LINEAR TIME LOGIC CONTROL OF DISCRETE-TIME LINEAR SYSTEMS

PAULO TABUADA AND GEORGE J. PAPPAS

ABSTRACT. The control of complex systems poses new challenges that fall beyond the traditional methods of control theory. One of these challenges is given by the need to control, coordinate and synchronize the operation of several interacting submodules within a system. The desired objectives are no longer captured by usual control specifications such as stabilization or output regulation. Instead, we consider specifications given by Linear Temporal Logic (LTL) formulas. We show that existence of controllers for discrete-time controllable linear systems and LTL specifications can be decided and that such controllers can be effectively computed. The closed-loop system is of hybrid nature, combining the original continuous dynamics with the automatically synthesized switching logic required to enforce the specification.

1. INTRODUCTION

1.1. **Motivation.** In recent years there has been an increasing interest in extending the application domain of systems and control theory from monolithic continuous plants to complex systems consisting of several concurrently interacting submodules. Examples range from multi-modal software control systems in the aerospace [DS97, LL00, TH04] and automotive industry [CJG02, SH05] to advanced robotic systems [KBr04, MSP05, DCG$^+$05]. This change in perspective is accompanied by a *shift in control objectives*. One is no longer interested in the stabilization or output regulation of individual continuous plants, but rather wishes to regulate the global system behavior[1] through the local control of each individual submodule or component. Typical specifications for this class of control problems include coordination and synchronization of individual modules, sequencing of tasks, reconfigurability and adaptability of components, etc. In order to address this emerging class of control problems we need to formally specify the desired system behavior:

How can we formally and succinctly specify the desired behavior of concurrently interacting systems?

The specification mechanism should also lead to controller design methods. These controllers will enforce a hybrid behavior on the controlled system since system evolution is influenced both by the continuous dynamics and by the discrete interaction (communication) between submodules. While many ad-hoc approaches have been reported in the literature for the design of such hybrid systems, any formal guarantee of operation can only be obtained through formal verification which is noticeably a hard[2] problem [HKPV98]. This suggests that one should aim at design methods that satisfy the specification by construction:

How can we design (hybrid) controllers ensuring satisfaction of specifications by construction, thereby avoiding or substantially reducing the need for formal verification?

Another dimension of this problem, that should not be neglected, is its computational aspect. As the number of modules increases, the possibilities of interaction between modules also increase thus rendering analysis of global behavior an extremely difficult task. This intrinsic complexity of concurrently interacting systems can only be addressed by *computational* synthesis methods, reducing error-prone human analysis or synthesis to the strictly necessary minimum. Only fully automated methods have the potential to scale and successfully address control problems for systems consisting of large numbers of interacting components:

The first author was partially supported by the National Science Foundation CAREER award 0446716 while the second author was partially supported by the National Science Foundation EHS 0311123 award.

[1]We use the term behavior in a rather informal way to refer to the set of system's trajectories that can be observed.

[2]Verification of simple specifications such as reachability is unfortunately undecidable for a large majority of systems of interest.

How can we render the design of controllers completely automated, from specification to implementation?

Motivated by the above described problems, we present in this paper an approach for the control of linear systems with objectives expressed in Linear Temporal Logic (LTL). There are two main reasons to describe control objectives in temporal logic. Firstly, temporal logic provides a formal specification mechanism allowing one to quantitatively define the desired behavior of a systems by prescribing the interaction between submodules. Secondly, temporal logic makes it possible to succinctly express complex objectives due to its similarity to natural language. In particular, temporal logic is well suited to express the novel class of specifications required by the control of concurrently interacting systems. These two reasons also justify the successful use of temporal logic as a specification language in the concurrency and computer aided verification communities [MP92, CPG99, McM93]. In the next section we show through simple examples how control specifications can be easily expressed in LTL.

The approach presented in this paper is also an important contribution towards the synthesis of correct by design systems. Temporal logic enables the use of powerful automata theoretic techniques lying at the heart of computational algorithms for control design. Transferring control design from the continuous to the finite world of automata (or transition systems) is in fact one of the major contributions of this paper. This transfer is also accompanied by the relevant refinement techniques allowing the transformation of finite automata models of the closed-loop dynamics into hybrid models where software controllers supervise continuous plants. In addition to presenting a fully automated design method, the resulting closed-loop systems satisfy the LTL specifications by construction, therefore resulting in correct designs for which no further validation or verification is necessary.

1.2. **Problem formulation.** Temporal logic allows one to succinctly describe many interesting temporal properties of systems. LTL formulas are built from predicates through the usual propositional connectives ($\vee, \wedge, \Rightarrow, \neg$) and two temporal operators: $\circ$ and $\mathbf{U}$. Informally, $\circ$ is read as "next" and a LTL formula $\circ\varphi$ is satisfied when formula φ is satisfied at the next time instant. The operator $\mathbf{U}$ is read as "until" and formula $\phi\,\mathbf{U}\varphi$ is satisfied when formula ϕ is satisfied until formula φ is satisfied. From the "until" operator, two commonly used operators can be defined: $\Box$ and $\diamond$. The first is read as "always", requiring that φ holds for all future time in order for $\Box\varphi$ to be satisfied. The operator $\diamond$ is read as "eventually" and $\diamond\varphi$ requires φ to hold at some time in the future. This set of operators permits the construction of formulas expressing many interesting control specifications which we now illustrate by simple examples.

Periodic synchronization: Consider two mobile robots performing a collaborative task. Each robot is sensing different information that should be shared with the other robot at least every three units of time. We consider robots described by discrete-time linear control systems:

$$x_1(t+1) = A_1x_1(t) + B_1u_1(t) \qquad x_2(t+1) = A_2x_2(t) + B_2u_2(t)$$

Vector $x_1 \in \mathbb{R}^2$ models the position of robot 1 while vector $x_2 \in \mathbb{R}^2$ models the position of robot 2. We model the exchange of information between the robots by the requirement that inter robot distance is reduced to less than $\delta > 0$ for communication to occur. This distance constraint is captured by the predicate:

$$\mathit{communicate} := d(x_1, x_2) \leq \delta$$

for some metric d. The desired inter robot communication specification can now be modeled in LTL as:

$$\varphi = \Box \diamond_3 \mathit{communicate}$$

where $\diamond_3$ is an abbreviation for "eventually within 3 units of time" and is defined by $\diamond_3 p = p \vee \circ p \vee \circ\circ p \vee \circ\circ\circ p$. Satisfaction of formula φ requires that at each time step $\diamond_3\mathit{communicate}$ holds, that is, communication will always occur within the next 3 time units.

Path planning and obstacle avoidance: Consider now a robot navigating in a environment cluttered with obstacles. Let Obs_i be a predicate modeling the location of obstacle $i \in I$ and let $Goal$ be a predicate modeling the destination location. Requiring the robot to reach the destination while avoiding the obstacles can be captured by $\diamond\, Goal \wedge \Box\neg\big(\bigvee_{i\in I} Obs_i\big)$.

Fault tolerance and recovery: Fault tolerance and recovery can also be specified in LTL. Let φ_1 be a LTL formula specifying the normal operation of the system, φ_2 a LTL formula describing the occurrence of a fault and φ_3 a LTL formula prescribing the desired fault recovery procedure. The formula $\Box\big(\varphi_1 \vee \big(\varphi_1 \mathbf{U}\,(\varphi_2 \wedge \circ\varphi_3)\big)\big)$ states that the system should always operate correctly (φ_1) or it should operate correctly until $\varphi_2 \wedge \circ\varphi_3$ holds. If this last formula is true, then the fault described by φ_2 occurs at some time t and is followed by the fault recovery procedure, defined by φ_3, at time $t+1$.

The previous examples represent only a small fraction of the interesting properties that can be specified through the use of LTL. The goal of this paper, synthesizing controllers enforcing LTL specifications, can thus be described as follows:

Problem 1.1. *Let Σ be a discrete-time linear control system and φ a LTL formula describing the desired behavior for Σ. Design a controller for Σ such that the closed-loop system satisfies φ.*

The solution to the above problem will require an interesting combination of computer science and control theoretic concepts and methods briefly described in the next section.

1.3. **Approach and main contributions.** The synthesis of controllers enforcing LTL specifications relies on the possibility of extracting finite models from continuous control systems. These finite abstractions will be equivalent (in a precise sense to be defined) to the continuous models therefore enabling the solution of control problems posed for continuous linear systems through discrete algorithmic techniques. Resulting discrete models for the closed-loop system are then refined, resulting in controllers for the original continuous system whose hybrid closed-loop behavior will satisfy the desired specification. The overall approach is pictured in Figure 1 and organized as follows.

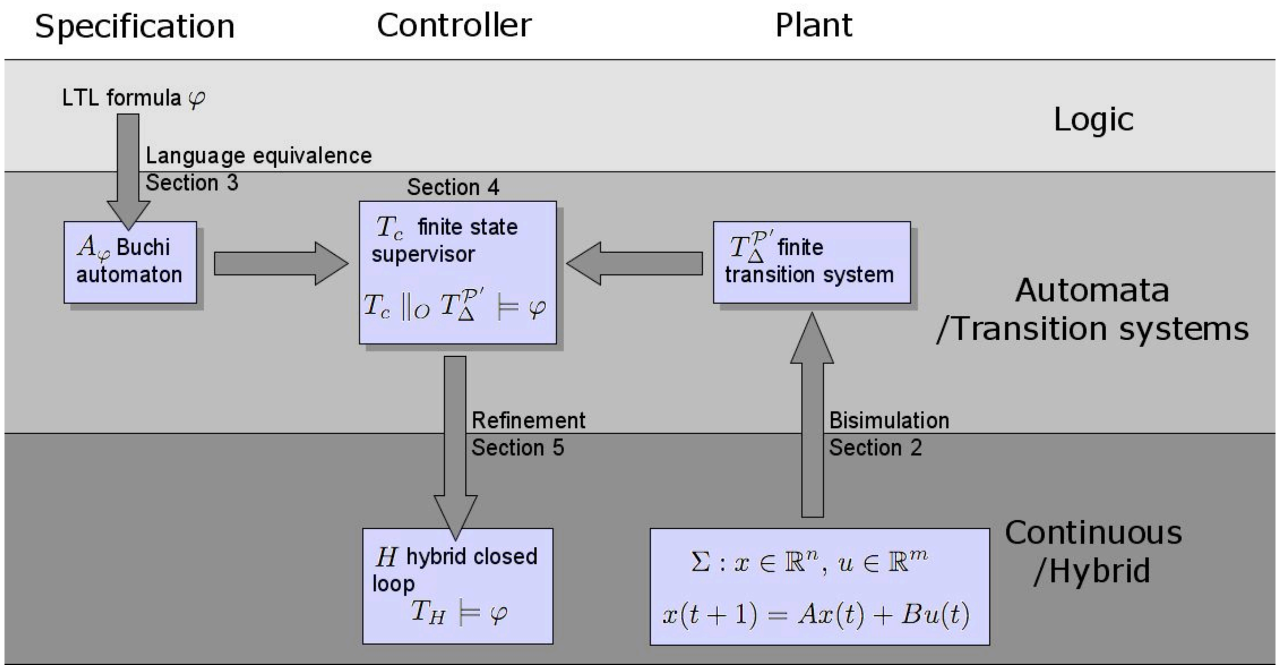

Specification
Controller
Plant
LTL formula φ
Logic
Language equivalence
Section 3
Section 4
A_φ Buchi automaton
T_c finite state supervisor
$T_c \parallel_O T_\Delta^{\mathcal{P}'} \models \varphi$
$T_\Delta^{\mathcal{P}'}$ finite transition system
Automata /Transition systems
Refinement
Section 5
Bisimulation
Section 2
H hybrid closed loop
$T_H \models \varphi$
$\Sigma : x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$
$x(t+1) = Ax(t) + Bu(t)$
Continuous /Hybrid

FIGURE 1. Intermediate steps in controller design.

In Section 2 we present one of the paper's main contribution. We show that any discrete-time controllable linear system admits finite abstractions (bisimulations) with respect to a certain class of observation functions defined by the system dynamics. The existence of such finite abstractions is one of the essential factors enabling the development of algorithms for system analysis and design. In this paper we will use finite abstractions of linear control systems to algorithmically synthesize controllers for LTL specifications. This will be done by constructing a finite supervisor for the discrete abstraction enforcing the LTL specification. Since supervisory synthesis is based on operational models such as finite state machines, Büchi automata or Petri-Nets, in Section 3 we introduce LTL and discuss the conversion of LTL formulas into Büchi automata. Supervisory synthesis is the subject of Section 4 where it is recalled that existence of finite supervisors enforcing infinite languages defined by Büchi automata can be decided and that such supervisors can be effectively computed. In Section 5 we refine the closed-loop behavior obtained by composing the finite abstraction with the discrete supervisor obtaining a hybrid closed-loop behavior enforcing the specification. The other main contribution of the paper is the possibility to synthesize this hybrid controller in a completely automated way. This result is formulated and proved also in Section 5. We conclude the paper at Section 6 with a discussion of the presented results.

The proposed methodology makes extensive use of both control and computer science concepts, notions and results. Since many of these notions may be unfamiliar to some readers we have decided to focus on the main aspects of the approach, thereby leaving for a later opportunity a more careful discussion of the algorithmic complexity issues as well as the many existing techniques to reduce complexity. We are also not explicitly addressing the centralized/decentralized nature of the resulting controllers, which although important, is a very difficult problem as many important decentralization questions are undecidable [Tri04]. For the same reasons we have decided to omit large examples and illustrate the introduced notions and algorithms with small, yet pedagogical examples.

1.4. **Related literature.**

The analysis and synthesis of systems based on temporal logic specifications is by now current practice in the concurrency and computer aided verification communities [MP92, CPG99, McM93]. Although this approach was initially devised for purely discrete systems, the seminal work of Alur and Dill on timed automata [AD94] showed that certain classes of hybrid systems could also be addressed. Subsequent extensions lead to results for multi-rate automata [ACH$^+$95] and rectangular hybrid automata [PV94, HM00] which lies on the decidability boundary [HKPV98]. These results were based on the construction of finite abstractions on which algorithms with guaranteed termination can be used for analysis and synthesis. Different classes of dynamics for which finite abstractions exist were introduced in [LPS00] by combining tools from logic and linear dynamical systems. See also [AHLP00] for a survey of these methods. Nonlinear dynamics were considered in [Bro99] where bisimulations based on foliations transverse to the nonlinear flow were introduced. In [SKA01] invariants are also exploited for a supervisory control approach to the control of hybrid systems. A different kind of dynamics, simple planar differential inclusions, was considered in [ASY01] where it was shown that qualitative analysis of system trajectories is decidable by making use of unique topological properties of the plane. Different approaches based on approximation techniques to obtain finite abstractions include the work in [CK01] for verification and [CKN98] for synthesis of supervisor controllers. Recently, a different abstraction technique based on quantifier elimination was introduced in [TK02]. This methodology allows one to obtain a sequence of finer finite abstractions that are sufficient to verify reachability related questions.

From the different mentioned approaches only the work described in [Bro99] address the problem of constructing (exact) finite abstractions of control systems. For linear systems, controllability can be exploited to compute the foliations required by the method in [Bro99] leading to finite abstractions of the vector field obtained by fixing the control inputs. Although at the technical level we do not make use of foliations, our construction can be seen as providing a way of integrating in the same finite object the different abstractions of [Bro99] obtained for different control inputs. However, our construction considers discrete-time systems, while the results in [Bro99] were developed for continuous time.

The construction of finite abstractions is also related to the study of reachability of quantized systems [BMP02, PLPB02, CP01]. For quantized systems, the original continuous dynamics is unchanged, but the set of available inputs is restricted to a finite set. This approach also provides an abstraction of the original control system, that can be regarded as a subsystem of the original one. Our approach differs from quantization based reachability in that we do not restrict the set of available inputs. Nevertheless, both approaches emphasize the advantages of having finite representations. Other related work includes the study of stabilization of linear systems with quantized observations [EM01, BL00].

Synthesis of controllers from temporal logic specifications had already been advocated in [AM95] where the authors postulate a discrete abstraction for the walking mechanism to be controlled. In [MD01], temporal logic is used to motivate the development of the synthesis procedures as well as to prove several facts regarding the proposed algorithms. Different automated synthesis procedures is reported in [HvS01], where it shown that synthesis of reachability specifications for hybrid systems with linear dynamics on convex polytopes can be performed by simply working with the polytopes vertices. Closer to our approach is the work reported in [VSS$^+$01], where it is shown that under certain controllability assumptions the controlled invariance problem for linear systems is decidable. Although our decidability results are also based on a controllability assumption, the problems being addressed are fundamentally different. We refine a given partition of the state space until a bisimulation is obtained while in [VSS$^+$01] a set is refined until controlled invariance is achieved. The goal of the refinement algorithms is therefore distinct, although termination is ensured in both cases by controllability.

Other related work, based on supervisory control of discrete event systems [RW89, KG95, CL99], includes synthesis for CTL* specifications [JK01] and real-time logic [Ost89]. However, synthesis from temporal logic specifications in the computer science community can be traced back to [EC82, MW84]. More recent work includes controller synthesis for branching time specifications [MT02a], decentralized control [MT02b, CMT99], control of synchronous systems [dAHM00, dAHM01] and synthesis for several different problems in timed automata including game theoretic approaches [FTM02], scheduling [AM02], optimal control [BFH$^+$01, ATP01] and synthesis from external specifications [DM02]. Although many of these works provide valuable inspiration, the proposed synthesis methodologies are only applicable to purely discrete systems or systems modeled by timed automata.

2. Finite quotients of controllable linear systems

In this section we show that finite abstractions of controllable linear systems exist and are effectively computable. These results will make a fundamental use of several computer science notions that we now review.

2.1. **Transition systems and bisimulations.**

Given a function $f : A \to B$ and a set $C \subseteq A$, we shall use the notation $f(C)$ to denote the subset of B defined by $\cup_{c\in C}\{f(c)\}$ while $f^{-1}(D)$ denotes the set $\{a \in A \mid f(a) \in D\}$ for some $D \subseteq B$. A partition $\mathcal{P}$ of the set A is a collection of sets $\mathcal{P} = \{P_i\}_{i\in I}$ satisfying $\cup_{i\in I} P_i = A$ and $P_i \cap P_j \neq \varnothing$ for $i \neq j$. Each partition induces a projection map $\pi_{\mathcal{P}} : A \to \mathcal{P}$ sending each $a \in A$ to the unique set $\pi_{\mathcal{P}}(a) = P \in \mathcal{P}$ containing a. Conversely, every surjective map $\pi : A \to B$ defines a partition of A defined by the collection of sets $\{\pi^{-1}(b)\}_{b\in B}$. An equivalence relation $R \subseteq A \times A$ on a set A induces a partition $\mathcal{P} = \{P_i\}_{i\in I}$ defined by $a, b \in P_i$ iff $(a, b) \in R$. The elements P_i of the partition $\mathcal{P}$ are the equivalence classes of R. Conversely, given a partition $\mathcal{P}$ on A we can define an equivalence relation $R \subseteq A \times A$ having the elements of $\mathcal{P}$ as equivalence classes. For this reason we will interchangeably work with partitions or equivalence relations according to what will be more useful. We say that partition $\mathcal{P}'$ refines or that it is a refinement of partition $\mathcal{P}$ when for every $P' \in \mathcal{P}'$ there exists a $P \in \mathcal{P}$ such that $P' \subseteq P$. Given a refinement $\mathcal{P}'$ of a partition $\mathcal{P}$ we can define a projection map $\pi_{\mathcal{P}'\mathcal{P}} : \mathcal{P}' \to \mathcal{P}$ taking every $P' \in \mathcal{P}$ to the unique element $\pi_{\mathcal{P}'\mathcal{P}}(P') = P \in \mathcal{P}$ such that $P' \subseteq P$.

We recall some formal language notions. Given a set S we denote by S^* the set of all finite strings obtained by concatenating elements in S. An element of S^* is therefore given by $s_1 s_2 \ldots s_n$ with $s_i \in S$ for $i = 1, \ldots, n$. By S^ω we denote the set of all infinite strings obtained by concatenating elements in S. An element of S^ω is an infinite string $s_1 s_2 s_3 \ldots$ with $s_i \in S$, $i \in \mathbb{N}$. Given a string s belonging to S^* or S^ω we denote by $s(i)$ the ith element of s. The length of a string $s \in S^*$ is denoted by $|s|$. A subset of S^* is called a language while a subset of S^ω is called an ω-language.

We also review the notion of transition systems that will be extensively used as an abstract model for control and computation.

Definition 2.1. A *transition system with observations* is a tuple $T = (Q, Q^0, \longrightarrow, O, \Upsilon)$ where:

- Q is a (possibly infinite) set of states,
- $Q^0 \subseteq Q$ is a set of initial states,
- $\longrightarrow \subseteq Q \times Q$ is a transition relation,
- O is a (possibly infinite) set of observations,
- $\Upsilon : Q \to O$ is a map assigning to each $q \in Q$ an observation $\Upsilon(q) \in O$.

A string $s \in Q^* \cup Q^\omega$ is a run of T if $(s(i), s(i+1)) \in \longrightarrow$, $i = 1, \ldots, |s| - 1$ for $s \in Q^*$ or $i \in \mathbb{N}$ for $s \in Q^\omega$. A run of T is initialized when $s(1) \in Q^0$.

The introduced notion of transition system differs from other notions encountered in the literature in that observations are not associated with transitions but rather with states. These two models can easily be seen equivalent given the well known equivalence between Moore and Mealy machines [HU79]. The presented model is, however, more natural since observations of control systems depend on the states and this structure is inherited by the several transition systems used in this paper to capture the dynamics of control systems.

We say that T is finite when Q, O are finite, and infinite otherwise. We will usually denote by $q \longrightarrow q'$ a pair (q, q') belonging to $\longrightarrow$. As we will only consider transition systems with observations, we shall refer to them simply as transition systems. Since the observation map $\Upsilon : Q \to O$ extends to a unique map of strings $\Upsilon : Q^* \cup Q^\omega \to O^* \cup O^\omega$ defined by:

$$\Upsilon(s) = \Upsilon\big(s(1)\big)\Upsilon\big(s(2)\big)\Upsilon\big(s(3)\big)\ldots$$

we shall abuse notation and use the same symbol Υ for both the observation map as well as for its induced string map. Given a state $q \in Q$, we denote by $\text{Pre}(q)$ the set of states in Q that can reach q in one step, that is:

$$\text{Pre}(q) = \{q' \in Q \mid q' \longrightarrow q\}$$

We extend Pre to sets $Q' \subseteq Q$ in the usual way:

$$\text{Pre}(Q') = \bigcup_{q' \in Q'} \text{Pre}(q')$$

Discrete-time linear control systems can be naturally embedded in the class of transition systems. Given a discrete-time linear control system Σ:

$$x(t+1) = Ax(t) + Bu(t), \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad t \in \mathbb{N}$$

there is an associated transition system T_Σ:

$$T_\Sigma = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, O, \Upsilon)$$

with $\longrightarrow \subseteq \mathbb{R}^n \times \mathbb{R}^n$ defined by $x \longrightarrow x'$ iff there exists a $u \in \mathbb{R}^m$ such that $x' = Ax + Bu$. To complete the definition of T_Σ we must also provide an observation set O and observation map Υ. The nature of the observation space and map depend on the problem being solved and are left unspecified for now. The described embedding is control abstract since the input value required to perform transition $x \longrightarrow x'$ is not explicitly captured by the transition system. However, this information can be recovered from the pair (x, x') by solving $x' = Ax + Bu$ for the input u. Since transition systems capture both control systems and software systems, we can synthesize controllers consisting of continuous and discrete (components) within the same framework.

Transition systems define different types of languages:

Definition 2.2. Let $T = (Q, Q^0, \longrightarrow, O, \Upsilon)$ be a transition system. The *language generated* by T, denoted by $L(T)$, is defined as:

$$L(T) = \big\{r \in O^* \mid r = \Upsilon(s) \text{ for some finite initialized run } s \text{ of } T\big\}$$

The ω*-language generated* by T is similarly defined:

$$L_\omega(T) = \big\{r \in O^\omega \mid r = \Upsilon(s) \text{ for some infinite initialized run } s \text{ of T } \big\}$$

The structural notion of bisimulation relates properties of different transition systems.

Definition 2.3. Let $T_i = (Q_i, Q_i^0, \longrightarrow_i, O, \Upsilon_i)$ with $i = 1, 2$ be transition systems and $R \subseteq Q_1 \times Q_2$ a relation. Relation R is said to be a *bisimulation* relation between T_1 and T_2 if the following hold for any $(q_1, q_2) \in R$:

(1) $\Upsilon_1(q_1) = \Upsilon_2(q_2)$;
(2) $q_1 \in Q_1^0$ implies $q_2 \in Q_2^0$ and $q_2 \in Q_2^0$ implies $q_1 \in Q_1^0$;
(3) $q_1 \longrightarrow_1 q_1'$ implies the existence of $q_2' \in Q_2$ satisfying $q_2 \longrightarrow_2 q_2'$ and $(q_1', q_2') \in R$;
(4) $q_2 \longrightarrow_2 q_2'$ implies the existence of $q_1' \in Q_1$ satisfying $q_1 \longrightarrow_1 q_1'$ and $(q_1', q_2') \in R$.

We shall use the notation $T_1 \cong T_2$ to denote the existence of a bisimulation relation between T_1 and T_2. Bisimilar transition systems share many properties including generated languages.

Proposition 2.4 (Adapted from [Mil89])**.** *Let T_1 and T_2 be transition systems and assume that $T_1 \cong T_2$. Then, the following equalities hold:*

$$L(T_1) = L(T_2) \qquad L_\omega(T_1) = L_\omega(T_2) \tag{2.1}$$

In addition to preserve language equivalence, bisimulations also preserve properties expressible in several temporal logics such as LTL, CTL, CTL* or μ-calculus [BCG88].

In this paper we shall construct finite bisimulations which are of a special form.

Definition 2.5. The *quotient* of a transition system $T = (Q, Q^0, \longrightarrow, O, \Upsilon)$ with respect to an equivalence relation $R \subseteq Q \times Q$ is the transition system $T_{/R} = (Q_{/R}, Q^0_{/R}, \longrightarrow_{/R}, O, \Upsilon_{/R})$ defined by:

- $Q_{/R} = \{S \subseteq Q \mid S \text{ is an equivalence class of } R\}$;
- $Q^0_{/R} = \pi_R(Q^0)$;
- $S \longrightarrow_{/R} S'$ in $T_{/R}$ if there exists $q \in S$ and $q' \in S'$ such that $q \longrightarrow q'$ in T;
- $\Upsilon_{/R}(S) = \Upsilon(q)$ for some $q \in S$.

Note that $\Upsilon_{/R}$ is well defined since $(q_1, q_2) \in R$ implies $\Upsilon(q_1) = \Upsilon(q_2)$. Furthermore, if R is a bisimulation relation between T and T it follows that the graph of the projection $\pi_R : Q \to Q_{/R}$, defined by $\{(q, S) \in Q \times Q_R \mid S = \pi_R(q)\}$, is a bisimulation relation between T and $T_{/R}$. $T_{/R}$ is therefore called a bisimilar quotient of T with respect to R.

2.2. **Finite bisimulations of controllable linear systems.**

In this section we show how finite bisimulations of controllable linear systems can be obtained. We make the following assumptions:

A.I Control system Σ is controllable.

A.II The columns of matrix B are linearly independent.

Assumption **A.II** results in no loss of generality since we can always remove linearly dependent columns from matrix B without destroying essential properties of Σ. Assumption **A.I** is essential for the existence of finite bisimulations. It has several important consequences, the first of which being the following decomposition of the state space.

Proposition 2.6 ([Bru70, Kal72]). *Let Σ be a discrete-time linear control system satisfying Assumptions* **A.I** *and* **A.II**. *Then, there exists a sequence of positive integers $\mu_1, \mu_2, \ldots, \mu_m$, called controllability indices of Σ, such that:*

$$\begin{aligned} & \operatorname{span}\{b_1, \ldots, b_m, Ab_1, \ldots, Ab_m, A^2b_1, \ldots, A^2b_m, \ldots, A^{n-1}b_1, \ldots, A^{n-1}b_m\} \\ = \; & \operatorname{span}\{b_1, Ab_1, A^2b_1, \ldots, A^{\mu_1-1}b_1, b_2, Ab_2, A^2b_2, \ldots, A^{\mu_2-1}b_2, \ldots, b_m, Ab_m, A^2b_m, \ldots, A^{\mu_m-1}b_m\} \end{aligned}$$

and $A^{\mu_i}b_i$ is linearly dependent of the vectors $b_1, Ab_1, \ldots, b_i, Ab_i, \ldots, A^{\mu_i-1}b_i$.

Using the controllability indices we can introduce the subspace $V \cong \mathbb{R}^{n-m}$ of $\mathbb{R}^n$ defined by:

$$V = \operatorname{span}\{b_1, Ab_1, \ldots, A^{\mu_1-2}b_1, b_2, Ab_2, \ldots, A^{\mu_2-2}b_2, \ldots, b_m, Ab_m, \ldots, A^{\mu_m-2}b_m\} \tag{2.2}$$

Subspace V naturally induces an observation map Υ for Σ defined as the natural projection from the state space $\mathbb{R}^n$ to the quotient space $\mathbb{R}^n/V \cong \mathbb{R}^m$:

$$\Upsilon : \mathbb{R}^n \to \mathbb{R}^n/V \tag{2.3}$$

Observation map Υ takes a vector $x \in \mathbb{R}^n$ into its equivalence class in $\mathbb{R}^n/V$ which we can identify with a point $y \in \mathbb{R}^m$. Observation map Υ uniquely determines transition system T_Σ associated with Σ. We now state this fact for later use.

Definition 2.7. Let Σ be a discrete-time linear control system satisfying Assumptions **A.I** and **A.II**. Transition system T_Σ associated with Σ is defined by $T_\Sigma = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathbb{R}^m, \Upsilon)$ with $x \longrightarrow x'$ iff there exists $u \in \mathbb{R}^m$ satisfying $x' = Ax + Bu$ and Υ defined by (2.3).

This choice of observation map is crucial in proving the first major contribution of this paper.

Theorem 2.8. *Let Σ be a discrete-time linear control system satisfying Assumptions* **A.I** *and* **A.II**. *For any finite partition $\mathcal{P}$ of the observation space of T_Σ there exists a finite refinement $\mathcal{P}'$ of the state space partition $\Upsilon^{-1}(\mathcal{P})$ such that the quotients of $T_\Sigma^{\mathcal{P}} = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathcal{P}, \pi_{\mathcal{P}} \circ \Upsilon)$ and $T_\Sigma^{\mathcal{P}'} = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathcal{P}', \pi_{\mathcal{P}'})$ with respect to $\mathcal{P}'$ and denoted by $T_\Delta^{\mathcal{P}}$ and $T_\Delta^{\mathcal{P}'}$, respectively, are finite bisimilar quotients.*

In order to prove Theorem 2.8 we state and prove a preparatory result ensuring that existence of finite bisimulations is not destroyed by changes of coordinates or invertible feedback.

Proposition 2.9. *Let Σ be a discrete-time linear control system satisfying Assumptions* **A.I** *and* **A.II** *and let Σ' be the discrete-time linear control system obtained from Σ through an invertible linear change of coordinates $x \mapsto Hx$ and an invertible linear feedback $(x, u) \mapsto Fx + Gu$. For any finite partition $\mathcal{P}$ of the observation space of T_Σ, there exists a finite refinement $\mathcal{Q}$ of the state space partition $\Upsilon^{-1}(\mathcal{P})$ making the quotient of T_Σ with respect to $\mathcal{Q}$ a finite bisimilar quotient iff there exists a finite refinement $\mathcal{Q}'$ of the state space partition $\Upsilon'^{-1}(H(\mathcal{P}))$ making the quotient of $T_{\Sigma'}$ with respect to $\mathcal{Q}'$ a finite bisimilar quotient.*

Proof. Assume that $\mathcal{Q}$ exists and let $\mathcal{Q}'$ be the finite partition $H(\mathcal{Q})$ of the state space of $T_{\Sigma'}$ (note that $H(\mathcal{Q})$ is a partition since H is an invertible matrix). It is clear that $\mathcal{Q}'$ refines $\Upsilon'^{-1}(H(\mathcal{P}))$. To show that $\mathcal{Q}'$ is a bisimulation relation between $T_{\Sigma'}$ and $T_{\Sigma'}$ consider $(z_1, w_1) \in \mathcal{Q}'$ and assume that $z_1 \longrightarrow z_2$ in $T_{\Sigma'}$. By definition of $\mathcal{Q}'$ and invertibility of H, there exists $(x_1, y_1) \in \mathcal{Q}$ such that $(Hx_1, Hy_1) = (z_1, w_1)$ and $x_1 \longrightarrow x_2$ in T_Σ with $Hx_2 = z_2$. It then follows from the fact that $\mathcal{Q}$ is a bisimulation relation between T_Σ and T_Σ that $y_1 \longrightarrow y_2$ in T_Σ with $(x_2, y_2) \in \mathcal{Q}$. Let u be the input triggering the transition $y_1 \longrightarrow y_2$. Then, input $Fy_1 + Gu$ triggers a transition $Hy_1 = w_1 \longrightarrow w_2 = Hy_2$ in T'_Σ and $(z_2, w_2) \in \mathcal{Q}'$ since $(z_2, w_2) = (Hx_2, Hy_2)$ and $(x_2, y_2) \in \mathcal{Q}$. This proves condition (3) in Definition 2.3 and condition (4) is proved using the same argument. Condition (2) is trivially satisfied since the set of initial states is $\mathbb{R}^n$ and H is invertible while condition (1) follows from the equality $\Upsilon = \Upsilon' \circ H$. □

We now return to the proof of Theorem 2.8.

Proof of Theorem 2.8. In view of Proposition 2.9 we can assume, without loss of generality, that Σ is in Brunovsky normal form since any controllable linear system can be transformed into this form by a change of coordinates and an invertible feedback [Bru70, Kal72]. Recall that the Brunovsky normal form of a controllable linear system with controllability indices $\mu_1, \ldots, \mu_m$ is given by:

$$\begin{array}{cccc} x_1(t+1) = x_2(t) & x_{\mu_1+1}(t+1) = x_{\mu_1+2}(t) & \ldots & x_{\mu_1+\ldots+\mu_{m-1}+1}(t+1) = x_{\mu_1+\ldots+\mu_{m-1}+2}(t) \\ x_2(t+1) = x_3(t) & x_{\mu_1+2}(t+1) = x_{\mu_1+3}(t) & \ldots & x_{\mu_1+\ldots+\mu_{m-1}+2}(t+1) = x_{\mu_1+\ldots+\mu_{m-1}+3}(t) \\ \vdots & \vdots & & \vdots \\ x_{\mu_1}(t+1) = u_1(t) & x_{\mu_1+\mu_2}(t+1) = u_2(t) & \ldots & x_{\mu_1+\ldots+\mu_m}(t+1) = u_m(t) \end{array} \tag{2.4}$$

A simple computation shows that for a control system in Brunovsky normal form Υ is of the form:

$$\Upsilon = \phi \circ \pi \tag{2.5}$$

where $\pi : \mathbb{R}^n \to \mathbb{R}^m$ is the projection map $\pi(x) = (x_1, x_{\mu_1+1}, \ldots, x_{\mu_1+\ldots+\mu_{m-1}+1})$ and ϕ is an arbitrary linear isomorphism. We will now introduce some notation to simplify the proof. We will use $x(t+i, u(t), u(t+1), \ldots, u(t+i-1))$ to denote $x(t+i)$ and to emphasize that $x(t+i, u(t), u(t+1), \ldots, u(t+i-1))$ is obtained from $x(t)$ by applying the sequence of inputs $u(t), u(t+1), \ldots, u(t+i-1)$. We will also denote by μ the largest controllability index.

We now note that it follows from equality (2.5) that input $u_k(t+j)$ will not affect $\Upsilon\big(x(t+i, u(t), \ldots, u(t+i-1))\big)$ when $\mu_k > i-j$. In other words:

$$\mu_k > i-j \quad \Rightarrow \quad \frac{\partial}{\partial u_k(t+j)} \Upsilon\big(x(t+i), u(t), \ldots, u(t+i-1)\big) = 0 \tag{2.6}$$

To illustrate this remark consider a control system defined by $x_1(t+1) = x_2(t)$ and $x_2(t+1) = u(t)$ with observation map $\Upsilon(x) = x_1$. Since the controllability index corresponding to input u is 2 we see that for $j = 0$:

$$(i=0) \qquad \frac{\partial}{\partial u(t)} \Upsilon(x(t)) = \frac{\partial}{\partial u(t)} x_1(t) = 0$$

$$(i=1) \qquad \frac{\partial}{\partial u(t)} \Upsilon\big(x(t+1, u(t))\big) = \frac{\partial}{\partial u(t)} x_2(t) = 0$$

This remark will be used several times in the proof.

Consider now the equivalence relation $R^k \subseteq \mathbb{R}^n \times \mathbb{R}^n$ recursively defined as follows:

$$\begin{array}{rll} (x(t), y(t)) \in R^0 & \text{iff} & \pi_{\mathcal{P}} \circ \Upsilon(x(t)) = \pi_{\mathcal{P}} \circ \Upsilon(y(t)) \\ (x(t), y(t)) \in R^{k+1} & \text{iff} & \forall u(t) \in \mathbb{R}^m \, \exists v(t) \in \mathbb{R}^m \wedge \forall v(t) \in \mathbb{R}^m \, \exists u(t) \in \mathbb{R}^m \quad \big(x(t+1, u(t)), y(t+1, v(t))\big) \in R^k \end{array}$$

We claim that $R^{\mu-1}$ is an auto-bisimulation relation. Since condition (1) in Definition 2.3 follows from the chosen observation function for $T_\Sigma^{\mathcal{P}}$ and $T_\Sigma^{\mathcal{P}'}$ and condition (2) follows from the equality $Q_0 = \mathbb{R}^n$, in order to prove the claim we only need to show that for any $(x(t), y(t)) \in R^{\mu-1}$:

(1) if $x(t) \longrightarrow x(t+1, u(t))$, then there exists $y(t+1, v(t))$ satisfying $y(t) \longrightarrow y(t+1, v(t))$ and $\big(x(t+1, u(t)), y(t+1, v(t))\big) \in R^{\mu-1}$;
(2) if $y(t) \longrightarrow y(t+1, v(t))$, then there exists $x(t+1, u(t))$ satisfying $x(t) \longrightarrow x(t+1, u(t))$ and $\big(x(t+1, u(t)), y(t+1, v(t))\big) \in R^{\mu-1}$.

We shall show only (1) since the same argument is valid for (2). We will prove (1) by showing that $(x(t), y(t)) \in R^{\mu-1}$ implies $(x(t), y(t)) \in R^{\mu}$ since (1) would then follow from the definition of R^k.

Consider then $(x(t), y(t)) \in R^{\mu-1}$ and let $x(t) \longrightarrow x(t+1, u(t))$. It follows from $(x(t), y(t)) \in R^{\mu-1}$ and the definition of R^k, the existence of $y(t) \longrightarrow y(t+1, v(t))$ satisfying $\big(x(t+1, u(t)), y(t+1, v(t))\big) \in R^{\mu-2}$. Making use of (2.6) for $j = 0$ we see that if we modify $v(t)$ to $\overline{v}(t)$ by changing the components $v_k(t)$ of $v(t)$ such that $\mu - 1 < \mu_k$ we will still have $\big(x(t+1, u(t)), y(t+1, \overline{v}(t))\big) \in R^{\mu-2}$ since $R^{\mu-2}$ is based on the equalities

$\pi_\mathcal{P}\circ\Upsilon\big(x(t+i,u(t),\ldots,u(t+i-1))\big)=\pi_\mathcal{P}\circ\Upsilon\big(y(t+i,v(t),\ldots,v(t+i-1))\big)$ for $1\leq i\leq\mu-1$. We thus define $\overline{v}(t)$ as:

$$\begin{array}{lll}\overline{v}_k(t)=u_k(t) & \text{if} & \mu-1<\mu_k\\ \overline{v}_k(t)=v_k(t) & \text{if} & \mu-1\geq\mu_k\end{array}$$

At this point we have $\big(x(t+1,u(t)),y(t+1,\overline{v}(t))\big)\in R^{\mu-2}$ and we consider $x(t+1,u(t))\longrightarrow x(t+2,u(t),u(t+1))$. It follows from $\big(x(t+1,u(t)),y(t+1,\overline{v}(t))\big)\in R^{\mu-2}$ and the definition of R^k, the existence of $y(t+1,\overline{v}(t))\longrightarrow y(t+2,\overline{v}(t),v(t+1))$ satisfying $\big(x(t+2,u(t),u(t+1)),y(t+2,\overline{v}(t),v(t+1))\big)\in R^{\mu-3}$. Making use of (2.6) for $j=1$ we see that if we modify $v(t+1)$ to $\overline{v}(t+1)$ by changing the components $v_k(t+1)$ of $v(t+1)$ such that $\mu-2<\mu_k$ we will still have $\big(x(t+2,u(t),u(t+1)),y(t+2,\overline{v}(t),\overline{v}(t+1))\big)\in R^{\mu-3}$ since $R^{\mu-3}$ is based on the equalities $\pi_\mathcal{P}\circ\Upsilon\big(x(t+i,u(t),\ldots,u(t+i-1))\big)=\pi_\mathcal{P}\circ\Upsilon\big(y(t+i,v(t),\ldots,v(t+i-1))\big)$ for $2\leq i\leq\mu-1$. We thus define $\overline{v}(t+1)$ as:

$$\begin{array}{lll}\overline{v}_k(t+1)=u_k(t+1) & \text{if} & \mu-2<\mu_k\\ \overline{v}_k(t+1)=v_k(t+1) & \text{if} & \mu-2\geq\mu_k\end{array}$$

If we keep on modifying v to $\overline{v}$ according to the above described process we will obtain:

$$\big(x(t+\mu-1,u(t),\ldots,u(t+\mu-2)),y(t+\mu-1,v(t),\ldots,v(t+\mu-2))\big)\in R^0$$

Consider now $x(t+\mu-1,u(t),\ldots,u(t+\mu-2))\longrightarrow x(t+\mu,u(t),\ldots,u(t+\mu-1))$ and let $\overline{v}(t+\mu-1)$ be an arbitrary element of $\mathbb{R}^m$. It then follows that $y(t+\mu-1,\overline{v}(t),\ldots,\overline{v}(t+\mu-2))\longrightarrow y(t+\mu,\overline{v}(t),\ldots,\overline{v}(t+\mu-1))$ and furthermore:

$$\begin{array}{rcl}\pi(x(t+\mu,u(t),\ldots,u(t+\mu-1))) & = & (u_1(t+\mu-\mu_1),\ldots,u_m(t+\mu-\mu_m))\\ & = & (\overline{v}_1(t+\mu-\mu_1),\ldots,\overline{v}_m(t+\mu-\mu_m))\\ & = & \pi(y(t+\mu,\overline{v}(t),\ldots,\overline{v}(t+\mu-1))\end{array}$$

in virtue of the way we defined $\overline{v}$ for $0\leq i\leq\mu-2$. We thus conclude from (2.5) that:

$$\big(x(t+\mu,u(t),\ldots,u(t+\mu-1)),y(t+\mu,\overline{v}(t),\ldots,\overline{v}(t+\mu-1))\big)\in R^0$$

which in turn implies $(x(t),y(t))\in R^\mu$ and concludes the proof of the claim.

To conclude the proof of the theorem we must show that $R^{\mu-1}$ has a finite number of equivalence classes. However, this follows at once from the observation that $R^{\mu-1}$ represents bisimilarity restricted to $\mu-1$ steps and defined with respect to a finite observation space.

□

The possibility of synthesizing controllers enforcing LTL specifications hinges on Theorem 2.8 as it guarantees that the behavior of T_Σ admits a finite representation in the form of a bisimilar quotient. This bisimulation is based on a finite partition of the observation space of T_Σ used to describe the control objectives through a LTL formula. We thus see that the observation space and map of T_Σ, naturally defined by the system dynamics under Assumption **II**, are essential ingredients of Theorem 2.8.

Since existence of finite bisimulations has been established, the following well known bisimulation algorithm can be used to compute the coarsest possible bisimulation [BFH90, KS90] provided that every set operation is effectively computable[3]. The algorithm starts with a transition system T_Σ and the initial partition $\Upsilon^{-1}(\mathcal{P})$ of $\mathbb{R}^n$ and terminates with the coarsest partition $\mathcal{P}'$ such that $T_{/\mathcal{P}'}$ is a bisimilar quotient of T.

[3]We use the expression effectively computable to denote the existence of an algorithm for a Touring machine implementing the desired set operation.

Algorithm 2.10 (Bisimulation Algorithm)**.**

```
set: $\mathcal{P}' = \Upsilon^{-1}(\mathcal{P})$
while: $\exists P, P' \in \mathcal{P}'$ such that $\varnothing \neq P \cap \mathrm{Pre}(P') \neq P$
    set: $P_1 = P \cap \mathrm{Pre}(P')$, $P_2 = P \cap \overline{\mathrm{Pre}(P')}$
    refine: $\mathcal{P}' = \big(\mathcal{P}' \backslash \{P\}\big) \cup \{P_1, P_2\}$
end while
```

As we are considering linear control systems it is natural to consider partitions of the observation space $\mathbb{R}^m$ defined by semi-linear sets. To ensure computability we restrict all the coefficients to live in $\mathbb{Q}$:

Definition 2.11. The class of semi-linear subsets of $\mathbb{R}^m$ consists of finite unions, intersections and complements of the following elementary sets:

$$\big\{x \in \mathbb{R}^m \mid f^T x + c \sim 0, \quad f \in \mathbb{Q}^m, \quad c \in \mathbb{Q}, \quad \sim \in \{=, >\}\big\}$$

Computability of the finite bisimulation is now a consequence of effective computability of intersections, unions and complements of semi-linear sets and the fact that Pre of a semi-linear set can be computed by quantifier elimination [BCR98], resulting in a semi-linear set. We thus have the following corollary to Theorem 2.8:

Corollary 2.12. *Let Σ be a discrete-time linear control system satisfying Assumptions* **A.I** *and* **A.II** *and defined by matrices A and B with rational entries. For any finite partition $\mathcal{P}$ of the observation space of T_Σ defined by semi-linear sets, the quotients of $T_\Sigma^{\mathcal{P}} = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathcal{P}, \pi_{\mathcal{P}} \circ \Upsilon)$ and $T_\Sigma^{\mathcal{P}'} = (\mathbb{R}^n, \mathbb{R}^n, \longrightarrow, \mathcal{P}', \pi_{\mathcal{P}'})$ with respect to partition $\mathcal{P}'$, whose existence is asserted by Theorem 2.8, are finite bisimilar quotients which are effectively computable.*

Example 2.13. We now illustrate Theorem 2.8 on a variation of the periodic synchronization example discussed in the introduction. Consider two identical vehicles moving on the same lane as shown in Figure 2.

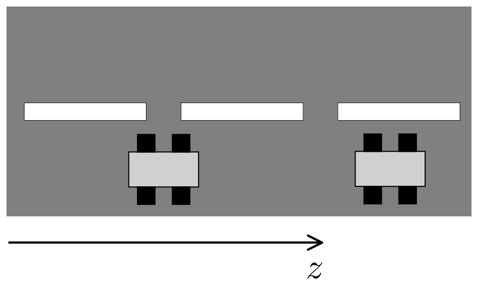


FIGURE 2. Identical vehicles following the same lane.

For our purposes it will be sufficient to consider the translational dynamics along the lane. Each vehicle is modeled as a discrete-time double integrator:

$$\begin{aligned} z_i(t+1) &= v_i(t) \\ v_i(t+1) &= u_i(t) \end{aligned}$$

where $i \in \{1, 2\}$ and z_i represents the translational position of car i. Since we will only be interested in controlling the spacing between the vehicles, we introduce new variables:

$$y_1 = z_2 - z_1 \qquad y_2 = v_2 - v_1 \qquad u = u_2 - u_1$$

leading to the following model:

$$\begin{aligned} y_1(t+1) &= y_2(t) \\ y_2(t+1) &= u(t) \end{aligned} \tag{2.7}$$

governing inter vehicle spacing measured by y_1. Observation map $\Upsilon : \mathbb{R}^2 \to \mathbb{R}$ satisfies $\Upsilon B = 0$ and if we model Υ by the row matrix $[a\ b]$ we obtain:

$$0 = \Upsilon B = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = b$$

To make the discussion concrete we take $a = 1$ leading to $\Upsilon y = y_1$ and note that other choices for a would equality work. The requirement that vehicles should come together (in order to communicate) at least every 3 seconds can be modeled by:

$$\begin{aligned} \varphi &= \Box \Diamond_3 \big(p_1 \wedge p_2\big) \\ p_1 &= y_1 + 1/2 > 0 \\ p_2 &= -y_1 + 1/2 > 0 \end{aligned} \tag{2.8}$$

Note that predicates p_1 and p_2 represent regions on the observation space $\mathbb{R} \cong \mathbb{R}^2/V = \mathbb{R}^2/\mathrm{span}\{B\}$. The formula $p_1 \wedge p_2$ is satisfied when the distance between vehicles is smaller than 1 and $\Box \Diamond_3 \big(p_1 \wedge p_2\big)$ ensures that such distance constraint is satisfied every 3 time steps.

If we denote by S_1 the set defined by $p_1 \wedge p_2$ and by S_2 its complement, that is:

$$S_1 = \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ -\frac{1}{2} < y_1 < \frac{1}{2}\Big\} \qquad S_2 = \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ y_1 \leq -\frac{1}{2} \vee y_1 \geq \frac{1}{2}\Big\}$$

we can use the bisimulation algorithm with the initial partition $\Pi = \{S_1, S_2\}$. Following Algorithm 2.10 we compute:

$$\begin{aligned} \mathrm{Pre}(S_1) &= \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ -\frac{1}{2} < y_2 < \frac{1}{2}\Big\} \\ S_2 \cap \mathrm{Pre}(S_1) &= \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ \Big(y_1 \leq -\frac{1}{2} \vee y_1 \geq \frac{1}{2}\Big) \wedge -\frac{1}{2} < y_2 < \frac{1}{2}\Big\} \end{aligned}$$

Since $S_2 \cap \mathrm{Pre}(S_1) \neq \varnothing$ and $S_2 \cap \mathrm{Pre}(S_1) \neq S_2$ we split S_2 into the sets S_{21} and S_{22} defined by:

$$\begin{aligned} S_{21} &= S_2 \cap \mathrm{Pre}(S_1) = \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ \Big(y_1 \leq -\frac{1}{2} \vee y_1 \geq \frac{1}{2}\Big) \wedge -\frac{1}{2} < y_2 < \frac{1}{2}\Big\} \\ S_{22} &= S_2 \cap \overline{\mathrm{Pre}(S_1)} = \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ \Big(y_1 \leq -\frac{1}{2} \vee y_1 \geq \frac{1}{2}\Big) \wedge \Big(y_2 \leq -\frac{1}{2} \vee y_2 \geq \frac{1}{2}\Big)\Big\} \end{aligned}$$

At this point the refined partition is given by $\Pi = \{S_1, S_{21}, S_{22}\}$. Choosing now S_1 and S_{21} from Π we compute:

$$\begin{aligned} \mathrm{Pre}(S_{21}) &= \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ y_2 \leq -\frac{1}{2} \vee y_2 \geq \frac{1}{2}\Big\} \\ S_1 \cap \mathrm{Pre}(S_{21}) &= \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ -\frac{1}{2} < y_1 < \frac{1}{2} \wedge \Big(y_2 \leq -\frac{1}{2} \vee y_2 \geq \frac{1}{2}\Big)\Big\} \end{aligned}$$

Again we verify that $S_1 \cap \mathrm{Pre}(S_{21}) \neq \varnothing$ and $S_1 \cap \mathrm{Pre}(S_{21}) \neq S_1$ which leads to the splitting of S_1 into S_{11} and S_{12} defined by:

$$\begin{aligned} S_{11} &= S_1 \cap \mathrm{Pre}(S_{21}) = \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ -\frac{1}{2} < y_1 < \frac{1}{2} \wedge \Big(y_2 \leq -\frac{1}{2} \vee y_2 \geq \frac{1}{2}\Big)\Big\} \\ S_{12} &= S_1 \cap \overline{\mathrm{Pre}(S_{21})} = \Big\{(y_1, y_2) \in \mathbb{R}^2 \ \Big|\ -\frac{1}{2} < y_1 < \frac{1}{2} \wedge \Big(-\frac{1}{2} < y_2 < \frac{1}{2}\Big)\Big\} \end{aligned}$$

This splitting leads to the new partition $\Pi = \{S_{11}, S_{12}, S_{21}, S_{22}\}$ which already defines a bisimulation since:

$$\begin{array}{llll} S_{11} \cap \mathrm{Pre}(S_{11}) = \varnothing & S_{12} \cap \mathrm{Pre}(S_{11}) = S_{12} & S_{21} \cap \mathrm{Pre}(S_{11}) = S_{21} & S_{22} \cap \mathrm{Pre}(S_{11}) = \varnothing \\ S_{11} \cap \mathrm{Pre}(S_{12}) = \varnothing & S_{12} \cap \mathrm{Pre}(S_{12}) = S_{12} & S_{21} \cap \mathrm{Pre}(S_{12}) = S_{21} & S_{22} \cap \mathrm{Pre}(S_{12}) = \varnothing \\ S_{11} \cap \mathrm{Pre}(S_{21}) = S_{11} & S_{12} \cap \mathrm{Pre}(S_{21}) = \varnothing & S_{21} \cap \mathrm{Pre}(S_{21}) = \varnothing & S_{22} \cap \mathrm{Pre}(S_{21}) = S_{21} \\ S_{11} \cap \mathrm{Pre}(S_{22}) = S_{11} & S_{12} \cap \mathrm{Pre}(S_{22}) = \varnothing & S_{21} \cap \mathrm{Pre}(S_{22}) = \varnothing & S_{22} \cap \mathrm{Pre}(S_{22}) = S_{21} \end{array}$$

This finite bisimulation has four discrete states and is graphically represented in Figure 3 where initial states are grey colored and observations are represented outside the circles denoting the states.

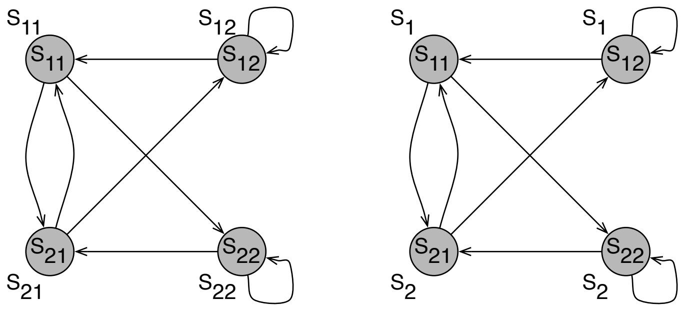


FIGURE 3. Finite bisimilar quotient $T_{\Delta}^{\mathcal{P}'}$ on the left and $T_{\Delta}^{\mathcal{P}}$ on the right.

Example 2.14. Consider now a linear control system described by the following matrices:

$$A = \begin{bmatrix} 2 & 0 & -1 \\ -1 & -7 & 11 \\ 0 & 4 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

It is not difficult to see that controllability holds and the controllability indices are given by $\mu_1 = 2$ and $\mu_2 = 1$. Vector space V is therefore spanned by the first column of B. Adopting the following matrix representation for the observation map $\Upsilon : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\Upsilon x = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

it follows that equality $\Upsilon b_1 = 0$ leads to $a_1 + a_2 + a_3 = 0$ and $b_1 + b_2 + b_3 = 0$. One possible choice for map Υ satisfying these equations is:

$$\Upsilon x = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Starting with the following observation space partition:

$$S_1 = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 > 0 \wedge y_2 > 0\} \qquad S_2 = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 > 0 \wedge y_2 \leq 0\}$$
$$S_3 = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 \leq 0 \wedge y_2 > 0\} \qquad S_4 = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 \leq 0 \wedge y_2 \leq 0\}$$

we obtain a finite bisimulation with 8 states defined by $T_{\Delta}^{\mathcal{P}} = (Q, Q^0, \longrightarrow, O, \Upsilon_{\Delta})$ where:

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\} \qquad Q^0 = Q \qquad O = \big\{S_1, S_2, S_3, S_4\big\}$$

$$\begin{array}{rcl} \longrightarrow & = & \big\{(q_1, q_1), (q_1, q_2), (q_1, q_3), (q_1, q_4), (q_3, q_1), (q_3, q_2), (q_3, q_3), (q_3, q_4)\big\} \\ & \cup & \big\{(q_2, q_5), (q_2, q_6), (q_2, q_7), (q_2, q_8), (q_4, q_5), (q_4, q_6), (q_4, q_7), (q_4, q_8)\big\} \\ & \cup & \big\{(q_5, q_1), (q_5, q_2), (q_5, q_3), (q_5, q_4), (q_7, q_1), (q_7, q_2), (q_7, q_3), (q_7, q_4)\big\} \\ & \cup & \big\{(q_6, q_5), (q_6, q_6), (q_6, q_7), (q_6, q_8), (q_8, q_5), (q_8, q_6), (q_8, q_7), (q_8, q_8)\big\} \end{array}$$

$$\Upsilon_\Delta(q_1) = S_1 \quad \Upsilon_\Delta(q_2) = S_1 \quad \Upsilon_\Delta(q_3) = S_2 \quad \Upsilon_\Delta(q_4) = S_2$$
$$\Upsilon_\Delta(q_5) = S_3 \quad \Upsilon_\Delta(q_6) = S_3 \quad \Upsilon_\Delta(q_7) = S_4 \quad \Upsilon_\Delta(q_8) = S_4$$

The sets associated with each discrete state were computed using Algorithm 2.10 and are given by:

$$\begin{array}{ll}
q_1: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 > 0 \wedge -x_1 - 11x_2 + 5x_3 > 0 \wedge x_1 + x_2 - 2x_3 > 0\big\} \\
q_2: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 > 0 \wedge -x_1 - 11x_2 + 5x_3 \leq 0 \wedge x_1 + x_2 - 2x_3 > 0\big\} \\
q_3: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 > 0 \wedge -x_1 - 11x_2 + 5x_3 > 0 \wedge x_1 + x_2 - 2x_3 \leq 0\big\} \\
q_4: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 > 0 \wedge -x_1 - 11x_2 + 5x_3 \leq 0 \wedge x_1 + x_2 - 2x_3 \leq 0\big\} \\
q_5: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 \leq 0 \wedge -x_1 - 11x_2 + 5x_3 > 0 \wedge x_1 + x_2 - 2x_3 > 0\big\} \\
q_6: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 \leq 0 \wedge -x_1 - 11x_2 + 5x_3 \leq 0 \wedge x_1 + x_2 - 2x_3 > 0\big\} \\
q_7: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 \leq 0 \wedge -x_1 - 11x_2 + 5x_3 > 0 \wedge x_1 + x_2 - 2x_3 \leq 0\big\} \\
q_8: & \big\{(x_1,x_2,x_3) \in \mathbb{R}^3 \mid x_2 - x_3 \leq 0 \wedge -x_1 - 11x_2 + 5x_3 \leq 0 \wedge x_1 + x_2 - 2x_3 \leq 0\big\}
\end{array}$$

To illustrate the computation of these sets we consider q_1. Starting from partition $\mathcal{P} = \{S_1, S_2, S_3, S_4\}$ we obtain the state space partition $\Upsilon^{-1}(\mathcal{P}) = \{\Upsilon^{-1}(S_1), \Upsilon^{-1}(S_2), \Upsilon^{-1}(S_3), \Upsilon^{-1}(S_4)\}$. If we denote by P the set $\Upsilon^{-1}(S_1)$, we have:

$$\begin{array}{rcl}
P &=& \{x \in \mathbb{R}^3 \mid x_2 - x_3 > 0 \wedge x_1 + x_2 - 2x_3 > 0\} \\
\mathrm{Pre}(P) &=& \{x \in \mathbb{R}^3 \mid \exists u \in \mathbb{R}^2 \; x_1 - 2x_3 - 15x_2 + u_2 > 0 \wedge -x_1 - 11x_2 + 5x_3 > 0\} \\
&=& \{x \in \mathbb{R}^3 \mid -x_1 - 11x_2 + 5x_3 > 0\} \\
\mathrm{Pre}(P) \cap P &=& \{x \in \mathbb{R}^3 \mid x_2 - x_3 > 0 \wedge x_1 + x_2 - 2x_3 > 0 \wedge -x_1 - 11x_2 + 5x_3 > 0\} = q_1
\end{array}$$

3. LTL formulas and Büchi automata

Linear Time Temporal logic is a very powerful specification mechanism since it allows one to express complex requirements through simple formulas. Even though the use of temporal logic is now widely used for verification of software systems [CPG99, MP92], we argue that temporal logic is equally relevant to synthesis problems. In this section we define LTL syntax and semantics, provide simple examples to illustrate the definitions, and discuss the translation of LTL formulas into Büchi automata.

3.1. LTL syntax and semantics.

We start with a finite set $\mathcal{P}$ of predicates from which more complex formulas can be built. Even though $\mathcal{P}$ can be an arbitrary finite set we shall keep in mind the particular case where $\mathcal{P}$ is the observation space of $T_\Sigma^{\mathcal{P}}$. LTL formulas are then obtained through the following recursive definition.

- **true**, **false** and p are LTL formulas for all $p \in \mathcal{P}$;
- if φ_1 and φ_2 are LTL formulas, then $\varphi_1 \wedge \varphi_2$ and $\neg\varphi_1$ are LTL formulas;
- if φ_1 and φ_2 are LTL formulas, then $\circ\varphi_1$ and $\varphi_1 \mathbf{U} \varphi_2$ are LTL formulas.

As usual, disjunction $\varphi_1 \vee \varphi_2$ and implication $\varphi_1 \Rightarrow \varphi_2$ are defined as the abbreviations of $\neg(\varphi \wedge \varphi_2)$ and $\neg\varphi_1 \vee \varphi_2$, respectively. The operator $\circ$ is read as "next", with the meaning that the formula it precedes will be true in the next time step. The second operator $\mathbf{U}$ is read as "until" and the formula $\varphi_1 \mathbf{U} \varphi_2$ specifies that φ_1 must hold until φ_2 holds. From the $\mathbf{U}$ operator we can define other commonly used operators:

$$\diamond\varphi = \mathbf{true}\,\mathbf{U}\,\varphi \tag{3.1}$$
$$\square\varphi = \neg\diamond\neg\varphi \tag{3.2}$$

Formula $\square\varphi$ is read as "always φ" and requires φ to be true for all future time, while formula $\diamond\varphi$ reads "eventually φ" and states that φ will become true at some point in the future. A unique interpretation of LTL formulas is obtained by defining LTL semantics. LTL formulas are interpreted over sequences of predicate[4] values $s \in \mathcal{P}^\omega$. We say that string s satisfies formula φ at time t, denoted by $s(t) \models \varphi$, if formula φ holds at time t along trajectory s. The satisfaction relation is defined as follows:

For any $p \in \mathcal{P}$, LTL formulas φ_1, φ_2, and $t \in \mathbb{N}$:

- $s(t) \models p$ iff $p = s(t)$,
- $s(t) \models \neg p$ iff $p \neq s(t)$,
- $s(t) \models \varphi_1 \wedge \varphi_2$ iff $s(t) \models \varphi_1$ and $s(t) \models \varphi_2$,
- $s(t) \models \circ\, \varphi_1$ iff $s(t+1) \models \varphi_1$,
- $s(t) \models \varphi_1 \,\mathbf{U}\, \varphi_2$ iff $\exists t' \geq t$ such that for all k, $t \leq k \leq t'$ $s(k) \models \varphi_1$ and $s(t') \models \varphi_2$.

Finally we say that a sequence s satisfies formula φ iff $s(0) \models \varphi$.

Example 3.1. As a first example consider the formula $\Box p$ for $p \in \mathcal{P}$. This formula defines an invariance property by requiring p to hold for all $t \in \mathbb{N}$. Such specification is useful, for example, in one wants to restrict the activity of a certain control system to a set of operating conditions defined by the predicate p. The semantics of $\Box p$ can be obtained from the semantics of $\mathbf{U}$ using the definition of $\Box$ given in (3.2) or given directly as $s \models \Box\varphi$ iff $s(i) \models \varphi$ for all $i \in \mathbb{N}$. It then follows that the unique string $s \in \mathcal{P}^\omega$ satisfying $\Box p$ is:

$$ppppppppppp\ldots$$

When each predicate $p \in \mathcal{P}$ is an element of a finite partition of the observation space O of T_Σ, requirement $\Box p$ specifies that trajectories of Σ should start in the set defined by p and stay in that set forever.

Example 3.2. Consider now the formula $p_1 \mathbf{U} p_2$. According to the above introduced semantics, every string satisfying this formula is of the form:

$$\begin{gathered}p_2?????????????\ldots\\ p_1p_2???????????\ldots\\ p_1p_1p_2?????????\ldots\\ p_1p_1p_1p_2???????\ldots\\ p_1p_1p_1p_1p_2?????\ldots\\ \vdots\end{gathered}$$

where we have used the symbol ? to denote an occurrence of any predicate in $\mathcal{P}$. The first string satisfies $p_1\mathbf{U}p_2$ by satisfying p_2, after which formula $p_1\mathbf{U}p_2$ no longer imposes any constraint on the string. The remaining strings satisfy $p_1\mathbf{U}p_2$ by initially satisfying p_1 until they satisfy p_2 at some later time. Once p_2 is satisfied, any predicate in $\mathcal{P}$ is allowed to occur in the string since $p_1\mathbf{U}p_2$ is already true. Operator $\mathbf{U}$ is very useful to capture temporal ordering of control requirements. One can specify, for example, that the temperature and humidity in a building should stay within certain bounds (as specified by predicates on the observation space) until the end of working hours, or that an aircraft should stay at a certain altitude until the descent phase is initiated, etc.

Example 3.3. More complex (and useful) formulas usually involve nesting of temporal operators. One such example is obtained by combining the $\Box$ operator with the formula $p_1\mathbf{U}p_2$ resulting in the formula $p_1\mathbf{U}\Box p_2$. Intuitively this formula requires p_2 to hold for all time, or that p_1 holds until at some later time p_2 holds for

[4]LTL formulas are usually interpreted over sequences of *sets* of predicate values. Since in this paper we identify $\mathcal{P}$ with a finite partition of the observation space O of some transition system, it follows that for each $t \in \mathbb{N}$ only one predicate is satisfied thus justifying our choice for LTL semantics.

all future time. A string satisfying $p_1\mathbf{U}\Box p_2$ is necessarily of the form:

$$\begin{gathered}p_2p_2p_2p_2p_2\cdots\\ p_1p_2p_2p_2p_2\cdots\\ p_1p_1p_2p_2p_2\cdots\\ p_1p_1p_1p_2p_2\cdots\\ \vdots\end{gathered}$$

Formula $p_1\mathbf{U}\Box p_2$ can be used to model convergence towards the operating conditions described by p_2 through a particular subset of the state space described by p_1.

Different combinations of $\Box$ and $\mathbf{U}$ result in formulas with different meaning. For example any string satisfying the formula $\Box(p_1\mathbf{U}p_2)$ must satisfy $p_1\mathbf{U}p_2$ at every time step, which implies that at every time step either p_2 holds or p_1 holds until p_2 holds at some future time. Examples of strings satisfying $\Box(p_1\mathbf{U}p_2)$ are given below:

$$\begin{gathered}p_2p_2p_2p_2p_2p_2p_2p_2p_2p_2p_2p_2p_2\cdots\\ p_2p_1p_2p_1p_2p_1p_2p_1p_2p_1p_2p_1p_2\cdots\\ p_1p_1p_1p_2p_1p_1p_2p_1p_1p_2p_1p_1p_2\cdots\\ p_1p_1p_1p_1p_1p_1p_1p_1p_1p_1p_2p_2p_2\cdots\end{gathered}$$

Example 3.4. We now return to Example 2.13. The only predicate appearing in formula (2.8), is an element of the observation space of T_Σ. According to LTL semantics and the abbreviation $\Diamond_3\varphi = \varphi\vee\circ\varphi\vee\circ\circ\varphi\vee\circ\circ\circ\varphi$, some of the infinite strings satisfying (2.8) are given by:

$$\begin{gathered}S_1S_1S_1S_1S_1S_1S_1\ldots\\ S_1S_2S_1S_2S_1S_2S_1\ldots\\ S_1S_1S_2S_1S_1S_2S_1\ldots\\ S_1S_2S_2S_1S_2S_2S_1\ldots\end{gathered}$$

The reader should verify for himself that in every of the above strings, every length 4 sub-string contains S_1. This shows that the above strings satisfy the specification formula since $\Box\Diamond_3 S_1$ requires that at every time step t, $\Diamond_3 S_1$ holds, meaning that S_1 should hold at t or that it should hold at $t+1$, $t+2$ or $t+3$. This simple example also shows how tedious and error prone it is to list all the possible strings satisfying the very simple formula (2.8).

When a LTL formula is interpreted over observed sequences in $L_\omega(T_\Sigma^{\mathcal{P}})$ each predicate corresponds to a subset of $\mathbb{R}^m$ and the specification defines how trajectories of Σ interact with these sets. This is a convenient and formal way of expressing control requirements for discrete-time linear systems. If every string in $L_\omega(T_\Sigma^{\mathcal{P}})$ satisfies formula φ we simply say that $T_\Sigma^{\mathcal{P}}$ satisfies φ which is denoted by $T_\Sigma^{\mathcal{P}}\models\varphi$. We shall use a similar notation for $T_\Sigma^{\mathcal{P}'}$ even though predicates in φ do not correspond to sets in $\mathcal{P}'$. We shall use the notation $T_\Sigma^{\mathcal{P}'}\models\varphi$ when for every $r\in L_\omega(T_\Sigma^{\mathcal{P}'})$, $\pi_{\mathcal{P}'\mathcal{P}}(r)\models\varphi$. In general it is not the case that $T_\Sigma^{\mathcal{P}'}\models\varphi$ and a controller T_c needs to be constructed to ensure satisfaction of φ by the closed-loop system. Such a controller is built from $T_\Delta^{\mathcal{P}'}$ and a Büchi automaton describing the specification. It is therefore necessary to translate the LTL specification formula φ into a Büchi automaton.

3.2. **Büchi automata.**

The strings satisfying a given LTL formula can also be compactly described in terms of a finite operational model. Such model is slightly more complex than a transition system since LTL formulas specify both the finite and infinite behavior of strings. Consider for example the formula $\Diamond p = \mathbf{true}\,\mathbf{U}\,p$ for $p\in\mathcal{P}$. This formula requires p to hold at some time in the future. Given a string $s\in\mathcal{P}^\omega$, we cannot decide if $s\models\Diamond p$ by looking at a finite prefix of s since p can always appear at a later point in time. This shows that we need to equip transition systems with an additional mechanism describing the behavior of strings "at infinity". These new transition systems are called Büchi automata.

Definition 3.5. A *Büchi automaton* is a tuple $A = (T_A, F) = ((Q, Q^0, \longrightarrow, O, \Upsilon), F)$, where T_A is a finite transition system and $F \subseteq Q$ is a set of final states. A string $s \in Q^\omega$ is a run of A if $s(1) \in Q^0$, $s(i) \longrightarrow s(i+1)$ for $i \in \mathbb{N}$ and there exist infinitely many $i \in \mathbb{N}$ such that $s(i) \in F$.

Since every Büchi automaton A carries an underlying transition system structure T_A, it also defines generated languages and ω-languages. In addition, final states allows one to introduce the notion of accepted language:

Definition 3.6. Let $A = (Q, Q^0, \longrightarrow, O, \Upsilon, F)$ be a Büchi automaton. The *language accepted* by A, denoted by $L_\omega(A)$, is defined as:

$$L_\omega(A) \quad = \quad \left\{ r \in O^\omega \mid r = \Upsilon(s) \text{ for some initialized run } s \text{ of } A \right\}$$

Büchi automata accept languages which are more general than the languages generated by transition systems. Since given a transition system T we can always construct a Büchi automaton A with $F = Q$, leading to $L_\omega(A) = L_\omega(T_A) = L_\omega(T)$, we can regard transition systems as a subclass of Büchi automata. The relevance of Büchi automata comes from the fact that for any LTL formula φ it is possible to construct a Büchi automaton A_φ accepting every string satisfying formula φ. This fact was first shown by Büchi [B6̈2] in the context of decidability of first and second order monadic theories of one successor. Although decidability of the translation between LTL formulas and Büchi automata was settled by Büchi's work, the complexity of such translations has been improved through the years by different authors. The resulting automata are, in the worst case, exponential in the length of the translated formula. However, current practice in computer aided verification shows that such worst case complexity is seldom achieved. Since this translation is well documented in the literature we point the interested reader to the survey [Wol00] and to the algorithms described in [SB00, GPVW96] for more details. We now return to the periodic synchronization example converting the specification formula into a Büchi automaton.

Example 3.7. Recall the specification formula discussed in Example 2.13 that we repeat here for convenience:

$$\varphi = \Box \diamond_3 \left(p_1 \wedge p_2 \right)$$

Specification formula φ can be translated into the transition system[5] T_φ represented in Figure 4 where $p_1 \wedge p_2$ has been abbreviated by S_1, and $\neg(p_1 \wedge p_2)$ by S_2.

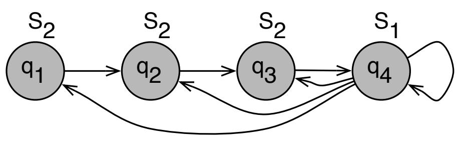


FIGURE 4. Transition system T_φ corresponding to specification formula φ.

Note that starting from any state of T_φ state q_4 will be necessarily reached in at most 3 steps. Since the observation associated with q_4 is S_1 this implies that at any time step, S_1 will hold no later than 3 time steps implying that any string generated by T_φ satisfies φ.

We shall not discuss further Büchi automata as we will not have the oportunity of using them in this paper. However, they are essential for the construction of a discrete controller or supervisor T_c for $T_\Delta^{\mathcal{P}'}$ enforcing φ as discussed in the next section.

[5]In this case the final states of Büchi automaton A_φ are all of its states and we can equivalently represent A_φ by its underlying transition system.

4. Supervisory synthesis

The existence of finite bisimulations for linear systems, discussed in Section 2, enables the design of controllers enforcing LTL specifications at the purely discrete level. Such control problems on infinite behaviors have been studied in the discrete event systems community [RW89, KG95, CL99] and in this section we review the results and concepts required for our purposes. We start by introducing a notion of parallel composition between transition systems modeling interaction between components. This interaction can be understood as a form of control where a supervisor is designed to modify (restrict) the behavior of another (transition) system by interconnection.

Definition 4.1. Let $T_1 = (Q_1, Q_1^0, \longrightarrow_1, O, \Upsilon_1)$ and $T_2 = (Q_2, Q_2^0, \longrightarrow_2, O, \Upsilon_2)$ be two transition systems with the same observation space O. The *parallel composition* of T_1 and T_2 (with observation synchronization) is denoted by:

$$T_1 \parallel_O T_2 = (Q_\parallel, Q_\parallel^0, \longrightarrow_\parallel, O, \Upsilon_\parallel)$$

where:

- $Q_\parallel = \{(q_1, q_2) \in Q_1 \times Q_2 \mid \Upsilon_1(q_1) = \Upsilon_2(q_2)\}$;
- $Q_\parallel^0 = \{(q_1, q_2) \in Q_1^0 \times Q_2^0 \mid \Upsilon_1(q_1) = \Upsilon_2(q_2)\}$;
- $(q_1, q_2) \longrightarrow_\parallel (q_1', q_2')$ for $(q_1, q_2), (q_1', q_2') \in Q_\parallel$ iff $q_1 \longrightarrow_1 q_1'$ and $q_2 \longrightarrow_2 q_2'$;
- $\Upsilon_\parallel(q_1, q_2) = \Upsilon_1(q_1) = \Upsilon_2(q_2)$.

The presented definition of parallel composition is not the usual synchronous product used in the supervisory control literature since we have defined transition systems with observations on the states rather than on the transitions. Nevertheless, the language of transition system $T_1 \parallel_O T_2$ can still be expressed in terms of the languages of T_1 and T_2.

Proposition 4.2. *Let T_1 and T_2 be transition systems with the same observation space O. The following equalities are always satisfied:*

$$L(T_1 \parallel_O T_2) = L(T_1) \cap L(T_2) \qquad L_\omega(T_1 \parallel_O T_2) = L_\omega(T_1) \cap L_\omega(T_2)$$

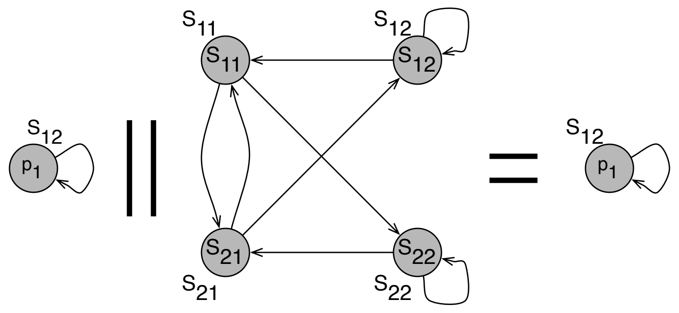
The previous proposition shows that a controller T_c can restrict the behavior of T_Σ through language intersection in order to eliminate strings $s \in L_\omega(T_\Sigma)$ which do not satisfy the specification formula φ. Furthermore, as asserted in the next result, a controller for $T_\Sigma^{\mathcal{P}'}$ can be obtained by working with the finite transition system $T_\Delta^{\mathcal{P}'}$.

Proposition 4.3. *Let Σ be a controllable linear system, let φ be a LTL formula with predicates denoting sets in a finite partition $\mathcal{P}$ of the observation space of T_Σ and let $\mathcal{P}'$ be the finite refinement of state space partition $\Upsilon^{-1}(\mathcal{P})$ whose existence is asserted by Theorem 2.8. There exists a controller T_c satisfying $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \models \varphi$ iff there exists a controller T_c satisfying $T_c \parallel_O T_\Delta^{\mathcal{P}'} \models \varphi$.*

Proof. It follows at once from the properties of bisimulation, see for example [Mil89], that for any transition system T_c, $T_\Sigma^{\mathcal{P}'} \cong T_\Delta^{\mathcal{P}'}$ implies $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \cong T_c \parallel_O T_\Delta^{\mathcal{P}'}$. The result now follows from Proposition 2.4. $\square$

At this point the reader may be wondering why the previous result is concerned with the existence of a controller for $T_\Sigma^{\mathcal{P}'}$ and not for $T_\Sigma^{\mathcal{P}}$. Since a controller modifies the behavior of the system to be controlled by parallel composition with observation synchronization, working with $T_\Sigma^{\mathcal{P}'}$ is preferable as the observation space of $T_\Sigma^{\mathcal{P}'}$ offers more detailed information regarding the dynamics of T_Σ than the observation space of $T_\Sigma^{\mathcal{P}}$.

Recalling that for any LTL formula φ we can always construct a Büchi automaton A_φ recognizing every string satisfying φ and recalling that from $T_\Delta^{\mathcal{P}'}$ we can construct a Büchi automaton $A_\Delta^{\mathcal{P}'}$ satisfying $L_\omega(A_\Delta^{\mathcal{P}'}) = L_\omega(T_\Delta^{\mathcal{P}'})$, the problem of constructing a controller enforcing φ can be conceptually reduced to the following steps:



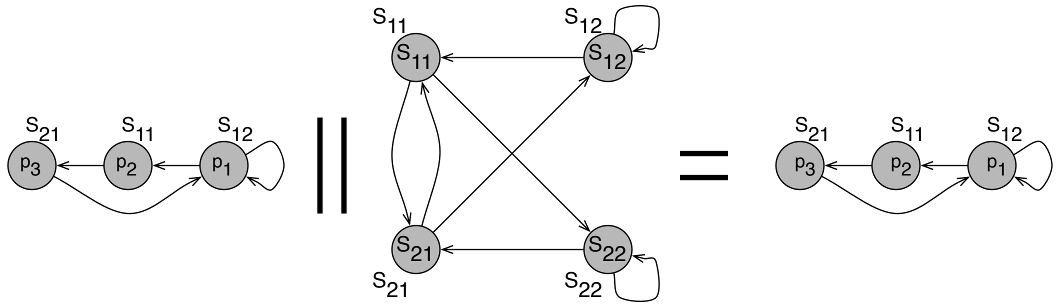


FIGURE 5. Parallel composition of $T_\Delta^{\mathcal{P}'}$ with two different controllers for the problem described in Example 2.13.

(1) Construct $T_\Sigma^{\mathcal{P}'}$ from Σ and $\mathcal{P}$;
(2) Construct $T_\Delta^{\mathcal{P}'}$ from $T_\Sigma^{\mathcal{P}'}$;
(3) Construct $A_\Delta^{\mathcal{P}'}$ from $T_\Delta^{\mathcal{P}'}$;
(4) Construct A_φ from φ;
(5) Construct a Büchi automaton controller A_c satisfying $\pi_{\mathcal{P}'\mathcal{P}}\big(L_\omega(A_c) \cap L_\omega(A_\Delta^{\mathcal{P}'})\big) \subseteq L_\omega(A_\varphi)$.

The first four steps have already been described in this paper and the fifth step has been extensively studied in the discrete event systems literature [Ram89, KGM92, TW94, KG95]. For the purposes of this paper we will simply assume the existence of A_c defined in (5). Note that if no such A_c exists, then Proposition 4.3 asserts that no controller for $T_\Sigma^{\mathcal{P}'}$ (and consequently for $T_\Sigma^{\mathcal{P}}$) exists. Furthermore, we will also assume that A_c can be modeled by a transition system, that is, there exists a transition system T_c satisfying $L_\omega(T_c) \cap L_\omega(A_\Delta^{\mathcal{P}'}) = L_\omega(A_c) \cap L_\omega(A_\Delta^{\mathcal{P}'})$. As discussed in [KG95], a finite supervisor that is implementable[6] necessarily has *finite* memory and therefore can only restrict the infinite behavior of $A_\Delta^{\mathcal{P}'}$ based on *finite* length observations. Therefore, for any implementable Büchi automaton controller A_c enforcing φ there exists a finite transition system T_c satisfying $L_\omega(T_c \parallel T_\Delta^{\mathcal{P}'}) = L_\omega(T_c) \cap L_\omega(T_\Delta^{\mathcal{P}'}) = L_\omega(A_c) \cap L_\omega(A_\Delta^{\mathcal{P}'})$. We refer the reader to [Ram89, KGM92, TW94, KG95] for more details on the existence and computation of T_c and return to Example 2.13.

Example 4.4. Figure 5 shows two different controllers enforcing LTL formula (2.8) on the transition system displayed in Figure 3. We can see that both controllers enforce LTL formula (2.8) on T_Δ since on the first case the language of the parallel composition consists of strings of the form $S_1S_1S_1S_1\ldots$, while in the second case it consists of strings in which occurrence of S_2 (if any) is immediately followed by an occurrence of S_1.

[6]We use the expression implementable to denote software, hardware or software and hardware implementations.

5. Refining the closed-loop

In the previous section we outlined how a finite controller T_c for $T_\Delta^{\mathcal{P}'}$ enforcing a desired LTL specification can be obtained. In this section we will see that we can also extract from T_c the continuous inputs required to enforce the specification on Σ. The explicit modeling of the control inputs available to Σ will result in a hybrid closed-loop behavior. This motivates the introduction of discrete-time linear hybrid systems and their corresponding transition systems.

Definition 5.1. A *discrete-time linear hybrid system* $H = (X, X^0, \{A_q, B_q\}_{q\in Q}, \delta, U)$ consists of the following elements:

- The state space $X = \coprod_{q\in Q} \mathbb{R}^{n_q}$ where Q is a finite set of states and $n_q \in \mathbb{N}$ for each $q \in Q$;
- A set of initial states $X^0 \subseteq X$,
- The continuous dynamics $\{A_q, B_q\}_{q\in Q}$ where for each $q \in Q$, $(A_q, B_q) \in \mathbb{R}^{n_q\times n_q} \times \mathbb{R}^{n_q\times m_q}$ defines a discrete-time linear control system $x(t + 1) = A_q x(t) + B_q u(t)$ with inputs restricted to the set $U(q(t), x(t)) \subseteq \mathbb{R}^{m_q}$;
- The discrete dynamics $\delta : Q \times \mathbb{R}^{n_q} \to 2^Q$ which assigns to each discrete $q \in Q$ and continuous $x \in \mathbb{R}^{n_q}$ state the discrete successor states $\delta(q(t), x(t)) \subseteq Q$.

Similarly to the purely continuous case, hybrid systems can also be embedded into the class of transition systems. Assuming the continuous dynamics to be controllable we can define transition system:

$$T_H = \big(X, X^0, \longrightarrow_H, O, \Upsilon\big)$$

associated with a discrete-time linear hybrid system $H = (X, X^0, \{A_q, B_q\}_{q\in Q}, \delta, U)$ by $(q, x) \longrightarrow_H (q', x')$ iff $x' = A_q x + B_q u$, $q' \in \delta(q, x)$ and $u \in U(q, x)$. The observation set and map are defined by $O = \coprod_{q\in Q} O_q$ and $\Upsilon(q, x) = \Upsilon_q(x)$ where O_q and Υ_q are the observation set and map associated with the control systems defined by (A_q, B_q) for each $q \in Q$. The importance of embedding linear hybrid systems into the class of transition systems resides in the possibility of formally defining a notion of correct implementation.

Definition 5.2. Let Σ be a controllable linear system, let φ be a LTL formula with predicates denoting sets in a finite partition $\mathcal{P}$ of the observation space of T_Σ and let T_c be a controller enforcing φ on the finite bisimilar quotient $T_\Delta^{\mathcal{P}'}$, that is, $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \models \varphi$. Linear hybrid system H is said to be a *correct implementation* of the closed-loop behavior $T_c \parallel_O T_\Sigma^{\mathcal{P}'}$ if $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \cong T_H$.

A hybrid implementation $H = (X, X^0, \{A_q, B_q\}_{q\in Q}, \delta, U)$ of a desired closed-loop behavior $T_c \parallel_O T_\Sigma^{\mathcal{P}'}$ can be immediately obtained from $T_c \parallel_O T_\Sigma^{\mathcal{P}'} = (Q_\parallel, Q_\parallel^0, \longrightarrow_\parallel, O, \Upsilon_\parallel)$ by defining:

$$X = Q_\parallel \tag{5.1}$$
$$X^0 = Q_\parallel^0 \tag{5.2}$$
$$A_q = A \tag{5.3}$$
$$B_q = B \tag{5.4}$$
$$\delta(q, x) = \{q' \in Q \mid q \longrightarrow_\parallel q'\} \tag{5.5}$$
$$U(q, x) = \big\{u \in \mathbb{R}^m \mid \pi_{\mathcal{P}'}(Ax + Bu) \in \Upsilon_\parallel(\delta(q, x))\big\} \tag{5.6}$$

The construction of H represents the last step required for the solution of Problem 1.1 as we now summarize in the following result.

Theorem 5.3. *Let Σ be a controllable linear system satisfying Assumptions* **A.I** *and* **A.II** *and defined by matrices A and B with rational entries. For any LTL formula φ with predicates denoting semi-linear sets on the observation space of T_Σ, it is decidable to determine the existence of a controller*[7] *T_c satisfying $T_c \parallel_O T_\Sigma^{\mathcal{P}'} \models \varphi$.*

[7]Note that depending on the formula φ, for example if it defines safety of liveness properties, one may have to impose additional constraints on T_c.

Furthermore, when such a controller exists it admits the hybrid implementation defined by (5.1) through (5.6) which is effectively computable.

Proof. By Corollary 2.12 we can effectively compute $T_{\Delta}^{\mathcal{P}'}$. Since we can also effectively compute A_{φ} from φ, it follows from standard results in supervisory control [Ram89, KGM92, TW94, KG95] that it is decidable to determine the existence of a controller T_c for $T_{\Delta}^{\mathcal{P}'}$ and also that T_c is effectively computable. The result now follows from the fact that steps (5.1) through (5.6) are effectively computable since the sets denoted by φ are semi-linear. $\square$

It is important to emphasize that the resulting hybrid controller implicitly defined by H can be obtained in a totally automated fashion. The closed-loop system is still a control system in the sense that at every state different future evolutions are possible under the action of different input values. This is natural since the closed-loop model can now be further controlled to satisfy additional objectives or optimized to extremize certain performance criteria. Another important characteristic of the presented method is the automatic synthesis of both the switching logic (implemented by software) and the continuous aspects of control. In fact, a software implementation of the controller implicitly defined by H can be automatically generated from H by translating each discrete state of H into code reading the state x from sensors, computing u based on q and x and sending u to the actuators. This fact is especially important since verification of hybrid systems is currently limited to systems with very simple continuous dynamics such as timed automata. The proposed approach thus results in systems that satisfy the specification by design while enlarging the class of system that can be shown to operate correctly.

Example 5.4. We now illustrate the construction of the hybrid implementation of the closed-loop systems displayed in Figure 5. We focus on the construction of U which is the only nontrivial element in the definition of H. The first closed-loop system in Figure 5 consists of a single discrete state and the corresponding set U is defined by:

$$U(p_1, x) = \Big\{u \in \mathbb{R} \mid \pi_{\mathcal{P}'}(Ax + Bu) \in \{p_1\}\Big\} = \Big\{u \in \mathbb{R} \mid -\frac{1}{2} < u < \frac{1}{2}\Big\}$$

The resulting hybrid implementation is represented in Figure 6. The second supervisor has three discrete

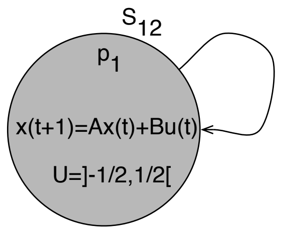


FIGURE 6. Hybrid implementation of the closed-loop behavior enforced by the first supervisor represented in Figure 5.

states for which we need to compute the input set U. For discrete state p_1 we have:

$$U(p_1, x) = \Big\{u \in \mathbb{R} \mid \pi_{\mathcal{P}'}(Ax + Bu) \in \{p_1, p_2\}\Big\} = \Big\{u \in \mathbb{R} \mid -\frac{1}{2} < u < \frac{1}{2} \vee \Big(u \leq -\frac{1}{2} \vee u \geq \frac{1}{2}\Big)\Big\} = \mathbb{R}$$

while for states p_2 and p_3 the set U is given by:

$$U(p_2, x) = \Big\{u \in \mathbb{R} \mid -\frac{1}{2} < u < \frac{1}{2}\Big\}$$

$$U(p_3, x) = \Big\{u \in \mathbb{R} \mid -\frac{1}{2} < u < \frac{1}{2}\Big\}$$

The resulting hybrid implementation is represented in Figure 7.

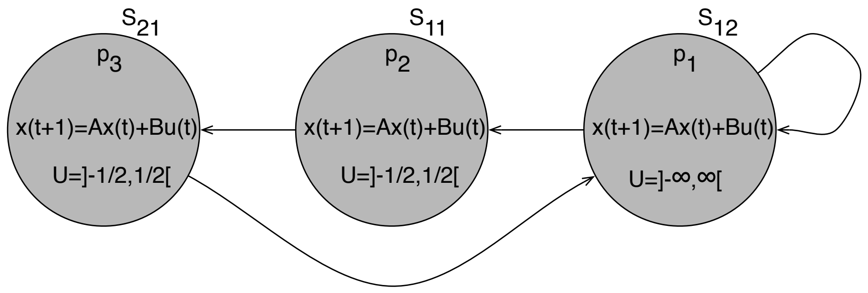


FIGURE 7. Hybrid implementation of the closed-loop behavior enforced by the second supervisor represented in Figure 5.

6. DISCUSSION

This paper presented an approach for the fully automated synthesis of controllers enforcing LTL specifications for linear systems. The resulting controllers are of hybrid nature combining the continuous dynamics of the original control system with the switching logic required to implement the desired specification. We can thus see these hybrid models as abstract descriptions of the embedded software required for its implementation. Since the resulting closed-loop system is guaranteed to satisfy the specification by construction, the presented synthesis technique enlarges the class of embedded systems for which formal guarantees of operation can be given.

The proposed approach can be further improved in terms of complexity. Compositional design techniques where different controllers are designed for different aspects of the specification and later combined into a controller for the overall specification allow one to overcome the complexity of translating LTL formulas into Büchi automata. Similarly, design with coarser finite abstractions of $T_\Sigma^{\mathcal{P}'}$ than $T_\Delta^{\mathcal{P}'}$ sidesteps the complexity involved in the construction of $T_\Delta^{\mathcal{P}'}$. The authors are currently investigating these issues as well as synthesis for other temporal logics such as CTL and μ-calculus.

REFERENCES

[ACH$^+$95] R. Alur, C.Courcoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicollin, A.Olivero, J. Sifakis, and S. Yovine. Hybrid automata: An algorithmic approach to specification and verification of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AHLP00] Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, 2000.

[AM95] M. Antoniotti and B. Mishra. ?Discrete-Event Models + Temporal Logic = Supervisory Controller: Automatic Synthesis of Locomotion Controllers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1441–1446, 1995.

[AM02] Yasmina Abdeddaim and Oded Maler. Preemptive job-shop scheduling using stopwatch automata. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2002*, volume 2280 of *Lecture Notes in Computer Science*, pages 113–126. Springer, 2002.

[ASY01] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 89–104. Springer-Verlag, 2001.

[ATP01] R. Alur, S. La Torre, and G. Pappas. Optimal paths in weighted timed automata. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer-Verlag, 2001.

[B6̈2] J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. Logic, Method and Philos. Sci.*, pages 1–12, Stanford, 1962. Stanford University Press.

[BCG88] M.C. Browne, E.M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[BCR98] J. Bochnak, M. Coste, and M-F. Roy. *Real Algebraic Geometry.* Springer-Verlag, 1998.

[BFH90] A. Bouajjani, J.C. Fernandez, and N. Halbwachs. Minimal model generation. In R.P. Kurshan and E.M.Clarke, editors, *CAV90: Computer Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 197–203. Springer Verlag, 1990.

[BFH$^+$01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer-Verlag, 2001.

[BL00] R. W. Brockett and D. Liberzon. Quantized feedback stabilization of linear systems. *IEEE Transactions on Automatic Control*, 45(7):1279–1289, July 2000.

[BMP02] A. Bicchi, A. Marigo, and B. Piccoli. On the reachability of quantized control systems. *IEEE Transaction on Automatic Control*, 47(4):546–563, April 2002.

[Bro99] Mireille Broucke. A geometric approach to bisimulation and verification of hybrid systems. In Fritz W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1999.

[Bru70] P. Brunovsky. A classification of linear controllable systems. *Kybernetika*, 6(3):173–188, 1970.

[CJG02] J.A. Cook, Sun Jing, and J.W. Grizzle. Opportunities in automotive powertrain control applications. In *Proceedings of the 2002 International Conference on Control Applications*, pages xlii–li, 2002.

[CK01] A. Chotinan and B.H. Krogh. Verification of infinite state dynamical systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46(9):1401–1410, 2001.

[CKN98] J.E.R. Cury, B.H. Krogh, and T. Ninomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control : Special Issue on Hybrid Systems*, 43(4):564–568, April 1998.

[CL99] C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, Boston, MA, 1999.

[CMT99] I. Castellani, M. Mukund, and P.S. Thiagarajan. Synthesizing distributed transition systems from global specifications. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science, FSTTCS 1999*, volume 1739 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 1999.

[CP01] Y. Chitour and B. Piccoli. Controllability for discrete systems with a finite control set. *Mathematics of Control, Signals and Systems*, 14(2):173–193, 2001.

[CPG99] Edmund M. M. Clarke, Doron Peled, and Orna Grumberg. *Model Checking*. MIT Press, 1999.

[dAHM00] Luca de Alfaro, Thomas A. Henzinger, and Freddy Y. C. Mang. The control of synchronous systems. In Catuscia Palamidessi, editor, *CONCUR 2000 - Concurrency Theory, 11th International Conference*, volume 1877 of *Lecture Notes in Computer Science*, pages 458–473. Springer, 2000.

[dAHM01] Luca de Alfaro, Thomas A. Henzinger, and Freddy Y. C. Mang. The control of synchronous systems, Part ii. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR 2001 - Concurrency Theory, 12th International Conference*, volume 2154 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[DCG$^+$05] P. Dario, M.C. Carrozza, E. Guglielmelli, C. Laschi, A. Menciassi, S. Micera, and F. Vecchi. Robotics as a future and emerging technology: biomimetics, cybernetics, and neuro-robotics in european project. *IEEE Robotics & Automation Magazine*, 12(2):29–45, 2005.

[DM02] Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In Helmut Alt and Afonso Ferreira, editors, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science, STACS 2002*, volume 2285 of *Lecture Notes in Computer Science*, pages 571–582. Springer, 2002.

[DS97] B. Dutertre and V. Stavridou. Formal requirements analysis of an avionics control system. *IEEE Transactions on Software Engineering*, 23(5):267 – 278, 1997.

[EC82] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[EM01] N. Elia and S. K. Mitter. Stabilization of linear systems with limited information. *IEEE Transactions of Automatic Control*, 46(9):1384–1400, September 2001.

[FTM02] Marco Faella, Salvatore La Torre, and Aniello Murano. Dense real-time games. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, pages 167–176, Copenhagen, July 2002. IEEE Computer Society Press.

[GPVW96] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In Piotr Dembinski and Marek Sredniawa, editors, *Protocol Specification, Testing and Verification XV, Proceedings of the 15th IFIP Workshop*, IFIP Conference Proceedings, pages 3–18, Warsaw, June 1996. Chapman & Hall.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[HM00] T.A. Henzinger and R. Majumdar. Symbolic model checking for rectangular hybrid systems. In S. Graf, editor, *TACAS 2000: Tools and algorithms for the construction and analysis of systems*, Lecture Notes in Computer Science, New-York, 2000. Springer-Verlag.

[HU79] John E. Hopcroft and Jeffery D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, USA, 1979.

[HvS01] L.C.G.J.M. Habets and J. H. van Schuppen. Control of piecewise-linear hybrid systems on simplices and rectangles. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Sience*, pages 261–274. Springer-Verlag, 2001.

[JK01] S. Jiang and R. Kumar. Supervisory control of discrete event systems with CTL* temporal logic specifications. In *Proceedings of the of the 40th IEEE Conference on Decision and Control*, Orlando, FL, December 2001.

[Kal72] R. E. Kalman. Kronecker invariants and feedback. In L. Weiss, editor, *Ordinary Differential Equations*, pages 459–471. Academic Press, New York, 1972.

[KBr04] J. Khurshid and H. Bing-rong. Military robots - a glimpse from today and tomorrow. In *Proceedings of the 8th Control, Automation, Robotics and Vision Conference*, pages 771–777, 2004.

[KG95] R. Kumar and V.K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, 1995.

[KGM92] R. Kumar, V. Garg, and S.I. Markus. On supervisory control of sequential behaviors. *IEEE Transactions on Automatic Control*, 37(12):1978–1985, Dec. 1992.

[KS90] P.C. Kanellakis and S.A. Smolka. CCS expressions, finite-state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[LL00] M. Lansdaal and L. Lewis. Boeing's 777 systems integration lab. *IEEE Instrumentation & Measurement Magazine*, 3(3):13 – 18, 2000.

[LPS00] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals and Systems*, 13(1):1–21, March 2000.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[MD01] T. Moor and J. M. Davoren. Robust controller synthesis for hybrid systems using modal logic. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MP92] Z. Manna and A. Pnueli. *The temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, 1992.

[MSP05] C.J. Morris, S.A. Stauth, and B.A. Parviz. Self-assembly for microscale and nanoscale packaging: Steps toward self-packaging. *IEEE Transactions on Advanced Packaging*, 28:600 – 611, 2005.

[MT02a] P. Madhusudan and P.S. Thiagarajan. Branching time controllers for discrete event systems. *Theoretical Computer Science*, 274:117–149, March 2002.

[MT02b] P. Madhusudan and P.S. Thiagarajan. A decidable class of asynchronous distributed controllers. In Lubos Brim, Mojmír Kretínský, and Antonín Kucera, editors, *Proceedings of the 13th International Conference on Concurrency Theory, CONCUR 2002*, volume 2421 of *Lecture Notes in Computer Science*, pages 145–160. Springer, 2002.

[MW84] Z. Manna and P. Wolper. Synthesis of communication processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6:68–93, 1984.

[Ost89] Jonathan S. Ostroff. *Temporal logic for real-time systems*. John-Wiley and Sons, 1989.

[PLPB02] Stefania Pancanti, Laura Leonardi, Lucia Pallottino, and Antonio Bicchi. Optimal control of quantized linear systems. In Claire Tomlin and Mark R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Sience, pages 351–363. Springer-Verlag, 2002.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular inclusions. In *Computer Aided Verification*, pages 95–104, 1994.

[Ram89] P.J. Ramadge. Some tractable supervisory control problems for discrete event systems modeled by Buchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, 1989.

[RW89] P. J. Ramadage and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE*, 77(1):81–98, 1989.

[SB00] F. Somenzi and R. Bloem. Efficient büchi automata from ltl formulae. In E. Allen Emerson and A. Prasad Sistla, editors, *Computer Aided Verification, 12th International Conference*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263. Springer-Verlag, 2000.

[SH05] Z. Sun and K. Hebbale. Challenges and opportunities in automotive transmission control. In *Proceedings of the 2005 American Control Conference.*, pages 3284 – 3289, 2005.

[SKA01] J.A. Stiver, X.D. Koutsoukos, and P.J. Antsaklis. An invariant based approach to the design of hybrid control systems. *International Journal of Robust and Nonlinear Control*, 11(5):453–478, 2001.

[TH04] J. Teutsch and E. Hoffman. Aircraft in the future ATM system - exploiting the 4D aircraft trajectory. In *Proceedings of the 23rd Digital Avionics Systems Conference, 2004. DASC 04.*, volume 1, pages 3.B.2–1 – 3.B.2.22, 2004.

[TK02] Ashish Tiwari and Gaurav Khanna. Series of abstractions for hybrid automata. In Claire Tomlin and Mark R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 465–478. Springer-Verlag, 2002.

[Tri04] S. Tripakis. Undecidable problems in decentralized observation and control for regular languages. *Information Processing Letters*, 90(1):21–28, 2004.

[TW94] J.G. Thistle and W. M. Wonham. Supervision of infinite behavior of discrete-event systems. *SIAM Journal on Control and Optimization*, 32(4):1098–1113, July 1994.

[VSS+01] R. Vidal, S. Schaffert, O. Shakernia, J. Lygeros, and S. Sastry. Decidable and semi-decidable controller synthesis for classes of discrete time hybrid systems. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 1243–1248, Orlando, Deember 2001.

[Wol00] Pierre Wolper. Constructing automata from temporal logic formulas: A tutorial. In E. Brinksma, H. Hermanns, and J. P. Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

Department of Electrical Engineering, 268 Fitzpatrick Hall, University of Notre Dame, Notre Dame, IN 46556

E-mail address: `ptabuada@nd.edu`

Department of Electrical and Systems Engineering, 200 South 33rd Street, University of Pennsylvania, Philadelphia, PA 19104

E-mail address: `pappasg@ee.upenn.edu`