PESSOA: A tool for embedded control software synthesis[*]

Manuel Mazo Jr.
Dept. of Electrical Engineering
UCLA
mmazo@ee.ucla.edu

Anna Davitian
Dept. of Electrical Engineering
UCLA
davitian@ee.ucla.edu

Paulo Tabuada
Dept. of Electrical Engineering
UCLA
tabuada@ee.ucla.edu

ABSTRACT

In the past years several different abstraction techniques were developed to facilitate the analysis and design of hybrid systems. In this paper we complement the theoretical work underlying abstractions based on approximate simulations and bisimulations by moving from theory to practice. We introduce a tool, named Pessoa, for the synthesis of correct-by-design embedded control software. We describe the theoretical underpinnings of Pessoa, its algorithmic implementation, and illustrate its use on the synthesis of control software for several examples.

1. INTRODUCTION

The design of embedded control software is fraught with difficulties stemming from the complex interactions between the software and the physical world. In this paper we describe a correct-by-design approach to the synthesis of embedded control software supported by the new tool Pessoa. This represents a shift of emphasis from the validation and verification of already designed software to the design process itself. Such approach follows the long tradition of control theory in providing methods for the design of continuous controllers enforcing desired specifications. However, the requirements for today's applications go far beyond the traditional stability and robustness specifications used in control theory. Furthermore, embedded control software is nowadays much more complex than the implementation of a continuous feedback control law. It requires different modes of operation resorting to possibly different feedback control laws, a policy determining when each mode should be employed, the interaction with other software processes, and even communication through wired or wireless networks. To capture this broad set of requirements we have in mind the use of linear temporal logic or automata on infinite strings. These specification mechanism can describe the desired evolution of the continuous physical quantities being controlled as well as the desired interaction of the control software with other software processes and/or communication protocols.

[*]This work was partially supported by the NSF awards 0717188, 0820061, and 0834771.

Pessoa addresses the previously described difficulties by automating the synthesis of control software. Starting from a specification and a finite abstraction for the continuous system being controlled, an abstraction for the desired control software can be synthesized by resorting to well known algorithms developed in supervisory control of discrete-event systems [KG95, CL99] or algorithmic game theory [dAHM01, AVW03]. The resulting description of the control software can then be refined to a controller acting on the original control system and compiled into code. The current version of Pessoa supports the construction of finite abstractions of control systems, the synthesis of controllers enforcing simple specifications, and the refinement of controllers to Simulink blocks that can be used to simulate the closed-loop behavior. Future versions of Pessoa will support more complex specifications and compilation of the synthesized controllers into code. The construction of the finite abstractions is based on approximate simulations and bisimulations recently investigated in [PGT08] and reviewed in Section 5 for the class of linear control systems. Although similar results are available for nonlinear systems, we focus on the linear case which is natively[1] supported by Pessoa. The implementation details of Pessoa are described in Section 6 and examples of its use are given in Section 7. Pessoa is currently being improved in many directions, some of which are described in Section 8.

Most of the tools available for hybrid systems such as Ariadne [Ari], PHAVer [PHA], KeYmaera [KeY], Checkmate [Che], and HybridSAL [Hyba], focus on verification problems. Tools for the synthesis of controllers are more recent and include LTLCon [LTL] for linear control systems and the Hybrid Toolbox [Hybb] for piece-wise affine hybrid systems. What sets Pessoa apart from the existing synthesis tools is the nature of the abstractions (approximate simulations and bisimulations) and the classes of systems admitting such abstractions (linear, nonlinear, and switched [Tab09]). Although Pessoa does not support nonlinear and switched systems natively, they can already be handled as illustrated by the examples in Section 7.

[1]Linear control systems are natively supported in Pessoa Version 1.0. Nonlinear and switched systems can also be handled by Pessoa but require some additional effort by the user. For further information please consult the documentation in `http://www.cyphylab.ee.ucla.edu/Pessoa`.

2. NOTATION

For the reader's convenience we collect here the notation used throughout the paper. Given a set $Z \subseteq \mathbb{R}^n$, we denote by $\operatorname{int} Z$ the interior of Z and by $\operatorname{diam} Z$ the diameter of Z, *i.e.*, the largest distance between any two points in Z. By $[Z]_\eta$ we denote the set:

$$\{z \in Z \mid z_i = k_i\eta \text{ for some } k_i \in \mathbb{Z} \text{ and } i = 1, 2, \dots, n\}.$$

The infinity norm of $x \in \mathbb{R}^n$ is denoted by $\|x\| = \max_{i=1}^n |x_i|$ where x_i is the ith entry of the vector x. The closed ball of radius $r \in \mathbb{R}$ centered at $x \in \mathbb{R}^n$ is given by:

$$\mathcal{B}_r(x) = \{y \in \mathbb{R}^n \mid \|y - x\| \le r\}.$$

The domain Z of a function $f : Z \to W$ is denoted by $\operatorname{dom} f$. Given a set $Z \subseteq W$, the natural inclusion map taking $z \in Z$ to $z \in W$ is denoted by $\imath : Z \hookrightarrow W$ while the identity function on a set Z is denoted by 1_Z. The symbol $\oplus$ will be used to denote the Minkowski sum of sets defined by:

$$Z \oplus W = \{x \in \mathbb{R}^n \mid x = z + w \text{ for some } z \in Z \text{ and } w \in W\}$$

for any sets $Z, W \subseteq \mathbb{R}^n$.

3. FORMAL MODELS FOR SOFTWARE AND CONTROL

We start with a notion of system allowing us to describe both software and control systems.

Definition 1. A *system* $S = (X, X_0, U, \longrightarrow, Y, H)$ is a sextuple consisting of:

- a set of *states* X;
- a set of *initial states* $X_0 \subseteq X$;
- a set of *inputs* U;
- a *transition relation* $\longrightarrow\ \subseteq X \times U \times X$;
- a set of *outputs* Y;
- an *output map* $H : X \to Y$.

System S is said to be finite when X has finite cardinality and metric when Y is equipped with a metric $\mathbf{d} : Y \times Y \to \mathbb{R}_0^+$.

The "dynamics" of a system is described by the transition relation and we follow the standard practice of denoting an element $(x, u, x') \in\ \longrightarrow$ by the more suggestive notation $x \xrightarrow{u} x'$. Existence of a transition $x \xrightarrow{u} x'$ entails that upon the reception of input u at state x, system S evolves to state x'. For such a transition, state x' is called a *u-successor*, or simply *successor*, of state x. Similarly, x is called a *u-predecessor*, or *predecessor*, of state x'. Note that, since $\longrightarrow\ \subseteq X \times U \times X$ is a relation, for any state and any input $u \in U$ there may be: no u-successors, one u-successor, or many u-successors. For conciseness, we denote the set of u-successors of a state x by $\mathrm{Post}_u(x)$. Since $\mathrm{Post}_u(x)$ may be empty, we denote by $U(x)$ the set of inputs $u \in U$ for which $\mathrm{Post}_u(x)$ is nonempty.

States x and x' are regarded as internal to the system and only the outputs $H(x)$ and $H(x')$ are externally visible.

Models similar to the one introduced in Definition 1 are routinely used to describe software for the purpose of verification. In addition to software, they can also be used to describe control systems. Before detailing such use, we introduce the notion of control system used in this paper.

Definition 2. A *linear control system* is a quintuple $\Sigma = (\mathbb{R}^n, \mathsf{U}, \mathcal{U}, A, B)$ consisting of:

- the state space $\mathbb{R}^n$;
- the input space $\mathsf{U} \subseteq \mathbb{R}^m$;
- a set of input curves $\mathcal{U}$ whose elements $\upsilon :]a, b[\to \mathsf{U} \subseteq \mathbb{R}^m$ are[2] essentially bounded piecewise continuous functions of time with $a < 0 < b$;
- matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ describing the system dynamics.

A piecewise continuously differentiable curve $\xi :]a, b[\to \mathbb{R}^n$ is said to be a *trajectory* or *solution* of Σ if there exist an input curve $\upsilon :]a, b[\to \mathsf{U}$ satisfying:

$$\frac{d}{dt}\xi(t) = A\xi(t) + B\upsilon(t) \tag{1}$$

for almost all $t \in]a, b[$.

Although we have defined trajectories over open sets, we shall refer to trajectories $\xi : [0, \tau] \to \mathbb{R}^n$ defined on closed sets $[0, \tau]$, $\tau \in \mathbb{R}^+$, with the understanding of the existence of a trajectory $\xi' :]a, b[\to \mathbb{R}^n$ such that $\xi = \xi'|_{[0,\tau]}$. We also write $\xi_{x\upsilon}(t)$ to denote the point reached at time $t \in [0, \tau]$ under the input υ from state x.

Since we have in mind the implementation of embedded control software on digital platforms, we consider system models reflecting the discrete nature of time in these platforms. In particular, we restrict attention to the case where $\mathcal{U}$ consists of piecewise constant curves. This assumption reflects the fact that in most digital implementations of feedback control laws, the inputs are kept constant while the feedback control law is being recomputed by the microprocessor. Given a control system Σ and a time quantization parameter $\tau \in \mathbb{R}^+$, we consider the system $S_\tau(\Sigma) = (X, X_0, U, \longrightarrow, Y, H)$ associated with Σ and defined by:

- $X = \mathbb{R}^n$;
- $X_0 = \mathbb{R}^n$;
- $U = \mathcal{U}$;
- $x \xrightarrow{u} x'$ if the solution ξ of (1) satisfies $\xi_{x\upsilon}(\tau) = x'$ with $u = \upsilon \in U$;
- $Y = X$;
- $H = 1_X$.

[2]We do not allow $\mathcal{U}$ to be the set of all curves from $]a, b[$ to U so that existence and uniqueness of solutions for the differential equation (1) can be guaranteed.

System $S_\tau(\Sigma)$ is infinite since its set of states is $\mathbb{R}^n$. In Section 5 we describe how the infinite system $S_\tau(\Sigma)$, describing the dynamics of control systems, can be replaced by a finite system for the purpose of synthesizing embedded control software. We use alternating similarity relationships to relate $S_\tau(\Sigma)$ to its finite abstraction since the abstraction process introduces nondeterminism to be treated as adversarial. The following definition extends alternating similarity, introduced by Alur and coworkers in [AHKV98], to an approximate context that is more appropriate for infinite systems such as $S_\tau(\Sigma)$.

Definition 3. Let S_a and S_b be metric systems with $Y_a = Y_b$ and let $\varepsilon \in \mathbb{R}_0^+$. A relation $R \subseteq X_a \times X_b$ is an *ε-approximate alternating simulation relation* from S_a to S_b if the following three conditions are satisfied:

1. for every $x_{a0} \in X_{a0}$ there exists $x_{b0} \in X_{b0}$ with $(x_{a0}, x_{b0}) \in R$;
2. for every $(x_a, x_b) \in R$ we have $\mathbf{d}(H_a(x_a), H_b(x_b)) \leq \varepsilon$;
3. for every $(x_a, x_b) \in R$ and for every $u_a \in U_a(x_a)$ there exists $u_b \in U_b(x_b)$ such that for every $x'_b \in \mathrm{Post}_{u_b}(x_b)$ there exists $x'_a \in \mathrm{Post}_{u_a}(x_a)$ satisfying $(x'_a, x'_b) \in R$.

We say that S_a is ε-approximately alternatingly simulated by S_b or that S_b ε-approximately alternatingly simulates S_a, denoted by $S_a \preceq^{\varepsilon}_{\mathcal{AS}} S_b$, if there exists an ε-approximate alternating simulation relation from S_a to S_b.

The notion of approximate alternating simulation is asymmetric in the sense that one system simulates while the other is simulated. Symmetrizing approximate alternating simulation leads to the stronger notion of approximate bisimulation where each system both simulates and is simulated.

Definition 4. Let S_a and S_b be metric systems with $Y_a = Y_b$ and let $\varepsilon \in \mathbb{R}_0^+$. A relation $R \subseteq X_a \times X_b$ is an *ε-approximate alternating bisimulation relation* between S_a and S_b if the following two conditions are satisfied:

1. R is an ε-approximate alternating simulation relation from S_a to S_b;
2. R^{-1} is an ε-approximate alternating simulation relation from S_b to S_a.

We say that S_a is ε-approximately alternatingly bisimilar to S_b, denoted by $S_a \cong^{\varepsilon}_{\mathcal{AS}} S_b$, if there exists an ε-approximate alternating bisimulation relation between S_a and S_b.

In the limiting case where $\varepsilon = 0$ we recover the exact notion of alternating (bi)simulation since $\mathbf{d}(H_a(x_a), H_b(x_b)) \leq 0$ implies $H_a(x_a) = H_b(x_b)$. In such case we denote $S_a \preceq^{0}_{\mathcal{AS}} S_b$ and $S_a \cong^{0}_{\mathcal{AS}} S_b$ simply by $S_a \preceq_{\mathcal{AS}} S_b$ and $S_a \cong_{\mathcal{AS}} S_b$, respectively.

4. SOFTWARE DESIGN AS A CONTROLLER SYNTHESIS PROBLEM

Regarding software design as a controller synthesis problem is an idea that has been recently gaining enthusiasts despite having been proposed more than 20 years ago [EC82, MW84]. The starting point is to regard the software to be designed as a system S_{cont} such that the composition $S_{cont} \times S_\tau(\Sigma)$ satisfies the desired specification. If the specification is given as another system S_{spec}, then we seek to synthesize a controller S_{cont} so that:

$$S_{cont} \times S_\tau(\Sigma) \preceq^{\varepsilon}_{\mathcal{AS}} S_{spec},$$

or even:

$$S_{cont} \times S_\tau(\Sigma) \cong^{\varepsilon}_{\mathcal{AS}} S_{spec}.$$

In general, this problem is not solvable algorithmically since $S_\tau(\Sigma)$ is an infinite system. We overcome this difficulty by replacing $S_\tau(\Sigma)$ by a finite abstraction S_{abs} for which we have the guarantee that if a controller satisfying:

$$S_{cont} \times S_{abs} \preceq^{\varepsilon}_{\mathcal{AS}} S_{spec}$$

exists then a controller S'_{cont} satisfying:

$$S'_{cont} \times S_\tau(\Sigma) \preceq^{\varepsilon}_{\mathcal{AS}} S_{spec}$$

also exists. We call S'_{cont} the refinement of S_{cont}. It is shown in [Tab09] that existence of an approximate alternating simulation relation from S_{abs} to $S_\tau(\Sigma)$ is sufficient to refine the controller S_{cont} acting on S_{abs} to the controller S'_{cont} acting on $S_\tau(\Sigma)$. If we can also establish the existence of an approximate alternating bisimulation relation between S_{abs} and $S_\tau(\Sigma)$, then we have the guarantee that if a controller exists for $S_\tau(\Sigma)$, a controller also exists for S_{abs}. Hence, this design flow is not only sound but also complete. Moreover, since S'_{cont} admits a finite description, it can be directly compiled into code executable in a digital platform. The computation of the finite abstraction S_{abs} is one of the problems solved by Pessoa as we describe in Section 6. Several examples attesting the effectiveness of Pessoa are presented in Section 7.

5. EXISTENCE OF SYMBOLIC MODELS FOR LINEAR CONTROL SYSTEMS

In Section 3 we saw that given a control system Σ and a desired time quantization τ we can represent the τ-discretized version of Σ as the system $S_\tau(\Sigma)$. Although $S_\tau(\Sigma)$ is not finite, it was shown in [PGT08] that when Σ is a linear system whose A matrix has eigenvalues with negative real-part, $S_\tau(\Sigma)$ can be related to a finite system by an approximate alternating bisimulation relation. More recent results [ZPJT09] showed that even when the eigenvalues of A have positive real-part, $S_\tau(\Sigma)$ can still be related to a finite system. In the later case, only an approximate alternating simulation relation can be established.

The abstraction process studied in [PGT08] requires a time quantization parameter $\tau \in \mathbb{R}^+$, a space quantization parameter $\eta \in \mathbb{R}^+$, and an input quantization parameter $\mu \in \mathbb{R}^+$. These parameters are then used to discretize Σ into $S_{\tau\eta\mu}(\Sigma)$.

Definition 5. The system

$$S_{\tau\eta\mu}(\Sigma) = (X_{\tau\eta\mu}, X_{\tau\eta\mu 0}, U_{\tau\eta\mu}, \xrightarrow[\tau\eta\mu]{}, Y_{\tau\eta\mu}, H_{\tau\eta\mu})$$

associated with a linear control system $\Sigma = (\mathbb{R}^n, \mathsf{U}, \mathcal{U}, A, B)$ and with $\tau, \eta, \mu \in \mathbb{R}^+$ consists of:

- $X_{\tau\eta\mu} = [\mathbb{R}^n]_\eta$;
- $X_{\tau\eta\mu 0} = X_{\tau\eta\mu}$
- $U_{\tau\eta\mu} = \{v \in \mathcal{U} \mid v(t) = v(t') \in [\mathsf{U}]_\mu \quad \forall t, t' \in [0, \tau] = \operatorname{dom} v\}$;
- $x \xrightarrow[\tau\eta\mu]{v} x'$ if there exist $v \in U$, and a trajectory $\xi_{xv} : [0, \tau] \to \mathbb{R}^n$ of Σ satisfying $\|\xi_{xv}(\tau) - x'\| \le \frac{\eta}{2}$;
- $Y_{\tau\eta\mu} = \mathbb{R}^n$;
- $H_{\tau\eta\mu} = \imath : X_{\tau\eta} \hookrightarrow \mathbb{R}^n$.

Note that the output set $Y_{\tau\eta\mu}$ is naturally equipped with the norm-induced metric $\mathbf{d}(y, y') = \|y - y'\|$.

Each transition of $S_{\tau\eta\mu}(\Sigma)$ describes the evolution of ξ up to an error of $\frac{\eta}{2}$. Hence, one might wonder to which extent can such approximation error be tolerated once several consecutive transitions are taken. The next result shows how η and μ can be chosen so as to guarantee that $S_{\tau\eta\mu}(\Sigma)$ is ε-approximately alternatingly bisimilar to $S_\tau(\Sigma)$.

Theorem 1 ([PGT08]). *Let Σ be a linear control system in which all the eigenvalues of the matrix A have negative real-part and $\mathcal{U}$ consists of piece-wise constant curves. For any desired precision $\varepsilon \in \mathbb{R}^+$, time quantization $\tau \in \mathbb{R}^+$, input quantization $\mu \in \mathbb{R}^+$, and for any space quantization $\eta \in \mathbb{R}^+$ satisfying:*

$$\left\| e^{A\tau} \right\| \varepsilon + \int_0^\tau \left\| e^{At} B \right\| dt \, \frac{\mu}{2} + \frac{\eta}{2} \le \varepsilon \tag{2}$$

the following holds:

$$S_{\tau\eta\mu}(\Sigma) \cong^{\varepsilon}_{\mathcal{AS}} S_\tau(\Sigma). \tag{3}$$

We note that $S_{\tau\eta\mu}(\Sigma)$ becomes a finite system once we restrict attention to a compact subset of the states of $S_\tau(\Sigma)$. We shall return to this observation once we review the results in [ZPJT09] that can be used when A has eigenvalues with positive real-part. We first recall that we denote by $\operatorname{Post}_u(x)$ the set of all the states of $S_\tau(\Sigma)$ that are u-successors of x. We shall abuse notation and denote by $\operatorname{Post}_u(\mathcal{B}_{\frac{\eta}{2}}(x))$ the set:

$$\operatorname{Post}_u(\mathcal{B}_{\frac{\eta}{2}}(x)) = \bigcup_{x' \in \mathcal{B}_{\frac{\eta}{2}}(x)} \operatorname{Post}_u(x').$$

Using Post we can construct an abstraction different from the one in Definition 5.

Definition 6. The system

$$S_{\tau\eta} = (X_{\tau\eta}, X_{\tau\eta 0}, U_{\tau\eta}, \xrightarrow[\tau\eta]{}, Y_{\tau\eta}, H_{\tau\eta})$$

associated with a linear control system $\Sigma = (\mathbb{R}^n, \mathsf{U}, \mathcal{U}, A, B)$ and with $\tau, \eta \in \mathbb{R}^+$ consists of:

- $X_{\tau\eta} = [\mathbb{R}^n]_\eta$;
- $X_{\tau\eta 0} = X_{\tau\eta}$
- $U_{\tau\eta} = \mathcal{U}$;
- $x \xrightarrow[\tau\eta]{v} x'$ if there exist $v \in U$ satisfying: $\operatorname{int}(\operatorname{Post}_v(\mathcal{B}_{\frac{\eta}{2}}(x)) \cap \mathcal{B}_{\frac{\eta}{2}}(x')) \ne \varnothing$;
- $Y_{\tau\eta} = \mathbb{R}^n$;
- $H_{\tau\eta} = \imath : X_{\tau\eta} \hookrightarrow \mathbb{R}^n$.

Note that the output set $Y_{\tau\eta}$ is naturally equipped with the norm-induced metric $\mathbf{d}(y, y') = \|y - y'\|$.

Whenever the set U has finite cardinality, the abstract model introduced in Definition 6 is an abstraction of $S_\tau(\Sigma)$ in the following sense.

Theorem 2 ([ZPJT09, JT09]). *Let Σ be any linear control system with $\mathcal{U}$ consisting of piece-wise constant curves and U having finite cardinality. For any desired precision $\varepsilon \in \mathbb{R}^+$, time quantization $\tau \in \mathbb{R}^+$, and for any space quantization $\eta \in \mathbb{R}^+$ satisfying:*

$$\eta \le 2\varepsilon \tag{4}$$

the following holds:

$$S_{\tau\eta}(\Sigma) \preceq^{\varepsilon}_{\mathcal{AS}} S_\tau(\Sigma). \tag{5}$$

As shown in [Tab09], existence of an approximate alternating simulation relation from $S_{\tau\eta\mu}(\Sigma)$ to $S_\tau(\Sigma)$ implies that any controller acting on $S_{\tau\eta\mu}(\Sigma)$ can be refined to a controller acting on $S_\tau(\Sigma)$ and enforcing the same specification. However, when a controller enforcing the desired specifications on $S_\tau(\Sigma)$ exists, there is no guarantee that it can be found by working with the abstraction $S_{\tau\eta}(\Sigma)$. Therefore, there is no loss of generality in assuming that $\mathcal{U}$ has already been quantized, *i.e.*, that U has finite cardinality. For this reason, the parameter μ does not play a role in the assumptions of Theorem 2.

Remark 1. *The conclusions of Theorem 2 remain valid if instead of $\operatorname{Post}_u(\mathcal{B}_{\frac{\eta}{2}}(x))$ we use any over-approximation for this set. This is crucial for nonlinear systems and useful for linear systems since over-approximations can be computed much faster than $\operatorname{Post}_u(\mathcal{B}_{\frac{\eta}{2}}(x))$.*

The symbolic models in Theorems 1 and 2 have countably infinite state sets. However, in practical applications the physical variables are restricted to a compact set. Velocities, temperatures, pressures, and other physical quantities cannot become arbitrarily large without violating the operational envelop defined by the control problem being solved. By making use of this fact, $S_{\tau\eta\mu}(\Sigma)$ and $S_{\tau\eta}(\Sigma)$ can be regarded as finite systems. To simplify the discussion in this paragraph, we will use $S_\bullet(\Sigma) = (X_\bullet, X_{\bullet 0}, U_\bullet, \xrightarrow[\bullet]{}, Y_\bullet, H_\bullet)$ to refer to both $S_{\tau\eta\mu}(\Sigma)$ and $S_{\tau\eta}(\Sigma)$. The first observation is that we can encode the operational envelop on the output map of $S_\bullet(\Sigma)$. We thus consider a compact set $D \subset \mathbb{R}^n$ and redefine the output set of $S_\bullet(\Sigma)$ to $Y_\bullet = D \cup \{*\}$ for some element $*$ not belonging to D. The symbol $*$ represents all the states that are "out of bounds" or "out of sensor range". The output map of $S_\bullet(\Sigma)$ is also redefined to:

$$H_\bullet(x) = \begin{cases} x & \text{if } x \in X \cap D \\ * & \text{if } x \notin X \cap D \end{cases}$$

The new output set is equipped with the metric:

$$\mathbf{d}(y, y') = \begin{cases} \frac{1}{2}\text{diam}(D) & \text{if} \quad y' = * \text{ and } y \in D \\ & \qquad \text{or } y = * \text{ and } y' \in D \\ 0 & \text{if} \quad y = * = y' \\ \|y - y'\| & \text{if} \quad y, y' \in D \end{cases}.$$

Although the redefined system $S_\bullet(\Sigma)$ is still countably infinite, it 0-approximately alternatingly simulates the finite system $S_{abs} = (X_{abs}, X_{abs0}, U_{abs}, \xrightarrow[abs]{}, Y_{abs}, H_{abs})$ consisting of:

- $X_{abs} = [D]_\eta \cup \{*\}$;
- $X_{abs0} = X_{abs} \cap X_{\bullet 0}$;
- $U_{abs} = U_\bullet$;
- $x \xrightarrow[abs]{u} x'$ in S_{abs} if $x, x' \in [D]_\eta$ and $x \xrightarrow[\bullet]{u} x'$ in $S_\bullet(\Sigma)$ or if $x \in [D]_\eta$, $x' = *$, and $x \xrightarrow[\bullet]{u} x''$ in $S_\bullet(\Sigma)$ with $x'' \in X_\bullet \backslash [D]_\eta$;
- $Y_{abs} = Y_\bullet$;
- $H_{abs} = 1_{X_{abs}}$.

The relation $R \subseteq X_{abs} \times X_\bullet$ defined by $(x_{abs}, x_\bullet) \in R$ if $x_{abs} = x_\bullet \in [D]_\eta$ or $x_{abs} = *$ and $x' \in X \backslash [D]_\eta$ is a 0-approximate alternating simulation relation from S_{abs} to $S_\bullet(\Sigma)$. Finiteness of S_{abs} now follows from compactness of D. Intuitively, S_{abs} is not more than the restriction of $S_\bullet(\Sigma)$ to the set D. For this reason, we implicitly assume that all the specifications that we are interested in contain the requirement that no trajectory should ever leave the set D, even if this is not explicitly stated.

6. INTRODUCING PESSOA

Pessoa[3] is a toolbox automating the synthesis of correct-by-design embedded control software. Although the core algorithms in Pessoa have been coded in C, the main functionalities are available through the Matlab command line. All the systems and sets manipulated by Pessoa are represented symbolically using Reduced Ordered Binary Decision Diagrams (ROBDDs) supported by the CUDD library [CUD].

[3]Pessoa Version 1.0 can be freely downloaded from `http://www.cyphylab.ee.ucla.edu/Pessoa/`.

Pessoa Version 1.0 offers three main functionalities:

1. the construction of finite symbolic models of linear[4] control systems;
2. the synthesis of symbolic controllers for simple specifications;
3. simulation of the closed-loop behavior in Simulink.

Each one of these functionalities is described in more detail in the following sections.

6.1 Computing symbolic models in PESSOA

Linearity of the control system being abstracted is exploited by Pessoa in different ways to simplify the computations. In particular, we make use of the variation of constants formula, *i.e.*, given a state $x \in X$ and a constant input $\upsilon \in \mathcal{U}$, the υ-successor of x in $S_\tau(\Sigma)$, given by $\xi_{x\upsilon}(\tau)$, can be computed as:

$$\xi_{x\upsilon}(\tau) = e^{A\tau}x + \int_0^\tau e^{A(\tau - t)}B\upsilon(t)dt.$$

We can thus express $\text{Post}_\upsilon(\mathcal{B}_{\frac{\eta}{2}}(x))$ as:

$$\text{Post}_\upsilon(\mathcal{B}_{\frac{\eta}{2}}(x)) = A_\tau(\mathcal{B}_{\frac{\eta}{2}}(x)) \oplus \{B_\tau \upsilon\}$$

where the matrices A_τ and B_τ are defined by:

$$A_\tau = e^{A\tau}, \quad B_\tau = \int_0^\tau e^{A(\tau - t)}Bdt.$$

The closed ball $\mathcal{B}_{\frac{\eta}{2}}(x)$ can be written as:

$$\mathcal{B}_{\frac{\eta}{2}}(x) = \{x\} \oplus \mathcal{B}_{\frac{\eta}{2}}(0)$$

and leads to the decomposition:

$$\text{Post}_\upsilon(\mathcal{B}_{\frac{\eta}{2}}(x)) = \{A_\tau x\} \oplus A_\tau(\mathcal{B}_{\frac{\eta}{2}}(0)) \oplus \{B_\tau \upsilon\}.$$

Note that the second and third terms can be computed only once, when evaluating $\text{Post}_u(\mathcal{B}_{\frac{\eta}{2}}(x))$ at the states $x \in X_{\tau\eta}$, since they do not depend on x. To speedup the computations further, the set $A_\tau(\mathcal{B}_{\frac{\eta}{2}}(0))$ is not computed exactly, but rather over-approximated as a union of hyper-rectangles commensurable with η. Despite this approximation, we still obtain an abstraction satisfying (2) as explained in Remark 1. The abstraction $S_{\tau\eta\mu}(\Sigma)$ introduced in Definition 5 does not require over-approximations since $\xi_{x\upsilon}(x)$ is readily computed as $A_\tau x + B_\tau \upsilon$.

The transition relations $\longrightarrow$ of $S_{\tau\eta\mu}(\Sigma)$ and $S_{\tau\eta}(\Sigma)$ are encoded in a ROBDD through the corresponding characteristic functions, *i.e.*, we encode the binary function:

$$T : X \times U \times X \to \{0, 1\}$$

satisfying $T(x, u, x') = 1$ iff $(x, u, x') \in \longrightarrow$. To speed up the computation of the ROBDD describing the function T, we first perform a change of coordinates taking $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$ to $\overline{X} \subseteq \mathbb{Z}^n$ and $\overline{U} \subseteq \mathbb{Z}^m$. In this manner we use the unsigned integer variables to encode the states and inputs, and to perform all the computations.

[4]Linear control systems are natively supported in Pessoa Version 1.0. Nonlinear and switched systems can also be handled by Pessoa but require some additional effort by the user. For further information please consult the documentation in `http://www.cyphylab.ee.ucla.edu/Pessoa`.

6.2 Synthesizing symbolic controllers in PESSOA

Pessoa currently supports the synthesis of controllers enforcing four[5] kinds of specifications defined using a target set $Z \subseteq X$ and a constraint set $W \subseteq X$:

1. **Stay:** trajectories start in the target set Z and remain in Z. This specification corresponds to the Linear Temporal Logic (LTL) formula[6] $\Box\varphi_Z$ where φ_Z is the predicate defining the set Z;
2. **Reach:** trajectories enter the target set Z in finite time. This specification corresponds to the LTL formula $\Diamond\varphi_Z$;
3. **Reach and Stay:** trajectories enter the target set Z in finite time and remain within Z thereafter. This specification corresponds to the LTL formula $\Diamond\Box\varphi_Z$;
4. **Reach and Stay while Stay:** trajectories enter the target set Z in finite time and remain within Z thereafter while always remaining within the constraint set W. This specification corresponds to the LTL formula $\Diamond\Box\varphi_Z \wedge \Box\varphi_W$ where φ_W is the predicate defining the set W.

Although simple, the above specifications already allow Pessoa to solve nontrivial synthesis problems as described in Section 7. Reach and stay specifications can be used to encode usual set regulation problems where the state is to be steered to a desired operating point set and forced to remain there. The fourth kind of specification complements reach and stay requirements by imposing state constraints, defined by the set W, that are to be enforced for all time.

The controllers for the above specifications are memoryless controllers that can be synthesized through fixed point computations as described in [Tab09]. All the fixed-points are computed symbolically using the ROBDD representation of the abstractions $S_{\tau\eta\mu}(\Sigma)$ or $S_{\tau\eta}(\Sigma)$, and a ROBDD representation for the sets Z and W. These sets can be specified as hyper-rectangles, by providing the corresponding vertices, or as arbitrary sets, by providing the corresponding characteristic functions. The finite state nature of the synthesized controllers permits a direct compilation into code. Although code generation is not yet supported in Version 1.0 of Pessoa, closed-loop simulation in Simulink is already available.

6.3 Simulating the closed-loop in Simulink

Pessoa also provides the possibility to simulate the closed-loop behavior in Simulink. For this purpose, Pessoa comes with a Simulink block implementing a refinement of any synthesized controller (see Figure 2). The controllers synthesized in Pessoa are, in general, nondeterministic. The Simulink block resolves this nondeterminism in a consistent fashion thus providing repeatable simulations. In order to increase the simulation speed, the Simulink block selects, among all the inputs available for the current state, the input with the shortest description in the ROBDD encoding the controller. Moreover, the input is chosen in a lazy manner, *i.e.*, the input is only changed when the previously used input cannot be used again. Other determinization strategies, such as minimum energy inputs, will be supported in future versions of Pessoa.

[5]Future versions of Pessoa will handle specifications given as linear temporal logic formulas or automata on infinite strings.

[6]The semantics of LTL would be defined in the usual manner over the output behaviors of $S_\tau(\Sigma)$.

Parameter	Value	Description
R	500×10^{-3}	Resistance
L	1500×10^{-6}	Inductance
J	250×10^{-6}	Moment of inertia
B	100×10^{-6}	Viscous friction coefficient
k	50×10^{-3}	Torque constant

Table 1: Parameters for the circuit in Figure 1 expressed in the international system of units.

7. EXAMPLES

All the computations for the examples were conducted on a MacBook Pro with a 2.26 GHz Intel Core 2 Duo processor and 2GB of memory.

7.1 DC Motor

The first example can be found in most undergraduate control textbooks and consists in regulating the velocity of a DC motor. The electric circuit driving the DC motor is shown in Figure 1. The dynamics Σ of this system comprises two linear differential equations:

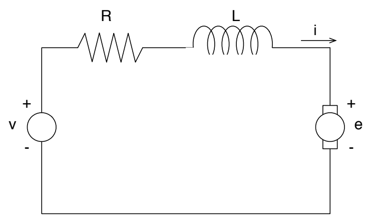

Figure 1: DC motor and associated electric circuit.

$$\dot{x}_1 = -\frac{B}{J}x_1 + \frac{k}{J}x_2 \quad (6)$$

$$\dot{x}_2 = -\frac{k}{L}x_1 - \frac{R}{L}x_2 + \frac{1}{L}u. \quad (7)$$

The variable x_1 describes the angular velocity of the motor, the variable x_2 describes the current i through the inductor, and the variable u represents the source voltage v that is treated as an input. The model parameters are shown in Table 1.

The control objective is to regulate the velocity around 20 rad/s. We select the domain D for the symbolic model to be:

$$D = [-1, 30] \times [-10, 10].$$

The input space is $\mathsf{U} = [-10, 10]$ and the quantization parameters are given by $\tau = 0.05$, $\eta = 0.5$, and $\mu = 0.01$. These quantization parameters were chosen so as to satisfy inequality (2) in Theorem 1 with $\varepsilon = 1$. Since the objective is to regulate the velocity to a desired set point, we consider the target set:

$$Z = [19.5, 20.5] \times [-10, 10]$$

constraining the velocity to a neighborhood of the desired set point and chose a "reach and stay" specification in Pessoa. The symbolic abstraction was computed in 18 minutes while the symbolic controller took less than one second to be synthesized. The closed loop behavior was simulated in Simulink using the symbolic controller block included in Pessoa and represented in Figure 2. The evolution of the velocity and input are displayed in Figure 3 for the initial condition $(x_1, x_2) = (0, 0)$.

In practical implementations the DC motor is connected to a constant voltage source through an H-bridge. By opening and closing the switches in the H-bridge we can only chose three different values for the voltage: -10V, 0V, and 10V. In order to synthesize a controller under these input constraints we redefine the input quantization to $\mu = 10$. This guarantees that u can only assume the desired three voltage levels. Velocity regulation now requires more frequent changes to the input voltage. Hence, we change the time quantization to $\tau = 0.0001$ and also the space quantization $\eta = 0.05$ so that we can capture the changes that occur during each sampling period of 0.0001 seconds. These quantization parameters no longer satisfy inequality (2) and settle for a symbolic abstraction related to $S_\tau(\Sigma)$ by an approximate alternating simulation. The abstraction is computed in 17 minutes and the controller synthesized in 108 seconds.

The time evolution of the velocity and current are obtained by simulating the closed-loop system with the new controller and can be seen in Figure 4. Although the velocity converges to a small neighborhood of 20 rad/s (see Figure 5), the values of the current through the inductor are quite large, attaining a peak of 10 Amperes. This can be improved by redefining the target set to:

$$Z = [19.5, 20.5] \times [-0.7, 0.7]$$

so as to reduce the current ripple to 0.7 Amperes around 0, and by introducing the constraint set W:

$$W = [-1, 30] \times [-3, 3]$$

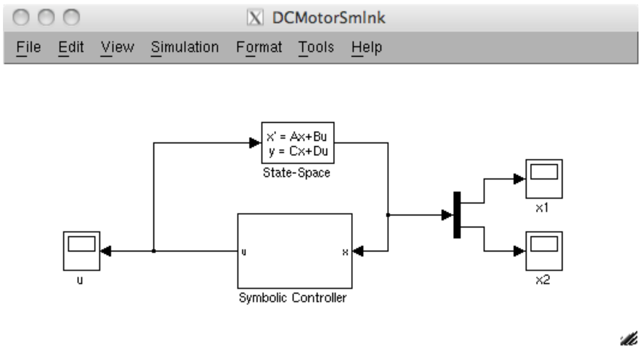

Figure 2: Simulink diagram for the closed-loop system depicting the symbolic controller block included in Pessoa.

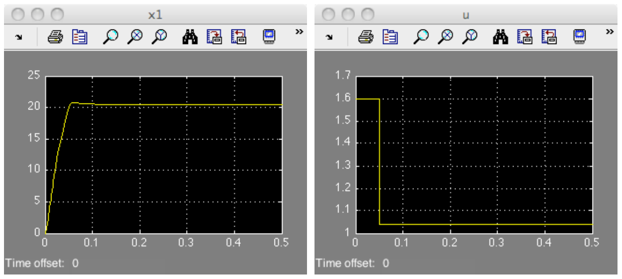

Figure 3: Evolution of velocity and input voltage for the DC motor example.

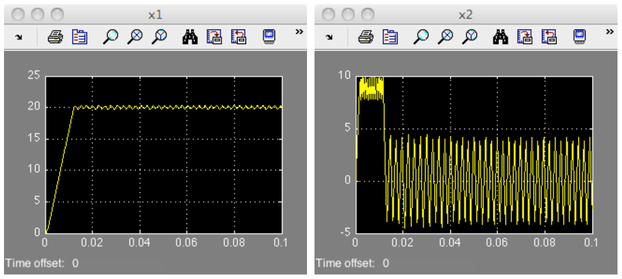

Figure 4: Evolution of velocity and current when the input voltage is restricted to -10, 0, and 10 Volts.

to limit the peak current to 3 Amperes. We synthesize a new controller enforcing the "reach and stay while stay" in 88 seconds. The closed-loop simulation results in Figure 6 show that the target set is still reached while the current ripple and peak values have been reduced to conform to the new target set and constraint set. Note how the peak current limitation forces slows the convergence to the target set Z.

7.2 Motion planning with obstacle avoidance

In this section we revisit another classical problem, motion planning with obstacle avoidance. We consider the usual chained form model Σ for a unicycle type robot:

$$\begin{aligned}\dot{x}_1 &= u_1\\ \dot{x}_2 &= u_2\\ \dot{x}_1 &= x_2 u_1.\end{aligned}$$

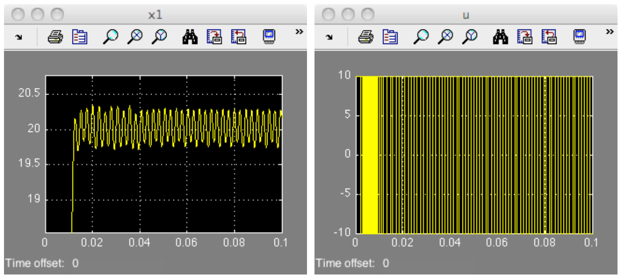

Figure 5: Evolution of velocity and input when the input voltage is restricted to -10, 0, and 10 Volts.

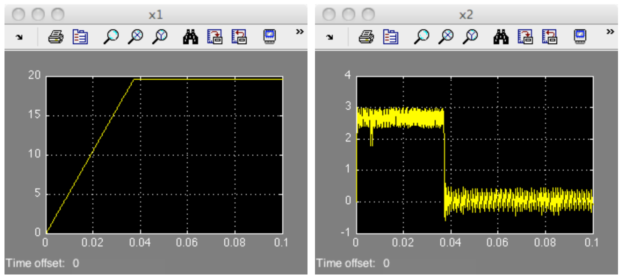

Figure 6: Evolution of velocity and current when the input voltage is restricted to -10, 0, and 10 Volts and state constraints are enforced.

Although Pessoa does not support nonlinear systems natively, nonlinear dynamics can be handled by providing Pessoa with a Matlab function describing an over-approximation of $\mathrm{Post}_u(\mathcal{B}_{\frac{\eta}{2}}(x))$ for $S_\tau(\Sigma)$. The motion planning problem with obstacle avoidance asks for a control law steering the robot from a desired initial position to a desired target set while avoiding obstacles. We work on the compact set $D = [1,4]\times[-2,2]\times[1,4] \subset \mathbb{R}^3$ and set $\mathsf{U} = [-1,1] \times [-1,1]$. The quantization parameters are $\tau = 0.5$, $\eta = 0.1$, and $\mu = 1$. This system fails all the known conditions for the existence of approximate bisimulations [PGT08, Tab09] but satisfies a nonlinear version of Theorem 2 proved in [JT09]. Therefore, we can use Pessoa to construct a symbolic model related to $S_\tau(\Sigma)$ by an ε-approximate alternating simulation relation with $\varepsilon = \frac{\eta}{2} = 0.05$. The target set to be reached by the robot is $[3.35, 3.65] \times [-2, 2] \times [2.35, 2.65]$. Since the controller to be synthesized only guarantees that the target set will be reached up to an error of ε, we reduce the desired target set to:

$$Z = [3.4, 3.6] \times [-2, 2] \times [2.4, 2.6]. \qquad (8)$$

The obstacle is defined by the set $[2.3, 2.7]\times[-2, 2]\times[1.6, 3.4]$. Once again, due to the approximation precision, we pass the following set to Pessoa as the constraint set:

$$W = \{x \in D \mid x \notin [2.25, 2.75] \times [-2, 2] \times [1.55, 3.45]\}. \quad (9)$$

The symbolic model is computed by Pessoa in 205 seconds and a controller enforcing the specification "reach and stay while stay" is computed in 12 seconds. The resulting closed-loop behavior is shown in Figure 7 for different initial conditions.

7.3 Control with shared actuators

The last example addresses the problem of controller synthesis under shared resources. We consider a control system that has permanent access to a low quality actuator and sporadic access to a high quality actuator. This scenario arises when the high quality actuator is connected to the controller through a shared network, or consumes large amounts of energy drawn from a shared batery. Moreover, we also assume that we do not have at our disposal a model for the other software tasks competing for the shared resources. This is typically the case when such software tasks are being concurrently designed. However, even if we had models for these software tasks, the complexity of synthesizing the control software using these models would be prohibitive. Therefore, we shall impose a simple fairness requirement mediating the access to the shared resources.

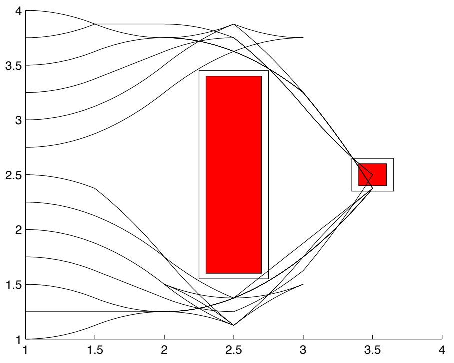

Figure 7: Trajectories of a unicycle robot reaching the target set while avoiding the obstacle. Also shown is the under-approximation of the target set and the over-approximation of the obstacle.

To make the ensuing discussion concrete, we assume that three tasks can have access to the shared resources, one of them being the control task. We use the expression time slot to refer to time intervals of the form $[k\tau, (k+1)\tau[$ with $k \in \mathbb{N}$ and where τ is the time quantization parameter. If we consider sequences of three consecutive time slots, the fairness requirement imposes the availability of the actuator in least one time slot. Possible availability sequences satisfying this assumption are:

```
|aaa|aaa|aaa|aaa|aaa|aaa|aaa|aaa|aaa|...
|aua|uau|aua|uau|aua|uau|aua|uau|aua|...
|aau|aau|aau|aau|aau|aau|aau|aau|aau|...
|uaa|uau|auu|uua|uua|auu|uua|uaa|aua|...
|uaa|uau|auu|uua|uua|auu|uua|uaa|aua|...
|uau|uau|uau|aua|uaa|uau|auu|aaa|aaa|...
```

where we denoted by $\mathtt{a}$ the availability of the resources, by $\mathtt{u}$ the unavailability, and separated the sequences of three time slots with the symbol $|$. Since the preceding sequences form an ω-regular language they can be described by the automaton represented in Figure 8. The system Σ to be controlled

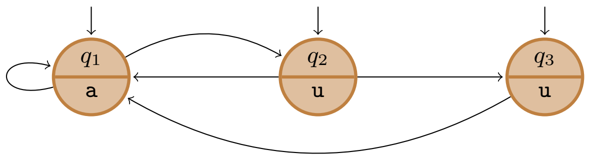

Figure 8: Automaton describing the availability of the shared resources. The lower part of the states is labeled with the outputs a and u denoting availability and unavailability of the shared resource, respectively.

is a double integrator:

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= u_{low} + u_{high}.\end{aligned}$$

where u_{low} denotes the input produced by the low quality actuator and u_{high} denotes the input produced by the high quality actuator. Any of the actuators generates piecewise constant inputs taking values in $\mathsf{U} = [-1, 1]$. However, when an input $u \in \mathsf{U}$ is requested to the low quality actuator, the actual generated input u_{low} is an element of the set $[u - 0.6, u + 0.6]$. In contrast, the high quality actuator always produces the input that is requested, *i.e.*, $u_{high} = u$. The control objective is to force the trajectories to remain within the target set $Z = [-1, 1] \times [-1, 1]$. The fairness constraint is also a control objective that can be expressed by resorting to a model for the concurrent execution of $S_\tau(\Sigma)$ and the automaton in Figure 8. When the automaton is in state q_1, any of the actuators can be used. However, when the automaton is in the state q_2 or q_3 only the low quality actuator can be used. Although this kind of specification is not natively supported in Pessoa, it can be handled by providing Pessoa with a Matlab file containing an operational model for the concurrent execution of $S_\tau(\Sigma)$ and the automaton in Figure 8. Choosing $D = [-1, 1] \times [-1, 1]$ as the domain of the symbolic abstraction, and $\tau = 0.1$, $\eta = 0.05$, and $\mu = 0.5$ as quantization parameters, Pessoa computes the symbolic abstraction in 109 seconds and synthesizes a controller in 2 seconds. The domain of the controller is shown in Figure 9 and two typical closed-loop behaviors are shown in Figures 10, 11, and 12. We can appreciate the controller forcing the trajectories to stay within the target set despite the low quality of the permanently available actuator. We note that if we require the high quality actuator to be permanently unavailable, Pessoa reports the non-existence of a solution.

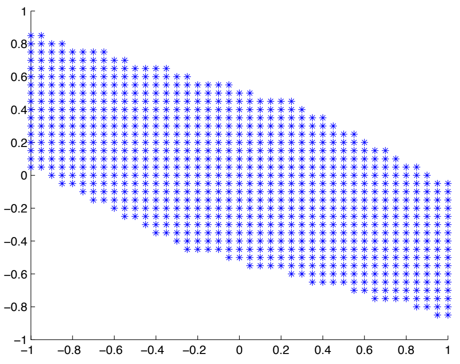

Figure 9: Domain of the controller forcing the double integrator to remain in $[-1, 1] \times [-1, 1]$ under the fairness constraints described by the automaton in Figure 8.

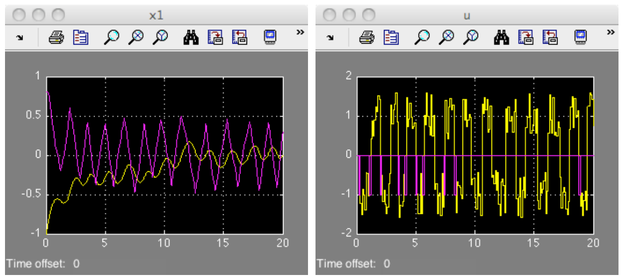

Figure 10: Evolution of the state variables (left figure) and inputs (right figure), from initial state $(x_1, x_2) = (-1, 0.8)$, when the automaton in Figure 8 is visiting the states $|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|\ldots$. The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta.

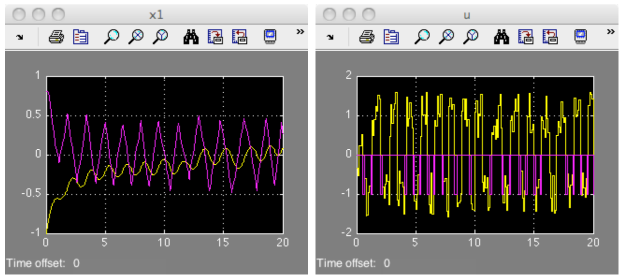

Figure 11: Evolution of the state variables (left figure) and inputs (right figure), from initial state $(x_1, x_2) = (-1, 0.8)$, when the automaton in Figure 8 is visiting the states $|q_1q_2q_1|q_2q_1q_2|q_1q_2q_1|q_2q_1q_2|q_1q_2q_1|q_2q_1q_2|\ldots$. The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta.

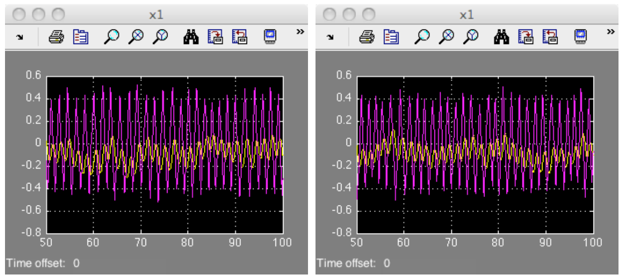

Figure 12: Evolution of the state variables. The left figure refers to the initial states and automaton evolution in Figure 10 while the right figure refers to the initial states and automaton evolution in Figure 11.

8. EXTENDING PESSOA

Pessoa is currently being extended in the following directions:

- Nonlinear and switched dynamics can already be used in Pessoa, albeit not natively. Future versions of Pessoa will provide native support for these classes of systems;
- Specifications with discrete memory can be used with Pessoa be encoding them in the plant dynamics as reported in Section 7.3. Future versions of Pessoa will natively support specifications given in LTL and automata on infinite strings;
- The state set of the abstractions computed by Pessoa is a grid resolution η. However, Theorem 2 and its more general version reported in [ZPJT09, JT09] do not require the use of a grid of constant resolution. We are currently working on extending Pessoa to multi-resolution grids with the objective of reducing the size of the computed abstractions and controllers.
- We are also extending Pessoa to support quantitative control objectives. Preliminary steps in this direction addressing the synthesis of time-optimal controllers are reported in [JT09].

9. CONCLUSIONS

In this paper we described a concrete approach to the synthesis of correct-by-design embedded control software. This approach is supported by the tool Pessoa that can be used to compute finite abstractions of continuous control systems, synthesize discrete controllers, and refine the synthesized controllers to Simulink blocks. The tool was illustrated on several examples. Pessoa is currently being improved by extending its scope as described in Section 8 and is freely available at: `http://www.cyphylab.ee.ucla.edu/Pessoa/`.

10. REFERENCES

[AHKV98] R. Alur, T. Henzinger, O. Kupferman, and M. Vardi. Alternating refinement relations. In *Proceedings of the 8th International Conference on Concurrence Theory*, number 1466 in Lecture Notes in Computer Science, pages 163–178. Springer, 1998.

[Ari] Ariadne: An open tool for hybrid system analysis. Electronically available at: `http://trac.parades.rm.cnr.it/ariadne/`.

[AVW03] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 28(1):7–34, 2003.

[Che] Checkmate: Hybrid system verification toolbox for matlab. Electronically available at: `http://www.ece.cmu.edu/~webk/checkmate/`.

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

[CUD] CUDD: CU Decision Diagram Package. Electronically available at: `http://vlsi.colorado.edu/~fabio/CUDD/`.

[dAHM01] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR 01: Concurrency Theory, 12th International Conference*, number 2154 in Lecture Notes in Computer Science, 2001.

[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.

[Hyba] Hybridsal. Electronically available at: `http://sal.csl.sri.com/hybridsal/`.

[Hybb] Hybrid Toolbox. Electronically available at: `http://www.dii.unisi.it/hybrid/toolbox`.

[JT09] Manuel Mazo Jr. and Paulo Tabuada. Approximate time-optimal control via approximate alternating simulations. 2009. Submitted for publication. Electronically available at: `http://www.cyphylab.ee.ucla.edu/`.

[KeY] Keymaera: A hybrid theorem prover for hybrid systems. Electronically available at: `http://symbolaris.com/info/KeYmaera.html`.

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

[LTL] LTLCon. Electronically available at: `http://iasi.bu.edu/~software/LTL-control.htm`.

[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.

[PGT08] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[PHA] Phaver: Polyhedral hybrid automaton verifyer. Electronically available at: `http://www-artist.imag.fr/~frehse/phaver_web/index.html`.

[Tab09] Paulo Tabuada. *Verification and Control of Hybrid Systems.* Springer, 2009.

[ZPJT09] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. 2009. Submitted for publication. Electronically available at: `http://www.cyphylab.ee.ucla.edu`.