PESSOA: towards the automatic synthesis of correct-by-design control software*

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

Anna Davitian
Aerovironment Inc.
davitian@avinc.com

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

ABSTRACT

In this paper we report on ongoing work aimed at providing tools for the synthesis of *correct-by-design* embedded control software. We introduce a tool, named Pessoa, that: 1) constructs finite abstractions of control systems; 2) synthesizes controllers based on the finite abstractions; 3) refines the controllers to Simulink blocks for closed-loop simulation. In addition to describing the main functionalities of Pessoa, we also illustrated its use in an example where the control software needs to access an actuator shared by other software tasks.

1. INTRODUCTION

With the increasing ubiquity of embedded systems in our daily life, the design of safe and reliable embedded control software becomes a challenge of paramount importance. Such designs are fraught with difficulties arising from the complex interactions between the physical world and the software. In order to ease this problem much work has been devoted to the construction of finite model abstractions for control systems. The use of finite abstractions enable, by resorting to computational tools, the verification of already designed control software, and the automatic synthesis of correct-by-design controllers enforcing predefined specifications for the closed-loop system.

In the present paper we introduce the tool Pessoa, which, based on the notion of *approximate (bi)simulation relations*, automates the synthesis of correct-by-design 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.

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

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]. The construction of abstractions of linear control systems is natively supported in Pessoa Version 1.0, nonlinear and switched systems can also be handled by Pessoa, as illustrated by the example in Section 4, but require some additional effort from the user[1].

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]).

2. SYSTEMS, RELATIONS, AND THE CONTROLLER SYNTHESIS PROBLEM

Let us start introducing the notion of system used in Pessoa 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: existence of a transition $(x, u, x') \in \longrightarrow$ entails that upon the reception of input u at state x, system

[1]For further information please consult the documentation in `http://www.cyphylab.ee.ucla.edu/Pessoa`.

S evolves to state x'. In [Tab09] it is shown how systems of this form can represent both software and control systems modeling physical processes. While software models naturally lead to finite systems, obtaining models for control systems leading to finite systems requires of some abstraction techniques. Informally, a control system Σ is a differential equation of the form $\dot{\xi} = f(\xi, \upsilon)$, where $\xi(t)$ denotes the state of the system at time t, $\upsilon(t)$ the controlled input, and $\dot{\xi}$ denotes the time derivative of ξ. Let us denote by $S_\tau(\Sigma)$ the exact discrete time system resulting from sampling Σ with period τ. By adequately discretizing the state space of the differential equation and the input space, with discretization steps η and μ respectively, and by selecting an adequate sampling time τ, it has been shown in [Tab09] and references therein, that useful finite abstractions S_{abs} for $S_\tau(\Sigma)$ can be obtained. Such finite abstractions can be related to the control system through a generalization of the notion of *alternating simulation relation* named *alternating approximate simulation relation* [Tab09]:

Definition 2. 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.

Symmetrizing approximate alternating simulation leads to the stronger notion of *approximate bisimulation*, denoted by $S_a \cong^\varepsilon_{\mathcal{AS}} S_b$, where each system both simulates and is simulated.

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 S_{cont} 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. Conditions on η, μ and τ to obtain approximate alternating simulation or bisimulation relations between S_{abs} and $S_\tau(\Sigma)$ can be found in [ZPJT09] and references therein.

3. PESSOA FEATURES

Pessoa[2] 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.

Pessoa Version 1.0 offers three main functionalities:

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

All the transition relations $\longrightarrow$ of the abstractions generated by Pessoa, and the sets used in the specifications, are stored as Reduced Order Binary Decision Diagrams (ROBDD)[3] through their corresponding characteristic functions. The usage of ROBDDs enables the efficient computation of reachable sets and pre-images of sets, both fundamental operations in the design of controllers.

Pessoa currently supports the synthesis of controllers enforcing four 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[4] $\Box\varphi_Z$ 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.

[2]Pessoa Version 1.0 can be freely downloaded from `http://www.cyphylab.ee.ucla.edu/Pessoa/`.
[3]ROBDD's supported by the CUDD library [CUD].
[4]The semantics of LTL would be defined in the usual manner over the behaviors of $S_\tau(\Sigma)$.

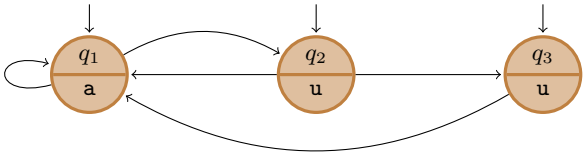

Figure 1: Automaton describing the availability of the shared resources. The lower part of the states is labeled with the outputs `a` and `u` denoting availability and unavailability of the shared resource, respectively.

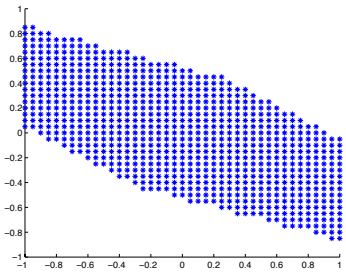

Figure 2: Domain of the controller forcing the double integrator to remain in $[-1,1]\times[-1,1]$ under the fairness constraints described by the automaton in Figure 1.

The controllers for the above specifications are memoryless controllers that can be synthesized through fixed point computations as described in [Tab09]. Furthermore, 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. For this purpose, Pessoa comes with a Simulink block implementing a refinement of any synthesized controller. 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.

4. EXAMPLE

In order to illustrate the capabilities of Pessoa 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

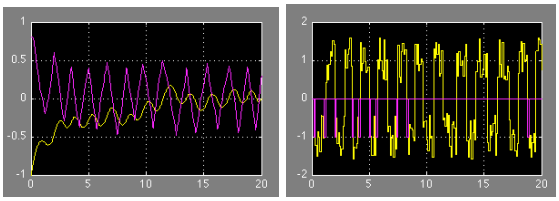

Figure 3: Evolution of the state variables (left figure) and inputs (right figure), from initial state $(x_1,x_2)=(-1,0.8)$, when the automaton in Figure 1 is visiting the states $|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|q_2q_3q_1|\ldots$. The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta.

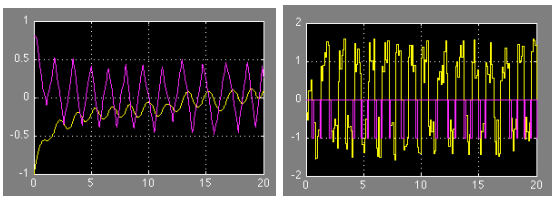

Figure 4: Evolution of the state variables (left figure) and inputs (right figure), from initial state $(x_1,x_2)=(-1,0.8)$, when the automaton in Figure 1 is visiting the states $|q_1q_2q_1|q_2q_1q_2|q_1q_2q_1|q_2q_1q_2|q_2q_1q_2|\ldots$. The input resulting from the low quality actuator is displayed in yellow while the input resulting from the high quality actuator is represented in magenta.

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.

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 at least one time slot. Some availability sequences satisfying this assumption are:
`|uaa|uau|auu|uua|uua|auu|uua|uaa|aua|...`
`|uau|uau|uau|aua|uaa|uau|auu|aaa|aaa|...`
where we denoted by `a` the availability of the resources, by `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 1.

The system Σ to be controlled 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 $\mathsf{U} = \{-1, -0.5, 0, 0.5, 1\}$. 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 1. 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 1. 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 (on a MacBook Pro with a 2.26 GHz Intel Core 2 Duo processor and 2GB of memory). The domain of the controller is shown in Figure 2 and two typical closed-loop behaviors are shown in Figures 3, and 4. 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.

5. 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 by encoding them in the plant dynamics as briefly reported in Section 4. 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, the results reported in [ZPJT09, JT10] 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 [JT10].

6. REFERENCES

[Ari] Ariadne: An open tool for hybrid system analysis. 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. 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. 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. Available at: `http://sal.csl.sri.com/hybridsal/`.

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

[JT10] Manuel Mazo Jr. and Paulo Tabuada. Approximate time-optimal control via approximate alternating simulations. 2010. To appear at the American Control Conference 2010. Available at: `http://www.cyphylab.ee.ucla.edu/`.

[KeY] Keymaera: A hybrid theorem prover for hybrid systems. 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. 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. 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. Available at: `http://www.cyphylab.ee.ucla.edu`.