Formal Methods for the Control of Large-scale Networked Nonlinear Systems with Logic Specifications

Basilica di Santa Maria di Collemaggio, 1287, L'Aquila

Lecture L14: Tools

Speaker: Alessandro Borri

Some time ago: Hybrid Systems Tools (2000–2010)

Hybrid Systems exhibit continuous and discrete dynamic behavior

Hybrid Systems Verification tools focus on automatically proving some properties (safety, reachability, ecc…)

Example of these tools are Ariadne, PHAVer, KeYmaera, Checkmate, HybridSAL,…

Control Synthesis tools usually restricted the attention to subclasses of Hybrid Systems:

- LTLCon (linear control systems)
- Hybrid Toolbox (piecewise-affine hybrid systems)

More recent tools dealing with logic specifications

LTL (Linear Temporal Logic) encodes formulae referring to time. On top of logical operators, it defines temporal operators such as Next, Until, Always, Eventually.

LTLMoP (Linear Temporal Logic MissiOn Planning) toolkit is a collection of Python applications for designing, testing, and implementing hybrid controllers generated automatically from task specifications written in Structured English or Temporal Logic

TuLiP: The temporal logic planning toolbox. Given a plant model and an LTL specification, design a controller to ensure that any execution of the system satisfies the specification.

PESSOA: correct-by-design embedded control software

PESSOA: A tool for embedded control software synthesis

Developed at UCLA's CyPhyLab, for the synthesis of correct-by-design embedded control software (2010-2011).

What was new in Pessoa?

- the nature of the abstractions (approximate simulations and bisimulations)
- the classes of systems admitting such abstractions (linear, nonlinear, and switched).

Fernando Pessoa was a 20th-century Portuguese poet and writer, and one of the most influential literates of the last century.

Lecture mostly based on:

[Mazo et al., CAV2010] M. Jr. Mazo, A. Davitian, and P. Tabuada, "Pessoa: A tool for embedded controller synthesis". In: Computer Aided Verification. Springer. 2010, pp. 566-569.

PESSOA and CUDD

More details:

- the core algorithms in Pessoa have been coded in C;
- the main functionalities are available through the Matlab command line;
- simulation of the closed-loop behavior in Simulink.

- All the systems and sets manipulated by Pessoa are represented symbolically using Reduced Ordered Binary Decision Diagrams (ROBDDs) supported by the CUDD library [CUD]. The package provides a large set of operations on BDDs.
- BDDs can be considered as a compressed representation of sets or relations. Unlike other compressed representations, operations are performed directly on the compressed representation, i.e. without decompression [BDDWiki].

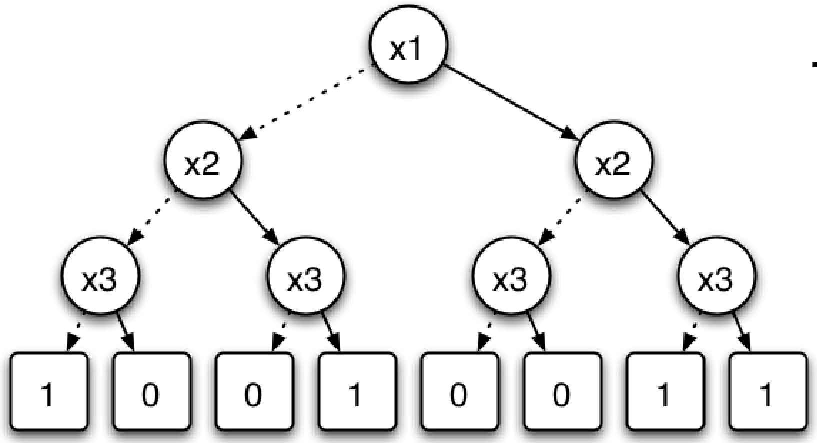
More details in:

[F. Somenzi, 1998] F. Somenzi, “CUDD: CU Decision Diagram Package”. Electronically available at: http://vlsi.colorado.edu/~fabio/CUDD/.

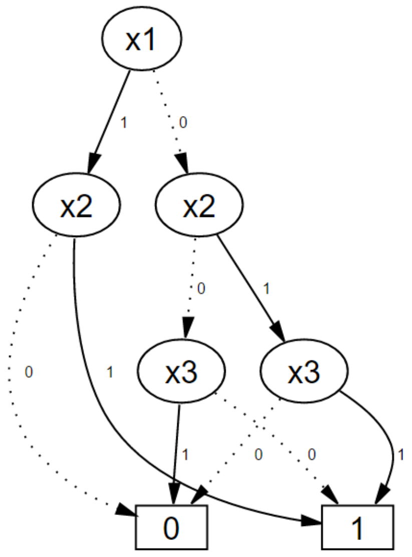
[BDDWiki] Binary decision diagram. In Wikipedia, The Free Encyclopedia. Retrieved 15:51, May 16, 2017, from https://en.wikipedia.org/w/index.php?title=Binary_decision_diagram

PESSOA and CUDD

From: https://en.wikipedia.org/wiki/Binary_decision_diagram



x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Binary decision tree and truth table for a boolean function and corresponding BDD

- In Pessoa, a sets of transition is decoded into a BDD T.
- T(x,u,x')=1 if (x,u,x') is a transition of the symbolic model.
- A preliminary change of coordinates transforms states and inputs in $\mathbb{R}^n$ and $\mathbb{R}^m$ into vectors of $\mathbb{Z}^n$ and $\mathbb{Z}^m$, respectively.
- In this way, integer variables are used to encode the states and inputs, and to perform all the computations.

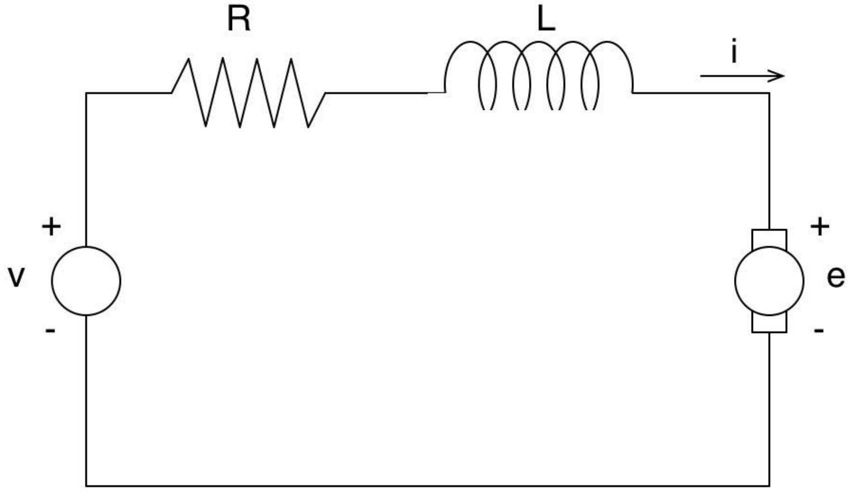
PESSOA: logical specifications

- **Stay**: trajectories start in the target set Z and remain in Z. This specification corresponds to the Linear Temporal Logic (LTL) formula $\square\varphi_Z$ where φ_Z is the predicate defining the set Z;
- **Reach**: trajectories enter the target set Z in finite time. This specification corresponds to the LTL formula $\lozenge\varphi_Z$;
- **Reach and Stay**: trajectories enter the target set Z in finite time and remain within Z thereafter. This specification corresponds to the LTL formula $\lozenge\square\varphi_Z$;
- **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 This specification corresponds to the LTL formula $\lozenge\square\varphi_Z \wedge \square\varphi_W$ where φ_W is the predicate defining the set W.

PESSOA: a linear example

Example 1: DC motor and associated electric circuit

$$\dot{x}_1 = -\frac{B}{J}x_1 + \frac{k}{J}x_2$$
$$\dot{x}_2 = -\frac{k}{L}x_1 - \frac{R}{L}x_2 + \frac{1}{L}u$$



- x_1 is the angular velocity
- x_2 is the current through the inductor
- u is the source voltage (control input).

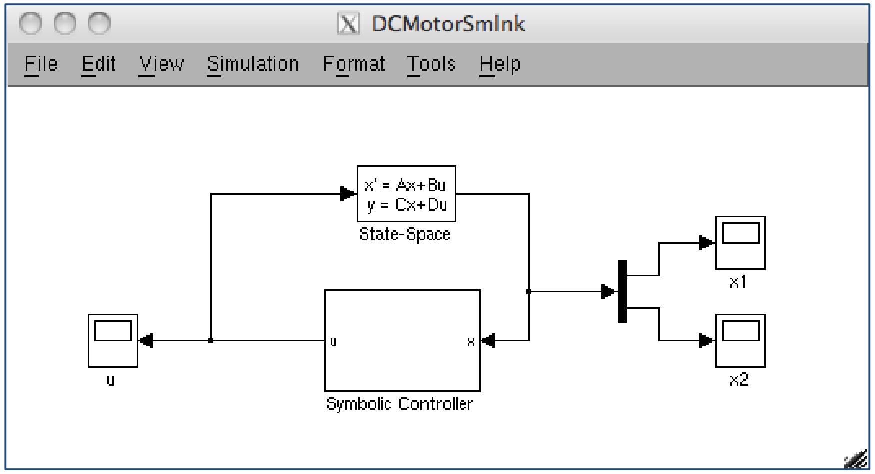
Parameter	Value	Description
R	500x10-3	Resistance
L	1500x10-6	Inductance
J	250x10-6	Moment of inertia
B	100x10-6	Viscous friction coefficient
k	50x10-3	Torque constant

PESSOA: a linear example

The system is turned into

$$\dot{x} = Ax + Bu$$

$$A = \begin{bmatrix} -\frac{B}{J} & \frac{k}{J} \\ -\frac{k}{L} & -\frac{R}{L} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}.$$



State and input spaces

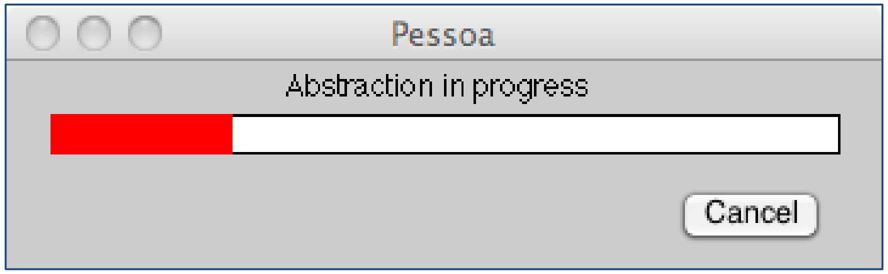
$$X = [-1, 30] \times [-10, 10] \quad U = [-10, 10]$$

Target set for a Reach and Stay Specification

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

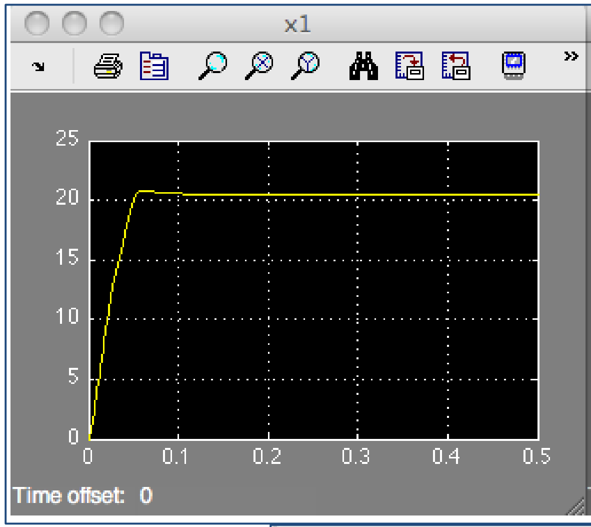
Quantization parameters

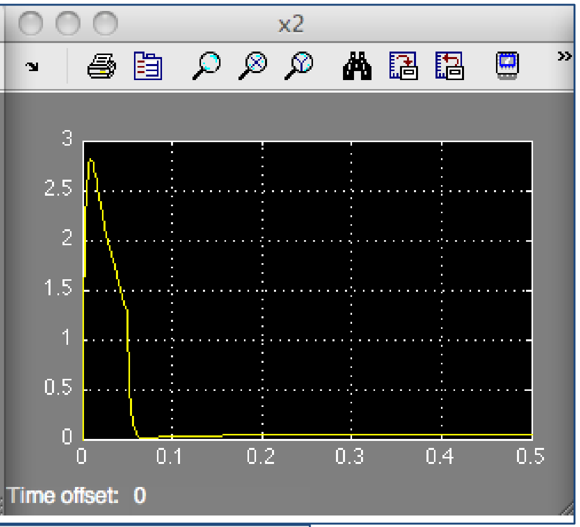
$$\tau = 0.05, \eta = 0.5, \mu = 0.01$$

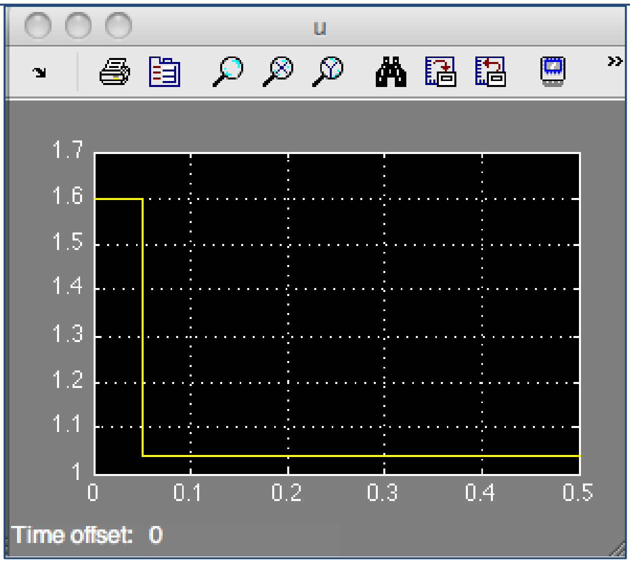


PESSOA: a linear example

Simulation results







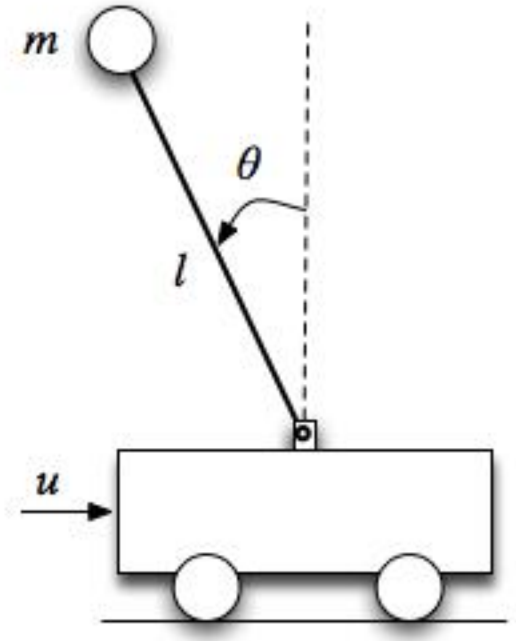
PESSOA: a nonlinear example

Example 2: Inverted pendulum on a cart

$$\dot{\theta} = \omega$$

$$\dot{\omega} = -\frac{g}{l} sin(\theta) - \frac{h}{ml^2}\omega + \frac{1}{ml} cos(\theta) u$$

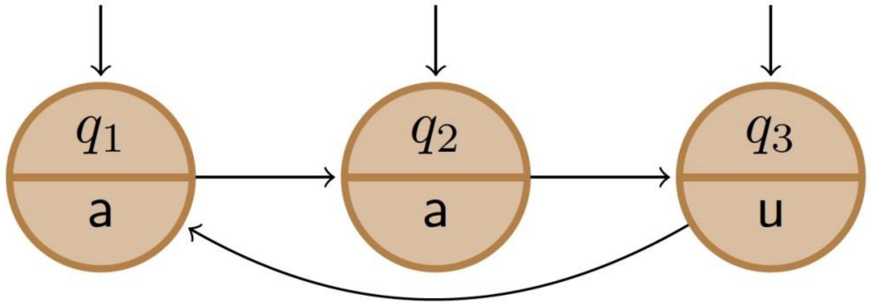
- θ is the angular position
- ω is the angular velocity of the point mass
- u is the applied force (control input)
- g=9.8 is gravity acceleration
- l=0.5 is the length of the rod
- m=0.5 is the mass
- h=2 is the coefficient of rotational friction



Since the system is unstable, we construct an abstraction that is approximately alternatingly simulated by the control system

PESSOA: a nonlinear example

Schedulability constraint: states are labeled with the outputs a and u denoting availability and unavailability of the microprocessor, respectively



State and input spaces

$$X = [-\pi/2, \pi/2] \times [-1, 1] \quad U = [-6, 6]$$

Target set for a Reach and Stay Specification

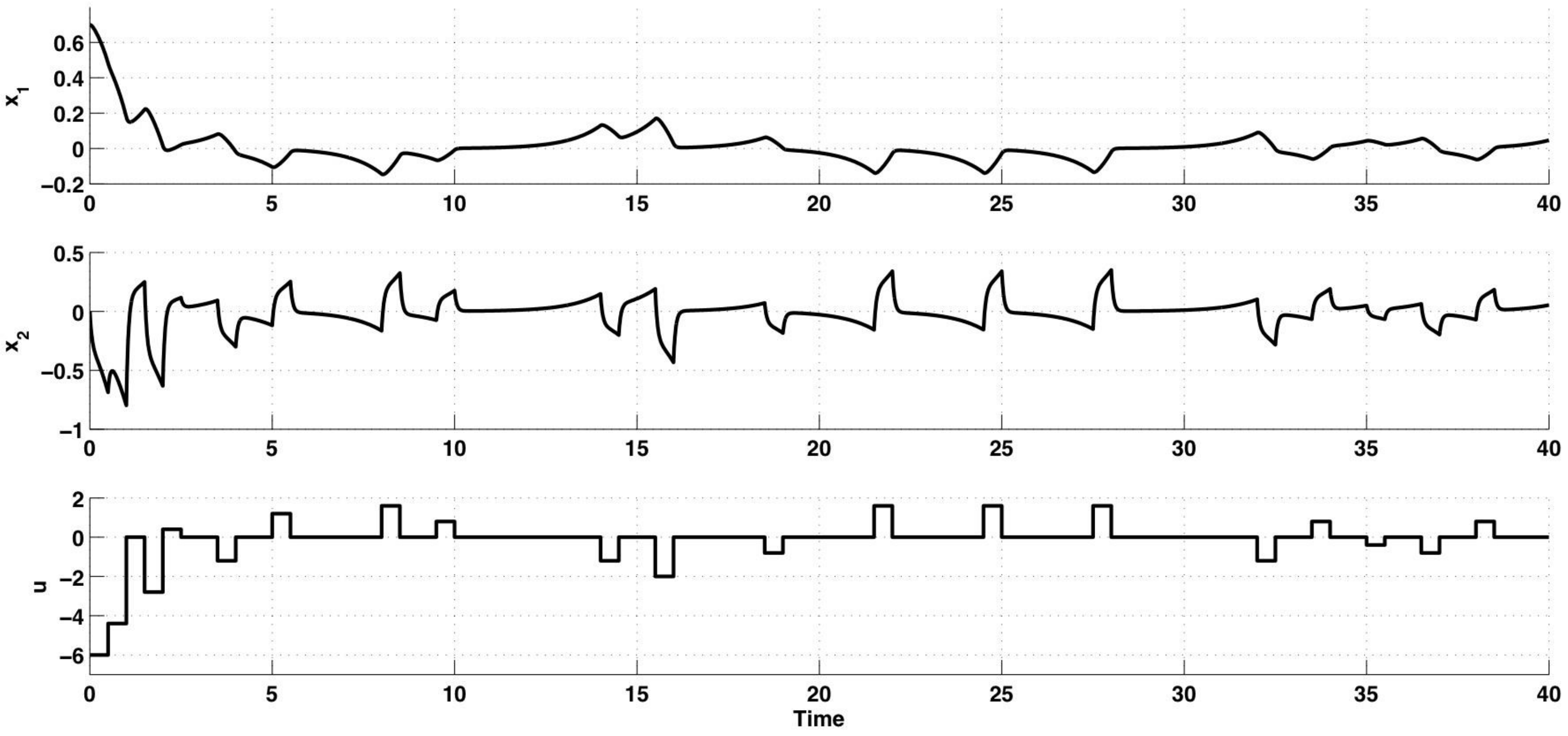
$$W = [-0.25, 0.25] \times [-1, 1]$$

Quantization parameters

$$\tau = 0.5, \eta = 0.02, \mu = 0.4$$

PESSOA: a nonlinear example

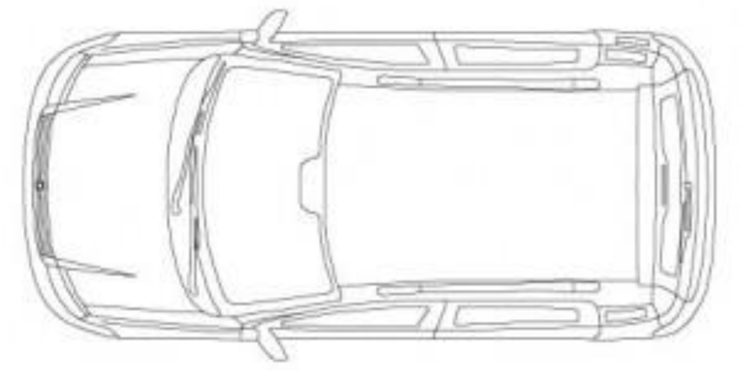
Simulation results



PESSOA: a nonlinear example

Example 3: unicycle

$$\dot{x} = v cos(\theta)$$
$$\dot{y} = v sin(\theta)$$
$$\dot{\theta} = \omega$$



- (x, y) denotes the position coordinates of the vehicle,
- θ is its orientation,
- (v, ω) are the control inputs, linear velocity and angular velocity respectively.

Since the system is unstable, we construct an abstraction that is approximately alternatingly simulated by the control system

PESSOA: a nonlinear example

State and input spaces

$$X = [1,5] \times [1,5] \times [-\pi,\pi] \quad U = [0,0.5] \times [-0.5,0.5]$$

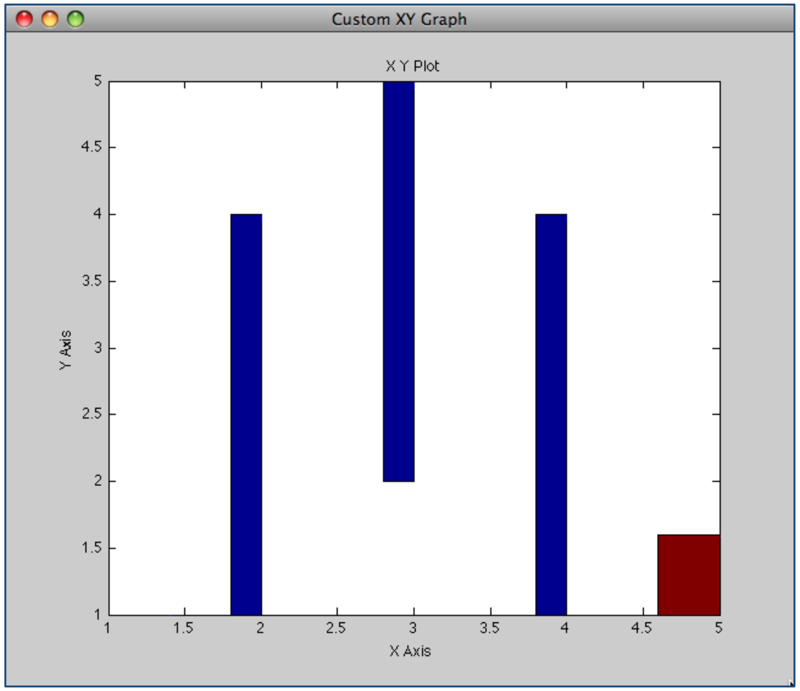
Specification Stay in Target Set while Staying in Safe Set

$$TargetSet = [4.5,5] \times [1,1.6] \times [-\pi,\pi]$$

Quantization parameters

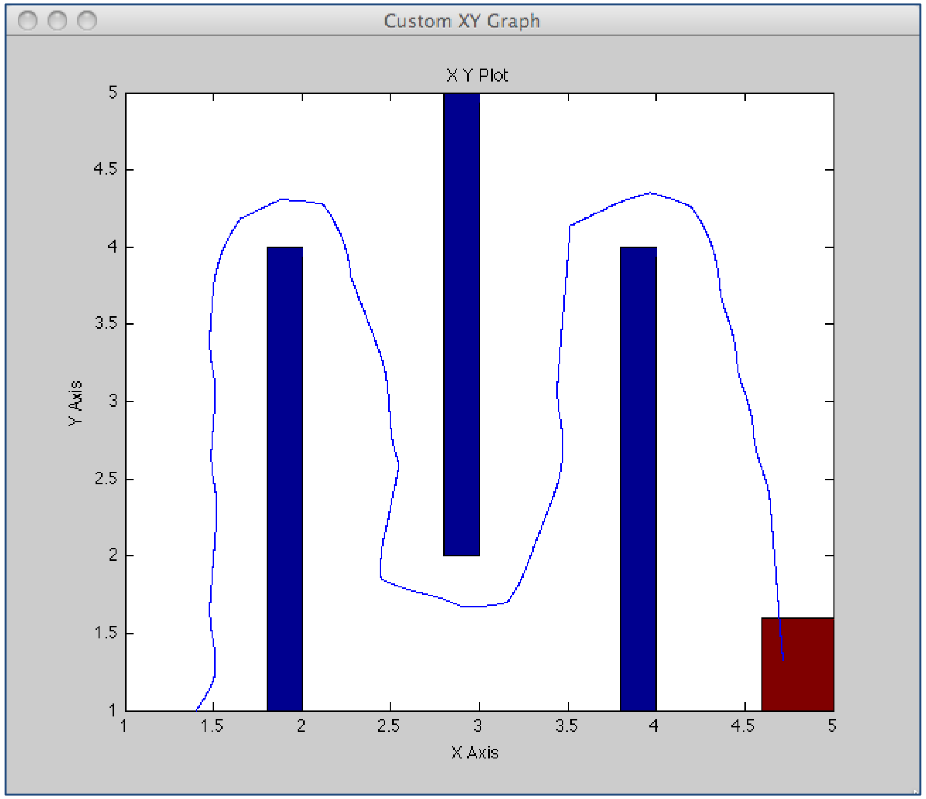
$$\tau = 0.5, \eta = 0.1, \mu = 0.5$$

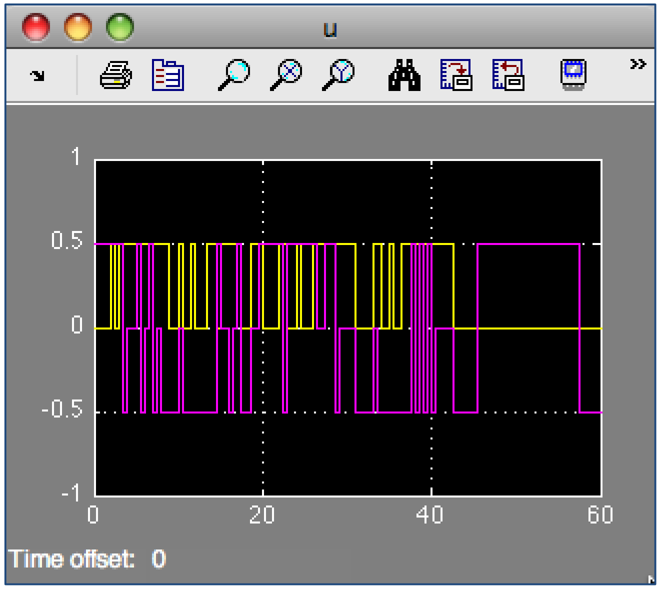
Remark: given the approximate nature of the model abstractions, the target set that will be reached is a set $\eta/2$ wider and taller than the red set. Similarly, the avoided obstacles are also $\eta/2$ thicker and longer than the blue obstacles.



PESSOA: a nonlinear example

Simulation results





Beyond PESSOA: CoSyMA

CoSyMA: a tool for controller synthesis using multi-scale abstractions

Developed at INRIA and University of Joseph Fourier, Grenoble (2012)

What is new in CoSyMA?

- Deals with incrementally stable switched systems
- Multi-scale abstractions

- Based on the CUDD library [CUD]
- Safety and time-bounded reachability specifications

More details in:

[Mouelhi et al., HSCC13] S. Mouelhi, A. Girard, and G. Gössler, “CoSyMA: a tool for controller synthesis using multi-scale abstractions”. Proceedings of the 16th international conference on Hybrid systems: computation and control. ACM, 2013.

Beyond PESSOA: SCOTS

SCOTS: A Tool for the Synthesis of Symbolic Controllers

Developed at Technical University of Munich (2016, ongoing).

What is new in SCOTS?

- Based on the more recent concept of feedback refinement relation (FRR)
- Does not require to include the abstraction as a building block (similarly to the integrated approach).

- Based on the CUDD library [CUD].
- Reachability and invariance specifications
- Synthesis via fixed point computations

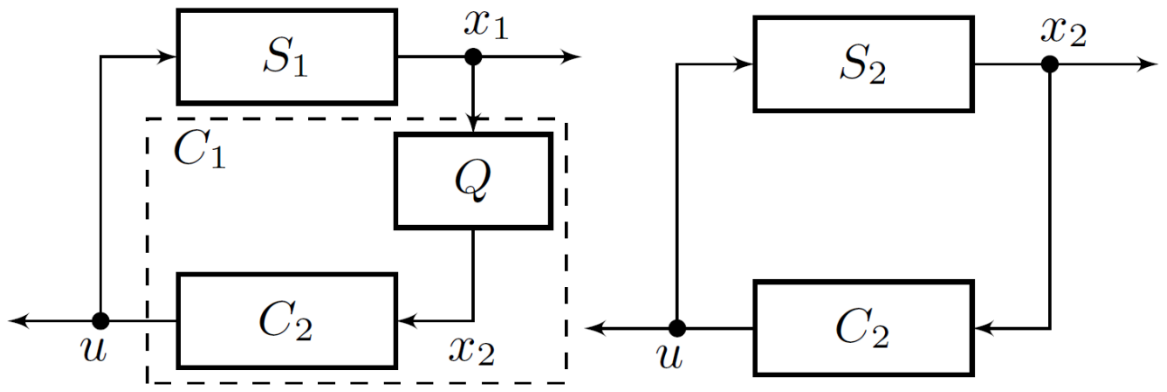
More details in:

[Rungger et al., HSCC16] M. Rungger, and M. Zamani, “SCOTS: A tool for the synthesis of symbolic controllers”. Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control. ACM, 2016.

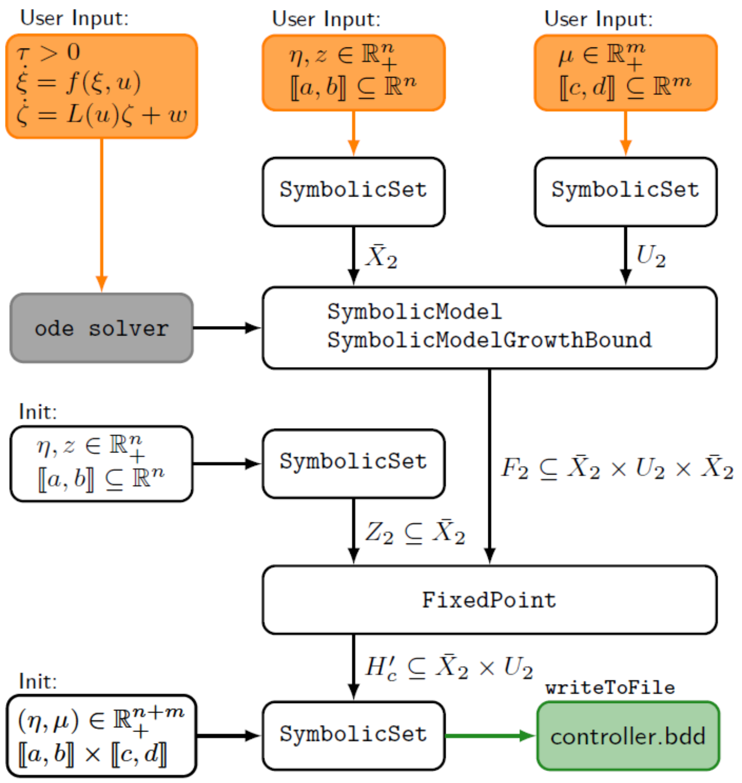
SCOTS: control design

Control design in 3 steps:

1. First, given a control problem (S_1, Σ_1), a finite simple system S_2 as a substitute of S_1, together with an abstract specification Σ_2 is computed. In this context, S_1 and S_2 are referred to as plant and symbolic model, respectively.
2. In the second step, a controller C_2, i.e., system that is feedback-composable with S_2, which solves the control problem (S_2, Σ_2) is computed.
3. Provided that the synthesis process of C_2 is successful, the controller C_2 is refined to a controller C_1 that solves the original problem (S_1, Σ_1) in the third step.

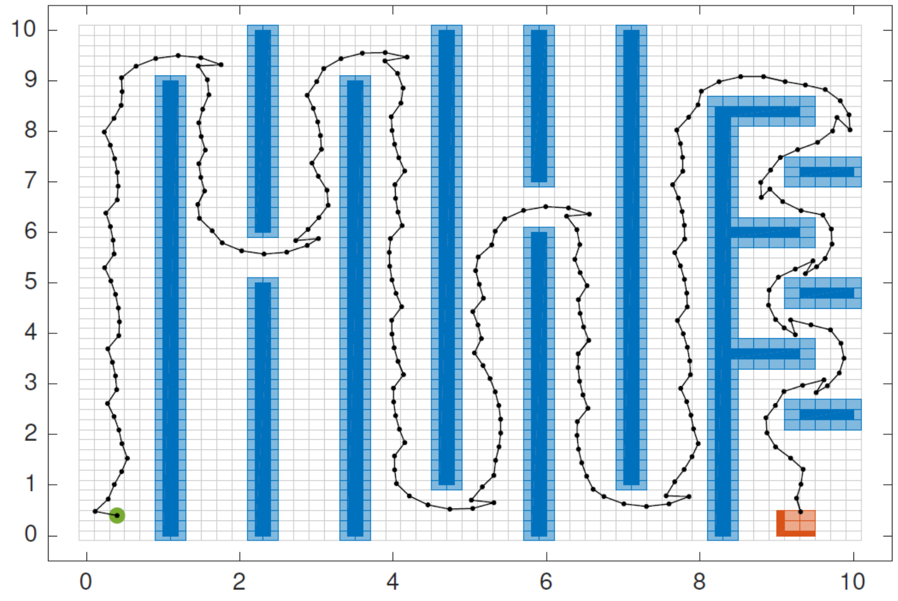


SCOTS: implementation



User Input:
$\tau > 0$
$\dot{\xi} = f(\xi, u)$
$\dot{\zeta} = L(u)\zeta + w$
User Input:
$\eta, z \in \mathbb{R}^n_+$
$[\![a, b]\!] \subseteq \mathbb{R}^n$
User Input:
$\mu \in \mathbb{R}^m_+$
$[\![c, d]\!] \subseteq \mathbb{R}^m$
SymbolicSet
SymbolicSet
$\bar{X}_2$
U_2
ode solver
SymbolicModel
SymbolicModelGrowthBound
Init:
$\eta, z \in \mathbb{R}^n_+$
$[\![a, b]\!] \subseteq \mathbb{R}^n$
SymbolicSet
$F_2 \subseteq \bar{X}_2 \times U_2 \times \bar{X}_2$
$Z_2 \subseteq \bar{X}_2$
FixedPoint
$H'_c \subseteq \bar{X}_2 \times U_2$
writeToFile
Init:
$(\eta, \mu) \in \mathbb{R}^{n+m}_+$
$[\![a, b]\!] \times [\![c, d]\!]$
SymbolicSet
controller.bdd

Implementation work flow



Example of reachability specification with collision avoidance