A Comparison of Control Problems for Timed and Hybrid Systems*

Franck Cassez[1]**, Thomas A. Henzinger[2]***, and Jean-François Raskin[3]†

[1] IRCCyN, Ecole Centrale de Nantes, France
[2] EECS Department, University of California, Berkeley, USA
[3] Département d'Informatique, Université Libre de Bruxelles, Belgium

Abstract. In the literature, we find several formulations of the control problem for timed and hybrid systems. We argue that formulations where a controller can cause an action at any point in dense (rational or real) time are problematic, by presenting an example where the controller must act faster and faster, yet causes no Zeno effects (say, the control actions are at times $0, \frac{1}{2}, 1, 1\frac{1}{4}, 2, 2\frac{1}{8}, 3, 3\frac{1}{16}, \ldots$). Such a controller is, of course, not implementable in software. Such controllers are avoided by formulations where the controller can cause actions only at discrete (integer) points in time. While the resulting control problem is well-understood if the time unit, or "sampling rate" of the controller, is fixed a priori, we define a novel, stronger formulation: the *discrete-time control problem with unknown sampling rate* asks if a sampling controller exists for *some* sampling rate. We prove that this problem is undecidable even in the special case of timed automata.

1 Introduction

Timed and hybrid systems are dynamical systems with both discrete and continuous components. A paradigmatic example of a hybrid system is a digital control program for an analog plant environment, like a furnace or an airplane: the controller state moves discretely between control modes, and in each control mode, the plant state evolves continuously according to physical laws. A natural model for hybrid systems is the *hybrid automaton*, which represents discrete components using finite-state machines and continuous components using real-numbered variables whose evolution is governed by differential equations or differential inclusions [ACH+95]. An interesting class of hybrid automata is the

* An abbreviated version of this paper will appear in the *Proceedings of the Fifth International Workshop on Hybrid Systems: Computation and Control* (HSCC), Lecture Notes in Computer Science, Springer-Verlag, 2002.
** Partially supported by the FNRS, Belgium, under grant 1.5.096.01.
*** Partially supported by the DARPA SEC grant F33615-C-98-3614, the AFOSR MURI grant F49620-00-1-0327, the NSF Theory grant CCR-9988172, and the MARCO GSRC grant 98-DT-660.
† Partially supported by a "Crédit aux chercheurs" from the Belgian National Fund for Scientific Research.

class of *rectangular automata* [HKPV98]: this is the most liberal generalization of *timed automata* [AD94] with interesting decidability properties. While in timed automata, all real-numbered variables are "clocks" with constant derivative 1, in rectangular automata, the derivatives of all variables are bounded by constants from below and above. If the bounds on the derivative of a variable cannot change between control modes unless the variable is reset, then the rectangular automaton is called *initialized*.

The distinction between continuous evolutions of the plant state (which is given by the real-numbered variables of a hybrid automaton) and discrete switches of the controller state (which is given by the location, or control mode, of the hybrid automaton) permits a natural formulation of the *safety control problem*: given an unsafe set U of plant states, is there a strategy to switch the controller state in real time so that the plant can be prevented from entering U? In other words, the hybrid automaton specifies a set of possible control modes, together with the plant behavior resulting from each mode, and the control problem asks for deriving a switching strategy between control modes that keeps the plant out of trouble. A switch in control mode can be caused by two events: sensing of a particular plant state (e.g., "turn off the furnace when the heat hits x degrees"); or arrival of a particular time instant (e.g., "turn off the furnace after y seconds"). This can be formalized in several ways, and consequently, a variety of similar but different mathematical formulations of the safety control problem for timed and hybrid systems can be found in the literature (e.g., [HW92,AMP95,MPS95,Won97,HHM99,HK99,ABD+00,TLS00]). We classify various formulations, compare their relative strengths, and fill in some gaps that had been left open previously. In doing so, we focus on *safety* control purely for reasons of clarity; the treatment of more general control objectives is orthogonal to the issues studied here [dAHM01].

One fundamental distinction is between *dense-time* and *discrete-time* (or *sampling*) models of control. In dense-time models, the controller observes the plant state continuously, and may cause a control switch as soon as a particular plant state is encountered. In discrete-time models, while the plant evolves continuously, the controller observes the plant state only at regularly spaced time instants, and may cause a control switch only at those sampling points. With discrete-time models, we can further distinguish between *known* and *unknown sampling rate*. A control problem of the first kind asks: given a positive rational number β, can an unsafe plant state be prevented by a controller that samples the plant state every β seconds? The more general discrete-time control problem with unknown sampling rate asks: is there a rational number β such that an unsafe plant state be prevented by a controller with sampling rate β? For example, a discrete-time control problem may not be solvable with sampling rate of 10 ms, but it may be solvable with sampling rate of 5 ms.

Similarly, with dense-time models, we can distinguish between *known* and *unknown switch conditions*. A control problem of the first kind asks: for each control mode, given a set P of predicates on the plant state, can an unsafe plant state be prevented by a controller that watches the predicates in P and

causes a mode switch only when the truth value of a predicate in P changes. The more general dense-time control problem with unknown switch conditions asks: is there a set P of predicates (typically chosen from some language) on the plant state such that an unsafe plant state be prevented by a controller choosing switch conditions from P? For example, a dense-time control problem may not be solvable by a controller that can switch mode by watching a sensor that indicates when a furnace hits a multiple of 10 degrees, but it may be solvable if the sensor has a granularity of 5 degrees.

The following three control problems have been studied in the literature: (1) The *known sampling rate* (known or unknown switch conditions) discrete-time control problem can be solved algorithmically for all rectangular automata [HK99] (see [HW92] for the timed case). (2) The *known switch conditions* dense-time control problem, while undecidable for general rectangular automata, is solvable for all initialized rectangular automata [HHM99]. (3) The *unknown switch conditions* dense-time control problem, while undecidable for initialized rectangular automata [HKPV98], is solvable for all timed automata [MPS95]. While none of these papers explicitly mention switch conditions, we obtain equivalent formulations in terms of switch conditions, which allow us to see that the three formulations (1)–(3) are of strictly increasing generality. Intuitively, if all switch conditions are known, then they are part of the hybrid automaton model, and the task of the controller is simply to disable or enable edge guards: at each plant state, the controller decides whether to cause a mode switch, or let an amount of time pass which is constrained only by the location invariant of the hybrid automaton. Since all timing constraints are already part of the model (in the form of location invariants and edge guards), this is also called *time-abstract control* [HHM99]. By contrast, in [MPS95,AMP95], the controller may strengthen the location invariants of the hybrid automaton model to achieve the control objective: at each plant state, the controller decides whether to cause a mode switch, or let an amount of time pass which is constrained by some new, derived predicate on the plant state. If the invariants are strengthened by polyhedral predicates, as in [AMP95,MPS95], this is called *polyhedral control*.

It is well-known that careless solutions of *dense-time* control problems may lead to unrealistic controllers that "stop time," either by blocking the progress of time altogether, or by causing mode switches at convergent points in time, say, $0, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$ [Won97]. Such Zeno phenomena can be avoided by lifting the control objective from a safety objective to a combined safety and liveness objective ("always avoid unsafe states and let time progress for another unit") [dAHM01]. However, we show for the first time that dense-time control models are more seriously flawed: we present an example that can be controlled by a dense-time controller but not by any discrete-time controller with unknown sampling rate. The dense-time controller causes mode switches at divergent points in time, say, $0, \frac{1}{2}, 1, 1\frac{1}{4}, 2, 2\frac{1}{8}, 3, 3\frac{1}{16}, \ldots$, where the difference between adjacent time instants cannot be bounded from below. Such a controller is, of course, not implementable by software. This observation has led us to define the *unknown sampling rate* discrete-time control problem, which asks whether a discrete-time controller ex-

ists for *some* sampling rate, and if so, for a derivation of the sampling rate. We show that this problem is, surprisingly, undecidable even in the very special case of timed automata with known switching conditions. The undecidability proof is interesting, because it requires the encoding of unbounded integer values using the difference between two clocks.

We believe that our results have practical significance for two reasons. First, they put in question the usefulness of dense-time control formulations. Some authors insist, for example, that the number of dense-time control actions in a time unit is bounded from above [ABD+00]. But even such a bound does not prevent pathological examples like the one mentioned before, where there are no more than two control actions in any one time unit, yet the time intervals between control actions must get smaller and smaller. This can be avoided only if the time difference between control actions is bounded by a constant ε from below, which, however, fundamentally changes the control problem to a discrete-time problem with time unit ε. Second, our results show that even discrete-time control problems can be solved only when the sampling rate, or time unit, is fixed *a priori*. It follows that from a decidability point of view, the control of timed and hybrid systems is considerably more difficult than we thought when we embarked on our original goal of comparing different formulations found in the literature. (This, of course, does not affect the applicability of symbolic semi-decision procedures such as [Won97,ABD+00,TLS00,dAHM01]).

The paper is organized as follows. In Section 2, we recall the definitions of safety control for transition systems, as well as the definitions of timed and rectangular automata. In Section 3, we formulate and compare the four basic variations of the safety control problem for hybrid systems: known switch conditions vs. unknown switch conditions dense-time, and known sampling rate vs. unknown sampling rate discrete-time. We also show that a dense-time controller may exist in cases where no sampling rate exists. Section 4 contains the proof that the existence of digital controllers —i.e., the unknown sampling rate discrete-time control problem— is undecidable for timed automata.

2 Prerequisites

2.1 Labeled transition systems and control

A *labeled transition system* S is a tuple $(Q, \Sigma, \rightarrow, Q_0)$, where Q is a (possibly infinite) set of states, Σ is a (possibly infinite) set of labels, $\rightarrow \subseteq Q \times \Sigma \times Q$ is a labeled transition relation, and $Q_0 \subseteq Q$ is a set of initial states. We write $q \xrightarrow{a} q'$ if $(q, a, q') \in \rightarrow$. A *run* of S is a finite sequence $\rho = q_0 a_0 q_1 a_1 \ldots q_n$ of states $q_i \in Q$ and labels $a_i \in \Sigma$ such that (1) $q_0 \in Q_0$ and (2) $q_i \xrightarrow{a_i} q_{i+1}$ for all $0 \leq i < n$. We write $dest(\rho) = q_n$ for the final state of ρ. A state $q \in Q$ is *reachable in* S if there exists a run ρ of S such that $q = dest(\rho)$. A set $F \subseteq Q$ of states is *reachable in* S if there exists a state in F that is reachable in S.

Definition 1 (Reachability). *The* reachability problem *for a class $\mathcal{C}$ of labeled transition systems is the following: given a labeled transition system $S \in \mathcal{C}$ and a set F of states of S, determine if F is reachable in S.* □

The labels of a labeled transition system S can be interpreted as control actions. A label $a \in \Sigma$ is *enabled* at the state $q \in Q$ if $q \xrightarrow{a} q'$ for some state $q' \in Q$. We write $\mathit{Enabled}(q)$ for the labels that are enabled at q. A *control map* for S is a function $\kappa\colon Q \to 2^{\Sigma}$ that maps every state $q \in Q$ to a set $\kappa(q) \subseteq \mathit{Enabled}(q)$ of enabled labels. The *closed-loop system* $\kappa(S)$ is the labeled transition system $(Q, \Sigma, \to_\kappa, Q_0)$, where $q \xrightarrow{a}_\kappa q'$ iff $q \xrightarrow{a} q'$ and $a \in \kappa(q)$. The control map κ is *deadlock-free* for S if $\kappa(q) \neq \emptyset$ for every state q that is reachable in the closed-loop system $\kappa(S)$.

Definition 2 (Safety control). *The* safety control problem *for a class $\mathcal{C}$ of labeled transition systems is the following: given a labeled transition system $S \in \mathcal{C}$ and a set F of states of S, determine if there exists a deadlock-free control map κ for S such that F is not reachable in the closed-loop system $\kappa(S)$.* □

The *safety verification problem* is the special case of the safety control problem where $|\Sigma| = 1$. Note that the safety verification problem can be reduced to the non-reachability problem, and vice versa.

2.2 Timed and rectangular automata

Let X be a finite set of real-valued variables. A *valuation* for X is a function $v\colon X \to \mathbb{R}$. We write $[X \to \mathbb{R}]$ for the set of all valuations for X. For a set $V \subseteq [X \to \mathbb{R}]$ of valuations, and $x \in X$, define $V(x) = \{v(x) \mid v \in V\}$. Let $\bowtie \in \{<, \leq, >, \geq\}$, let $x_1, \ldots, x_n, x \in X$, and let $c_1, \ldots, c_n, c \in \mathbb{Z}$. A *rectangular inequality* over X is a formula of the form $x \bowtie c$; a *triangular inequality* over X is a formula of the form $x_1 - x_2 \bowtie c$; a *polyhedral inequality* over X is a formula of the form $c_1 x_1 + \cdots + c_n x_n \bowtie c$. A *rectangular predicate* over X is a conjunction of rectangular inequalities over X; the set of all rectangular predicates over X is denoted $\mathit{Rect}(X)$. A *polyhedral predicate* over X is a boolean combination of polyhedral inequalities over X. For a polyhedral predicate p over X, and a valuation v for X, we write $v(p)$ for the boolean value that is obtained from p by replacing each variable $x \in X$ with $v(x)$. The polyhedral predicate p defines the set $[\![p]\!] = \{v : X \to \mathbb{R} \mid v(p) = \mathsf{true}\}$ of valuations.

Definition 3 (Rectangular automaton [HKPV98]**).** *A* rectangular automaton *H is a tuple $(L, X, \Sigma, \mathsf{init}, E, \mathsf{inv}, \mathsf{flow})$. (1) L is a finite set of* locations *representing the discrete state of the automaton. (2) X is a finite set of real-valued* variables *representing the continuous state. We write $\dot{X} = \{\dot{x} \mid x \in X\}$ for the set of corresponding dotted variables, which represent first derivatives. (3) Σ is a finite set of* events *disjoint from $\mathbb{R}$. (4) $\mathsf{init}\colon L \to \mathit{Rect}(X)$ is the* initial condition*. If the automaton starts in location ℓ, then each variable $x \in X$ has a value in the interval $[\![\mathsf{init}(\ell)]\!](x)$. (5) $E \subseteq L \times \mathit{Rect}(X) \times \Sigma \times \mathit{Rect}(X) \times 2^X \times L$ is a finite set of* edges*. Every edge $(\ell, \gamma, \sigma, \alpha, U, \ell') \in E$ represents a change from location ℓ to location ℓ' with guard γ, label σ, and reset assignment α, which asserts that each variable $x \in U$ is nondeterministically reset to a value in the interval $[\![\alpha]\!](x)$, and each variable in $X \setminus U$ is left unchanged. (6) inv:*

$L \to \mathit{Rect}(X)$ *is the* invariant condition. *The automaton can stay in location* ℓ *as long as each variable* $x \in X$ *has a value in the interval* $[\![\mathsf{inv}(\ell)]\!](x)$. *(7)* flow*:* $L \to \mathit{Rect}(\dot{X})$ *is the* flow condition. *We require that for every location* $\ell \in L$ *and variable* $x \in X$*, the interval* $[\![\mathsf{flow}(\ell)]\!](\dot{x})$ *is bounded. If the automaton is in location* ℓ*, then each variable* $x \in X$ *can evolve nondeterministically with a derivative in the interval* $[\![\mathsf{flow}(\ell)]\!](\dot{x})$. □

Definition 4 (Initialized rectangular automaton). *An* initialized rectangular automaton *is a rectangular automaton that satisfies the following: for all edges* $(\ell, \gamma, \sigma, \alpha, U, \ell') \in E$ *and variables* $x \in X$*, if* $[\![\mathsf{flow}(\ell)]\!](x) \neq [\![\mathsf{flow}(\ell')]\!](x)$*, then* $x \in U$*. In other words, every time the bounds on the derivative of a variable change between two locations, the variable is reset.* □

Definition 5 (Timed automaton). *A* timed automaton *is an initialized rectangular automaton such that (1)* $\mathsf{flow}(\ell) \equiv \wedge_{x \in X}(\dot{x} = 1)$ *for all locations* $\ell \in L$*, and (2)* $\alpha \equiv \wedge_{x \in U}(x = 0)$ *for all edges* $(\ell, \gamma, \sigma, \alpha, U, \ell') \in E$*. In other words, all variables advance at the rate of time —they are called* clocks*— and variables can be reset only to 0.*[1] □

Let H be a rectangular automaton with locations L and variables X. The state space of H is $L \times [X \to \mathbb{R}]$; that is, every state $\langle \ell, v \rangle$ of H consists of a discrete part $\ell \in L$ and a continuous part $v\colon X \to \mathbb{R}$. A *polyhedral* (resp. *rectangular*) *state predicate* pred for H is a function that maps every location in L to a polyhedral (resp. rectangular) predicate $\mathsf{pred}(\ell)$ over X. Note that the initial and invariant conditions of H are rectangular state predicates for H. The polyhedral state predicate pred defines the set $[\![\mathsf{pred}]\!] = \{\langle \ell, v \rangle \in L \times [X \to \mathbb{R}] \mid v \in [\![\mathsf{pred}(\ell)]\!]\}$ of states. A state set $P \subseteq L \times [X \to \mathbb{R}]$ is *polyhedral* if there exists a polyhedral state predicate pred such that $[\![\mathsf{pred}]\!] = P$. We define three semantics for rectangular automata. The first two permit location changes at all real-valued points in time; the third only at multiples of a fixed sampling unit.

Definition 6 (Dense-time semantics). *The* dense-time semantics *of a rectangular automaton* $H = (L, X, \Sigma, \mathsf{init}, E, \mathsf{inv}, \mathsf{flow})$ *is the labeled transition system* $S_H^{dense} = (Q, \Sigma_{dense}, \to_{dense}, Q_0)$*, where* $Q = [\![\mathsf{inv}]\!]$ *and* $\Sigma_{dense} = \Sigma \cup \mathbb{R}_{>0}$ *and* $Q_0 = [\![\mathsf{init}]\!]$*, and* $\to_{dense}$ *is defined as follows. (1) Discrete transitions for each event* $\sigma \in \Sigma$*: let* $\langle \ell, v \rangle \xrightarrow{\sigma}_{dense} \langle \ell', v' \rangle$ *iff there exists an edge* $(\ell, \gamma, \sigma, \alpha, U, \ell') \in E$ *such that* $v \in [\![\gamma]\!]$*, and* $v'(x) \in [\![\alpha]\!](x)$ *for all* $x \in U$*, and* $v'(x) = v(x)$ *for all* $x \in X \backslash U$*. (2) Continuous transitions for each duration* $\delta \in \mathbb{R}_{>0}$*: let* $\langle \ell, v \rangle \xrightarrow{\delta}_{dense} \langle \ell', v' \rangle$ *iff* $\ell = \ell'$ *and for each variable* $x \in X$*, there exists a differentiable function* $f_x\colon [0, \delta] \to [\![\mathsf{inv}(\ell)]\!](x)$ *such that (i)* $f_x(0) = v(x)$*, (ii)* $f_x(\delta) = v'(x)$*, and (iii)* $\dot{f}_x(t) \in [\![\mathsf{flow}(\ell)]\!](x)$ *for all* $0 < t < \delta$. □

The time-abstract semantics hides the duration of continuous transitions.

[1] Unlike [AD94], we do not allow triangular inequalities in the definition of a timed automaton. However, every timed automaton with triangular inequalities can be transformed into one that accepts the same timed language and has only rectangular predicates.

Definition 7 (Time-abstract semantics). *Let* $H = (L, X, \Sigma, \mathsf{init}, E, \mathsf{inv}, \mathsf{flow})$ *be a rectangular automaton and* $S_H^{dense} = (Q, \Sigma_{dense}, \rightarrow_{dense}, Q_0)$ *its dense-time semantics. The* time-abstract semantics *of* H *is the labeled transition system* $S_H^{abstr} = (Q, \Sigma_{abstr}, \rightarrow_{abstr}, Q_0)$*, where* $\Sigma_{abstr} = \Sigma \cup \{time\}$ *and* $\rightarrow_{abstr}$ *is defined as follows. (1) For each* $\sigma \in \Sigma$*, let* $q \xrightarrow{\sigma}_{abstr} q'$ *iff* $q \xrightarrow{\sigma}_{dense} q'$*. (2) Let* $q \xrightarrow{time}_{abstr} q'$ *iff* $q \xrightarrow{\delta}_{dense} q'$ *for some duration* $\delta \in \mathbb{R}_{>0}$*.* □

If we assume that discrete transitions alternate with continuous transitions of a fixed duration β, then we obtain a discrete-time semantics with time unit β. The *sampling semantics*, with sampling unit β, further assumes that all discrete transitions (i.e., location changes) represent moves of the controller, and all continuous transitions (i.e., variable evolutions) represent moves of the plant. The intuition behind this semantics is that the plant evolves continuously, observed by a digital controller whose control decisions are separated by time β.[2]

Definition 8 (Sampling semantics). *Let* $H = (L, X, \Sigma, \mathsf{init}, E, \mathsf{inv}, \mathsf{flow})$ *be a rectangular automaton and* $S_H^{dense} = (Q, \Sigma_{dense}, \rightarrow_{dense}, Q_0)$ *its dense-time semantics. Let* $\beta \in \mathbb{Q}_{>0}$ *be a sampling unit. The* β-sampling semantics *of* H *is the labeled transition system* $S_H^{sample}(\beta) = (Q \times \{Control, Plant\}, \Sigma_{sample}, \rightarrow_{sample}, Q_0 \times \{Control\})$*, where* $\Sigma_{sample} = \Sigma \cup \{\beta\}$ *and* $\rightarrow_{sample}$ *is defined as follows. (1) Discrete control transitions for each* $\sigma \in \Sigma$*: let* $\langle q, Control\rangle \xrightarrow{\sigma}_{sample} \langle q', Plant\rangle$ *iff* $q \xrightarrow{\sigma}_{dense} q'$*. (2) Continuous plant transitions of fixed duration* $\beta \in \mathbb{Q}_{>0}$*: let* $\langle q, Plant\rangle \xrightarrow{\beta}_{sample} \langle q', Control\rangle$ *iff* $q \xrightarrow{\beta}_{dense} q'$*.* □

3 Control of Timed and Rectangular Automata

3.1 Dense-time control

In dense-time control, the controller can make a decision to change location — i.e., to switch control mode— at every real-valued time instant $t \in \mathbb{R}_{\geq 0}$. We distinguish between dense-time control with known switch conditions [HHM99] and unknown switch conditions [AMP95,MPS95,AMPS98].

Known switch conditions. A known switch conditions dense-time controller decides in every state either to issue a control action or to let time pass. If the controller issues a control action σ, then an edge with label σ is taken. In location ℓ, if the controller lets time pass, then an arbitrary positive amount of time $\delta \in \mathbb{R}_{>0}$ is selected so that the invariant of ℓ is not violated. As the actual amount of time that will pass is not decided by the controller, the model is time-abstract.

Definition 9 (Known switch conditions dense-time safety control). *The* known switch conditions dense-time safety control problem *for a class* $\mathcal{C}$ *of rectangular automata is the following: given a rectangular automaton* $H \in \mathcal{C}$ *and*

[2] The sampling semantics of [HK99] is more general in that it allows more than one discrete transition between two continuous transitions of duration β. All results presented here apply also to that case.

a rectangular state predicate fin *for* H*, solve the safety control problem for the time-abstract semantics* S_H^{abstr} *and state set* $[\![\mathsf{fin}]\!]$*; that is, determine if there exists a deadlock-free control map* κ *for* S_H^{abstr} *such that* $[\![\mathsf{fin}]\!]$ *is not reachable in the closed-loop system* $\kappa(S_H^{abstr})$*.* □

Unknown switch conditions. An unknown switch conditions dense-time controller controls a rectangular automaton by strengthening the invariants and guards that appear in the automaton. In particular, in location ℓ, such a controller may let time pass up to some upper bound, which can be stronger than the invariant of ℓ. In order to synthesize control maps, the constraints added by the controller cannot be arbitrary, but must be expressible in some language. Following [AMPS98], we consider constraints that can be expressed as polyhedral predicates. This is captured by the notion of a polyhedral control map.

Let $H = (L, X, \Sigma, \mathsf{init}, E, \mathsf{inv}, \mathsf{flow})$ be a rectangular automaton, and let κ be a control map for the dense-time semantics S_H^{dense}. For a state set $P \subseteq L \times [X \to \mathbb{R}]$ and an event $\sigma \in \Sigma$, define $Post_H^\sigma(P) = \{q' \mid \exists q \in P.\ q \xrightarrow{\sigma}_{dense} q'\}$ and $Post_{\kappa,H}^\sigma(P) = \{q' \mid \exists q \in P.\ q \xrightarrow{\sigma}_{dense} q' \text{ and } \sigma \in \kappa(q)\}$. Furthermore, define $Post_H^{time}(P) = \{q' \mid \exists q \in P,\ \delta \in \mathbb{R}_{>0}.\ q \xrightarrow{\delta}_{dense} q'\}$ and $Post_{\kappa,H}^{time}(P) = \{q' \mid \exists q \in P,\ \delta \in \mathbb{R}_{>0}.\ q \xrightarrow{\delta}_{dense} q' \text{ and } \delta \in \kappa(q)\}$. Note that if P is polyhedral, then $Post_H^\sigma(P)$ is a polyhedral state set of H for all events $\sigma \in \Sigma$, and $Post_H^{time}(P)$ is also a polyhedral state set of H [ACH+95]. A control map κ for the dense-time semantics S_H^{dense} is *polyhedral* if the following two conditions are satisfied. (1) For every event $\sigma \in \Sigma$, there exists a polyhedral state set P_σ of H such that for every polyhedral state set P of H, we have $Post_{\kappa,H}^\sigma(P) = Post_H^\sigma(P \cap P_\sigma)$. (2) There exists a polyhedral state set P_{time} of H such that for every polyhedral state set P of H, we have $Post_{\kappa,H}^{time}(P) = Post_H^{time}(P) \cap P_{time}$. Note that the polyhedral state set P_σ can be used to strengthen edge guards, while P_{time} can be used to strengthen location invariants.

Definition 10 (Unknown switch conditions dense-time safety control). *The* unknown switch conditions dense-time safety control problem *for a class* $\mathcal{C}$ *of rectangular automata is the following: given a rectangular automaton* $H \in \mathcal{C}$ *and a rectangular state predicate* fin *for* H*, determine if there exists a deadlock-free polyhedral control map* κ *for the dense-time semantics* S_H^{dense} *such that the state set* $[\![\mathsf{fin}]\!]$ *is not reachable in the closed-loop system* $\kappa(S_H^{dense})$*.* □

It is easy to see that if the answer to the known switch conditions dense-time control problem $\langle H, \mathsf{fin}\rangle$ is YES, then the answer to the unknown switch conditions dense-time control problem $\langle H, \mathsf{fin}\rangle$ is also YES. Indeed, if the answer to the known (resp. unknown) switch conditions dense-time control problem $\langle H, \mathsf{fin}\rangle$ is YES, then we can constructively strengthen the guards (resp. both invariants and guards) of H with a finite set of polyhedral predicates such that the resulting hybrid automaton $\kappa(H)$ (which may no longer be rectangular) is non-blocking and contains no reachable states in $[\![\mathsf{fin}]\!]$.

	KSC	USC	KSR	USR
Timed automata	$\surd$ [MPS95]	$\surd$ [MPS95]	$\surd$ [HW92]	?
Initialized rectangular automata	$\surd$ [HHM99]	$\times$ [HKPV98]	$\surd$ [HK99]	?
Rectangular automata	$\times$ [HHM99]	$\times$ [HKPV98]	$\surd$ [HK99]	?

Table 1. Decidability results for safety control problems. KSC stands for "known switch conditions dense-time"; USC for "unknown switch conditions dense-time"; KSR for "known sampling rate discrete-time"; USR for "unknown sampling rate discrete-time"; $\surd$ stands for decidable, $\times$ for undecidable, and ? for previously open. In the next section, it will be shown that all three previously open problems are undecidable (cf. Corollary 1).

3.2 Discrete-time control

In discrete-time control, the controller samples the plant state once every β time units, for some sampling unit $\beta \in \mathbb{Q}_{>0}$, and issues control actions at these points in time. Control actions, as before, may cause a change in location. Between control actions the plant evolves for β time units without interruption. We distinguish between discrete-time control with a known sampling unit β [HK99], and the more ambitious problem of synthesizing a suitable sampling unit, which has not been studied before.

Definition 11 (Known sampling rate discrete-time safety control). *The* known sampling rate discrete-time safety control problem *for a class $\mathcal{C}$ of rectangular automata is the following: given a rectangular automaton $H \in \mathcal{C}$, a sampling unit $\beta \in \mathbb{Q}_{>0}$, and a rectangular state predicate* fin *for H, solve the safety control problem for the sampling semantics $S_H^{sample}(\beta)$ and state set $[\![\mathsf{fin}]\!]$; that is, determine if there exists a deadlock-free control map κ for $S_H^{sample}(\beta)$ such that $[\![\mathsf{fin}]\!]$ is not reachable in the closed-loop system $\kappa(S_H^{sample}(\beta))$.* □

Definition 12 (Unknown sampling rate discrete-time safety control). *The* unknown sampling rate discrete-time safety control problem *for a class $\mathcal{C}$ of rectangular automata is the following: given a rectangular automaton $H \in \mathcal{C}$ and a rectangular state predicate* fin *for H, determine if there exist a sampling unit $\beta \in \mathbb{Q}_{>0}$ and a deadlock-free control map κ for the sampling semantics $S_H^{sample}(\beta)$ such that the state set $[\![\mathsf{fin}]\!]$ is not reachable in the closed-loop system $\kappa(S_H^{sample}(\beta))$.* □

3.3 Comparison between dense-time and discrete-time control

To compare the different classes of controllers we first recall what is known about the decidability of the various control problems defined above. The results from the literature are summarized in Table 1. For all decidable entries, suitable control maps can be synthesized algorithmically. Next we study two examples that shed some light on the relative merits of the different models.

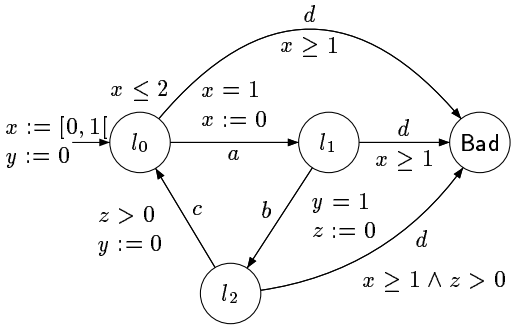

Fig. 1. A timed automaton that cannot be controlled by any sampling controller, no matter how small the sampling unit, but can be controlled by a dense-time controller (cf. Theorem 1).

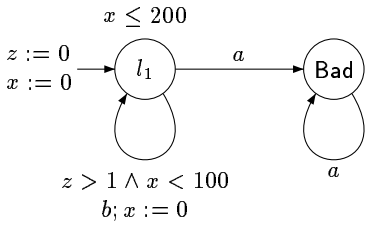

Fig. 2. A timed automaton that cannot be controlled by any time-abstract controller, but can be controlled by a sampling controller (cf. Theorem 2).

Unknown sampling rate discrete-time control is not as powerful as dense-time control. Our definitions of dense-time control ignore Zeno phenomena; for example, a controller that blocks the progress of time by causing control actions at convergent points in time, such as $0, \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$, is considered legal. Zeno phenomena have been studied extensively, and are usually avoided either by lifting the control objective from safety to liveness [Won97,dAHM01], or by assumption, as in [AMPS98], where it is imposed that every loop in the control structure of a hybrid automaton must contain a clock that is reset to 0 and then tested to be greater than 1 ("strongly non-zeno automata"). However, dense-time control models are more seriously flawed: we present a timed automaton H that can be controlled by a dense-time controller, but there is no sampling unit, however small, such that H can be controlled by a sampling controller. The dense-time controller issues control actions at divergent points in time, such as $0, \frac{1}{2}, 1, 1\frac{3}{4}, 2, 2\frac{7}{8}, 3, 3\frac{15}{16}, \ldots$, but the difference between adjacent time instants cannot be bounded from below.

Consider the timed automaton of Figure 1. The control objective is to avoid the location Bad. Thus, the controller must keep the automaton looping in locations l_0, l_1, and l_2. To do so, let us show that the controller must leave location l_2 faster and faster. The first time that the automaton enters l_0, the variables x and y have the values x_0 and y_0, respectively, where $x_0 < 1$ and $y_0 = 0$. To keep looping, the controller must ensure that $x \leq 1$ when reentering location l_0. We now describe the evolution of the value of x at the successive times location l_0 is entered: by x_i and y_i, we denote the values of x and y, respectively, when entering l_0 after i iterations of the loop. The loop consists in taking successively the edges labeled by a, b, and c. When crossing a, we have $x = 1$, and when entering l_1, we have $x = 0$ and y equals the amount of time δ_i^0 spent by the automaton in location l_0. The automaton stays in location l_1 for time $\delta_i^1 = 1 - \delta_i^0$, and then the edge labeled by b is crossed. The automaton stays in location l_2 for time $\delta_i^2 > 0$. So the value of x when reentering l_0 is $x_{i+1} = x_i + \delta_i^2$. Remember that the value

of x must be less or equal to 1 when entering location l_0 in order to avoid being forced to go to the Bad location. So we must have $\sum_{i=1}^{\infty} \delta_i^2 < 1 - x_0$. Such a converging sequence of δ_i^2 can be enforced by a dense-time controller with unknown switch conditions. This controller imposes the additional invariant $x < 1$ on l_2, and the d edges are labeled with the guard false, ensuring that they can never be taken. It is not difficult to see that this new automaton is non-blocking and cannot visit the Bad location. Note that our controller only takes three control actions per time unit, and so is not excluded by the definition of control given in [ABD+00], which requires the number of control actions per time unit to be bounded from above. As a direct consequence of the discussion above, the automaton of Figure 1 modified with the additional invariant $x < 1$ on location l_2 is controllable by a dense time controller with known switch conditions.

Now let us establish that there does not exist a sampling controller that can avoid the location Bad, no matter how small the sampling unit. If we fix the sampling unit to be $\beta \in \mathbb{Q}_{>0}$, then the automaton reaches for the first time l_2 with $x_0 = \beta$ and $y_0 = 0$. Then, after each iteration of the loop, when entering l_0 the value of x has grown by β (the time spent in location l_2). Thus, after n iterations of the loop, for $n = \min_j(j \cdot \beta > 1 - \beta)$, the value of x when entering l_0 is strictly greater than 1. Thus only the d edge leaving l_0 can be taken, leading to the Bad location. Finally, let us note that this justification is still valid when considering the automaton of Figure 1 modified with the additional invariant $x < 1$ on location l_2.

Theorem 1. *There exist a timed automaton H and a rectangular state predicate* fin *such that the answer to the known switch conditions dense-time safety control problem $\langle H, \mathsf{fin} \rangle$ is* YES*, and the answer to the unknown sampling rate discrete-time safety control problem $\langle H, \mathsf{fin} \rangle$ is* NO. □

Known switch conditions dense-time control is not as powerful as discrete-time control. Let us now consider the timed automaton of Figure 2. The control objective is again to avoid the location Bad. The automaton is not known switch conditions dense-time controllable. In fact, when entering for the first time l_1, we have $z = 0$, and the controller can decide either to take the edge labeled by a, or to let time pass. The decision to take the edge labeled by a is excluded, as this would lead directly to the Bad location. Unfortunately, the controller cannot allow time to pass either. In fact, if the controller chooses to let time pass, then it agrees to wait for an undetermined amount of time, including delays $100 \leq \delta \leq 200$. If $\delta = 200$, then the automaton reaches a state where only a is enabled, which leads to the Bad location. This means that this simple system cannot be controlled by a dense-time controller with known switch conditions. On the other hand, the automaton is controllable by an unknown switch conditions dense-time controller. A simple way to control the automaton is to add the invariant $x < 100$ to location l_1, and to add the guard false to the edge labeled by a. The automaton can also be controlled by a discrete-time controller with a sampling unit $\beta < 100$.

Theorem 2. *There exist a timed automaton H, a sampling unit β, and a rectangular state predicate* fin *such that the answer to the known sampling rate discrete-time safety control problem $\langle H, \beta, \mathsf{fin}\rangle$ is* YES*, and the answer to the known switch conditions dense-time safety control problem $\langle H, \mathsf{fin}\rangle$ is* NO. □

4 Unknown Sampling Rate Control is Undecidable

We establish the undecidablity of the unknown sampling rate discrete-time safety control problem for the restricted class of timed automata. Actually, we prove the undecidability of a weaker question, namely, the unknown-rate discrete-time reachability problem.

Definition 13 (Unknown-rate discrete-time reachability). *The* unknown-rate discrete-time reachability problem *for a class $\mathcal{C}$ of rectangular automata is the following: given a rectangular automaton $H \in \mathcal{C}$ and a rectangular state predicate* fin *for H, determine if there exists a sampling unit $\beta \in \mathbb{R}_{>0}$ such that the state set $[\![\mathsf{fin}]\!]$ is not reachable in the sampling semantics $S_H^{sample}(\beta)$.* □

Our main result is the following.

Theorem 3. *The unknown-rate discrete-time reachability problem is undecidable for timed automata.* □

As non-reachability can be reduced to safety control, we obtain the following.

Corollary 1. *The unknown sampling rate discrete-time safety control problem is undecidable for timed automata.*

To establish Theorem 3, we reduce the control state reachability problem for two-counter machines to the unknown-rate discrete-time reachability problem for timed automata. The former is well-known to be undecidable, thus implying the undecidability of the latter. A two-counter machine M consists of two counters (initially 0) and a finite sequence of instructions, each taken from the following: increment a counter; test a counter for zero and branch conditionally; decrement a counter if its value is not zero. Given M and a control state u of M, we construct a timed automaton H_M and a location l_u of H_M such that the execution of M reaches u iff there exists a sampling unit $\beta \in \mathbb{Q}_{>0}$ such that a state with discrete part l_u is reachable in the sampling semantics $S_{H_M}^{sample}(\beta)$. The key property of H_M is that if the automaton is sampled every $\frac{1}{b}$ time units, for some $b \in \mathbb{Q}_{>0}$, then it can simulate the execution of M as long as the counter values are less than $\lfloor b \rfloor$. When a counter overflow occurs, then H_M goes into a terminal location different from l_u. If the execution of M reaches u, then it does so with some maximal counter values, and H_M can reach l_u for a sufficiently large choice of b. On the other hand, if the execution of M does not reach u, then H_M cannot reach l_u for any choice of b.

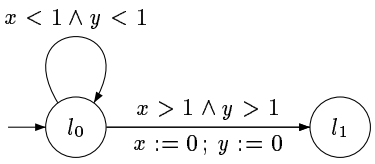

Fig. 3. Widget for zero testing

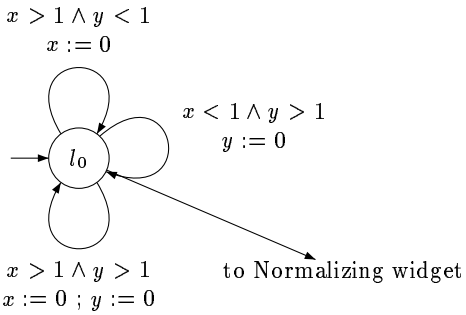

Fig. 4. Idling widget

Configuration encoding. First we need to decide how to encode the configurations of the counter machine M. A configuration of M consists of a control state and two counter values $c_1, c_2 \in \mathbb{N}$. The control state w of M is directly encoded by a location l_w of H_M. A counter value c is encoded by the difference between two clocks x and y. Recall that H_M is sampled every $\frac{1}{b}$ time units. If clocks are always reset to 0, then the clock difference $x - y$ is a multiple of $\frac{1}{b}$ at every state in every run of the sampling semantics $S^{sample}_{H_M}(\frac{1}{b})$. Moreover, we constrain the values of x and y to lie in the interval $I_b = [0, \frac{\lfloor b+1 \rfloor}{b}]$; then $|x-y| \in I_b$. We encode counter value c using only clock values $x, y \in I_b$, and impose the following on the automaton H_M:

- If $x \geq y$, then $c = (x - y) \cdot b$.
- If $x < y$, then $c = \left(\frac{\lfloor b+1 \rfloor}{b} - (y - x)\right) \cdot b$.

In this way the maximal counter value we can encode with $x, y \in I_b$ is exactly $\lfloor b \rfloor$. Note that when $x < y$, we cannot encode the counter value 0. Thus, $c = 0$ is always encoded with $x \geq y$.

We now give widgets for encoding the three types of instructions of the counter machine, and a widget for what we call "normalization" of the encoding. As we want our result to be as general as possible, and *robust* in the sense of [HR00], we avoid the use of equality in the rectangular predicates of H_M (the proof, especially the widget for zero testing, would be somewhat easier if the use of equality were allowed).

Zero testing. We assume two clocks x and y encoding counter value c with $x \geq y$. A widget for zero testing without using equality is shown in Figure 3. More formally, there exists $\frac{1}{b} \in \mathbb{Q}_{>0} \setminus \frac{1}{\mathbb{N}}$ such that the state $\langle l_1, x = 0, y = 0 \rangle$ is reachable from $\langle l_0, x, y \rangle$ in the sampling semantics $S^{sample}_{H_M}(\frac{1}{b})$ iff $x = y$ in l_0. This gives us a way to test if $c = 0$ when its value is encoded by $(x - y) \cdot b$. As we will see in the next paragraph, we can restrict ourselves to zero testing when $x \geq y$. In the sequel we assume $\frac{1}{b} \in \mathbb{Q}_{>0} \setminus \frac{1}{\mathbb{N}}$, so that value 1 is never hit for both x and y at any sampling point of any run.

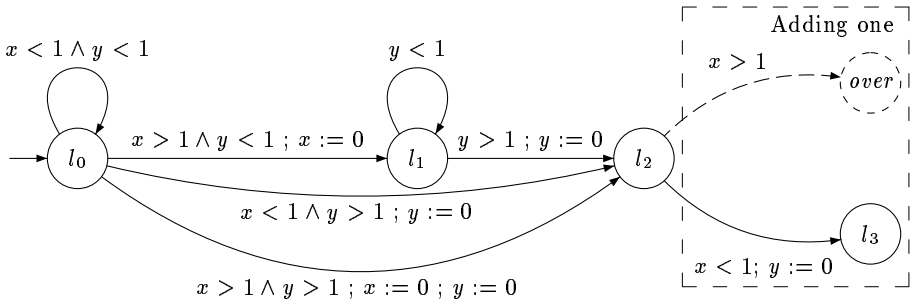

Fig. 5. Widget for normalizing (and incrementing) the encoding of a counter value

Normalizing the encoding. Before addressing the encoding of incrementing and decrementing a counter, we give a widget for obtaining a normal form for the encoding of counter value c. By *normal form* we mean that $c = (x-y) \cdot b$ can be written in a unique manner with $y = 0$ and $x \geq y$. The widget for normalization is shown in Figure 5 (the dashed box contains extra material that will be dealt with later). If either $x \geq y$ and $c = (x-y) \cdot b$, or $x < y$ and $c = \left(\frac{\lfloor b+1 \rfloor}{b} - (y-x)\right) \cdot b$, then the counter value c is in normal form when entering location l_2, that is, $y = 0$ and $x \geq y$.

Incrementing a counter. To add one to counter value c encoded by clocks x and y, we first normalize the encoding of c using the widget of Figure 5. To increment c we need to let time pass until the next edge transition, that is, we need to wait $\frac{1}{b}$ time units in location l_2. Thus we add a single edge from l_2 to l_3 resetting y; see Figure 5 inside the dashed box.

Decrementing a counter. To decrement a counter value we use a slightly modified version of the widget for normalization, namely, an anti-normal form: encode counter value c with $x = 0$ and $y > 0$, and after $\frac{1}{b}$ time units reset x.

Counter overflow and idling. Counter overflows are dealt with when adding one to a counter. The dashed transition to location *over* in Figure 5 corresponds to a counter overflow. Finally, we have to manage concurrently two counters C_1 and C_2. While C_1 is updated, time elapses, and we need to keep the value of C_2. This is done by using an *idling* widget, shown in Figure 4, for the counter that is not in the instruction to be performed. An instruction of the counter machine involving C_1 is encoded by the synchronous composition of the appropriate widget for C_1 and the idling widget for C_2. This completes the proof sketch for Theorem 3.

References

[ABD+00] E. Asarin, O. Bournier, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. In *Proc. IEEE*, 88:1011–1025, 2000.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AMP95] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems II*, LNCS 999, pp. 1–20. Springer, 1995.

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *IFAC Symp. System Structure and Control*, pp. 469–474. Elsevier, 1998.

[dAHM01] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR: Concurrency Theory*, LNCS 2154, pp. 536–550. Springer, 2001.

[HHM99] T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *CONCUR: Concurrency Theory*, LNCS 1664, pp. 320–335. Springer, 1999.

[HK99] T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.

[HKPV98] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *J. Computer and System Sciences*, 57:94–124, 1998.

[HR00] T.A. Henzinger and J.-F. Raskin. Robust undecidability of real-time and hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1790, pp. 145–159. Springer, 2000.

[HW92] G. Hoffmann and H. Wong-Toi. The input-output control of real-time discrete-event systems. In *RTSS: Real-time Systems Symp.*, pp. 256–265. IEEE, 1992.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS: Theoretical Aspects of Computer Science*, LNCS 900, pp. 229–242. Springer, 1995.

[TLS00] C. Tomlin, J. Lygeros, and S. Sastry. A game-theoretic approach to controller design for hybrid systems. In *Proc. IEEE*, 88:949–970, 2000.

[Won97] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *CDC: Conf. Decision and Control*, pp. 4607–4612. IEEE, 1997.