)
5. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., Lime, D.: UPPAAL-Tiga: Time for playing games! In: CAV'07. LNCS, vol. 4590, pp. 121–125. Springer (Jul 2007)
6. Behrmann, G., David, A., Larsen, K.G., Håkansson, J., Pettersson, P., Yi, W., Hendriks, M.: Uppaal 4.0. In: QEST'06. pp. 125–126. IEEE (Sep 2006)
7. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Updatable timed automata. Theor. Computer Science 321(2-3), 291–345 (Aug 2004)
8. David, A., Grunnet, J.D., Jessen, J.J., Larsen, K.G., Rasmussen, J.I.: Application of model-checking technology to controller synthesis. In: FMCO'10. LNCS, vol. 6957, pp. 336–351. Springer (Nov 2012)

9. DeCastro, J., Kress-Gazit, H.: Synthesis of nonlinear continuous controllers for verifiably-correct high-level, reactive behaviors. IJRR 34(3), 378–394 (Mar 2014)
10. Duggirala, P.S., Mitra, S., Viswanathan, M.: Verification of annotated models from executions. In: EMSOFT'13. pp. 1–10. IEEE (Sep 2013)
11. Frazzoli, E., Dahleh, M.A., Feron, E.: Maneuver-based motion planning for nonlinear systems with symmetries. IEEE Trans. Robotics 21(6), 1077–1091 (Dec 2005)
12. Fu, J., Topcu, U.: Computational methods for stochastic control with metric interval temporal logic specifications. Tech. Rep. 1503.07193, ArXiv (Mar 2015)
13. Julius, A.A., Pappas, G.J.: Trajectory based verification using local finite-time invariance. In: HSCC'09. LNCS, vol. 5469, pp. 223–236. Springer (Apr 2009)
14. Koiran, P., Cosnard, M., Garzon, M.: Computability with low-dimensional dynamical systems. Theor. Computer Science 132(1), 113–128 (Sep 1994)
15. Le Ny, J.L., Pappas, G.J.: Sequential composition of robust controller specifications. In: ICRA'12. pp. 5190–5195. IEEE (May 2012)
16. Liu, J., Prabhakar, P.: Switching control of dynamical systems from metric temporal logic specifications. In: ICRA'14. pp. 5333–5338. IEEE (May 2014)
17. Majumdar, A., Ahmadi, A.A., Tedrake, R.: Control design along trajectories with sums of squares programming. In: ICRA'13. pp. 4054–4061. IEEE (May 2013)
18. Majumdar, A., Tedrake, R.: Robust online motion planning with regions of finite time invariance. In: WAFR'12. STAR, vol. 86, pp. 543–558. Springer (Jun 2013)
19. Maler, O., Batt, G.: Approximating continuous systems by timed automata. In: FMSB'08. LNBI, vol. 5054, pp. 77–89. Springer (Jun 2008)
20. Maler, O., Manna, Z., Pnueli, A.: From timed to hybrid systems. In: Real-time: theory in practice. LNCS, vol. 600, pp. 447–484. Springer (1992)
21. Mason, M.T.: The mechanics of manipulation. In: ICRA'85. vol. 2, pp. 544–548. IEEE (Mar 1985)
22. Quottrup, M.M., Bak, T., Zamanabadi, R.I.: Multi-robot planning : a timed automata approach. In: ICRA'04. vol. 5, pp. 4417–4422. IEEE (Apr 2004)
23. Sloth, C., Wisniewski, R.: Timed game abstraction of control systems. Tech. Rep. 1012.5113, ArXiv (Dec 2010)
24. Sloth, C., Wisniewski, R.: Complete abstractions of dynamical systems by timed automata. Nonlinear Analysis: Hybrid Systems 7(1), 80–100 (Feb 2013)
25. Sontag, E.D.: Mathematical control theory: deterministic finite dimensional systems. Springer (1998)
26. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, J.W.: LQR-trees: Feedback motion planning via sums-of-squares verification. IJRR 29(8), 1038–1052 (Jul 2010)