ISSN 1992-6502 (Print)
Vol. 17, no. 6 (59), pp. 38-42, 2013.
Vestnik UGATU
ISSN 2225-2789 (Online)
http://journal.ugatu.ac.ru

UDC 004.7

MODEL BASED DESIGN OF CONTROL SOFTWARE FOR NONLINEAR DISCRETE TIME HYBRID SYSTEMS

V. ALIMGUZHIN[1], F. MARI[2], I. MELATTI[3], I. SALVO[4], E. TRONCI[5]

[1] alimguzhin@di.uniroma1.it, [2] mari@di.uniroma1.it, [3] melatti@di.uniroma1.it,
[4] salvo@di.uniroma1.it, [5] tronci@di.uniroma1.it

Sapienza University of Rome, Italy

Submitted 2013, June 10

Abstract. Many *Embedded Systems* are indeed *Software Based Control Systems*, that is control systems whose controller consists of *control software* running on a microcontroller device. This motivates investigation on *Formal Model Based Design* approaches for automatic synthesis of embedded systems control software. This paper addresses control software synthesis for discrete time *nonlinear* hybrid systems. We present a methodology to over approximate the dynamics of a discrete time nonlinear hybrid system $\mathcal{H}$ by means of a discrete time *linear* hybrid system $\mathcal{L}_{\mathcal{H}}$, in such a way that controllers for $\mathcal{L}_{\mathcal{H}}$ are guaranteed to be controllers for $\mathcal{H}$. We present experimental results on control software synthesis for the inverted pendulum, a challenging and meaningful control problem.

Keywords: model based design; Embedded Systems; hybrid systems; discrete time.

1. INTRODUCTION

Many *Embedded Systems* are indeed *Software Based Control Systems* (SBCSs). An SBCS consists of two main subsystems: the *controller* and the *plant*, that together form a *closed loop system*. Typically, the plant is a physical system whereas the controller consists of *control software* running on a microcontroller. Software generation from models and formal specifications forms the core of *Model Based Design* of embedded software. This approach is particularly interesting for SBCSs since in such a case system level specifications are much easier to define than the control software behavior itself.

Traditionally, the control software is designed using a *separation-of-concerns* approach. That is, *Control Engineering* techniques are used to design *functional specifications* (*control law*) from the closed loop system level specifications, whereas *Software Engineering* techniques are used to design control software implementing functional specifications. Such a separation-of-concerns approach has several drawbacks. For example, correctness of the control software is not formally verified and issues concerning non-functional requirements (such as computational resources, control software *Worst Case Execution Time*, WCET), are considered very late in the SBCS design activity and this could lead to new iterations of the control design (e.g., if the WCET is greater than the sampling time).

The previous considerations motivate research on methods and tools focusing on control software synthesis. The objective is that from the plant model, from formal specifications for the closed loop system behavior and from *Implementation Specifications* (that is, number of bits used in the quantization process) such methods can generate correct-by-construction control software satisfying the given specifications.

The tool QKS [1] has been designed following an SBCS model based design approach. Given a plant modeled as a *Discrete Time Linear Hybrid System* (DTLHS) QKS automatically synthesises control software meeting given safety and liveness closed loop specifications. The dynamics of a DTLHS is modeled as a set of *linear constraints* over a set of continuous as well as discrete variables describing system state, system inputs and disturbances. Although the control software synthesis problem for DTLHSs is undecidable [3], the semi-algorithm implemented in QKS usually succeeds in generating control software.

However, the dynamics of many interesting hybrid systems cannot be directly modeled by linear constraints. This motivates the focus of the present paper: control software synthesis for *nonlinear Discrete Time Hybrid Systems* (DTHS).

The present paper is a survey on the on-going research on Model Based Control Software Synthesis. More technical details can be found in [1–5]. We present a general approach to *overapproximate* (that is possibly allowing more behaviours than) a

given DTHS $\mathcal{H}$ by means of a DTLHS $\mathcal{L}_{\mathcal{H}}$ such that controllers for $\mathcal{L}_{\mathcal{H}}$ are guaranteed to be controllers for $\mathcal{H}$. Control software for $\mathcal{H}$ is thus obtained by giving as input to the tool QKS [1] the linear plant model $\mathcal{L}_{\mathcal{H}}$. We show the effectiveness of our approach by presenting experimental results on the inverted pendulum benchmark, a challenging and well studied example in control synthesis.

2. BACKGROUND

2.1. Predicates

An *expression* $E(X)$ over a set of variables X is an expression of the form $\sum_{i\in[n]} a_i\, f_i(X)$, where $f_i(X)$ are possibly non linear functions and a_i are rational constants. For example, $3sinx$, $\frac{3}{2}logxy$, x^y, x are expressions over $\{x, y\}$. $E(X)$ is a *linear expression* if it is a linear combination of variables $\sum_{i\in[n]} a_i\, x_i$, i.e. for all i, $f_i(X) = x_i$ for some $x_i \in X$. A *constraint* is an expression of the form $E(X) \leq b$, where b is a rational constant. A *predicate* is a logical combination of constraints. A *conjunctive predicate* is a conjunction of constraints. We also write $E(X) \geq b$ for $-E(X) \leq -b$, $E(X) = b$ for $(E(X) \leq b) \wedge (E(X) \geq b)$, and $a \leq x \leq b$ for $(x \geq a) \wedge (x \leq b)$. Given a constraint $C(X)$ and a boolean variable $y \notin X$, the *guarded constraint* $y \rightarrow C(X)$ (if y then $C(X)$) denotes the predicate $(y = 0) \vee C(X)$. Similarly, $\overline{y} \rightarrow C(X)$ denotes $(y = 1) \vee C(X)$. A *guarded predicate* is a conjunction of either constraints or guarded constraints. A guarded predicate is *linear* if it contains only linear expressions.

2.2. Control Problem for a Labeled Transition System

A *Labeled Transition System* (LTS) is a tuple $\mathcal{S} = (S, A, T)$ where S is a (possibly infinite) set of states, A is a (possibly infinite) set of *actions*, and $: S \times A \times S \rightarrow \mathbb{B}$ is the *transition relation* of $\mathcal{S}$. Let $s \in S$ and $a \in A$. The set $\mathrm{Adm}(\mathcal{S}, s) = \{a \in A \mid \exists s' : T(s, a, s')\}$ is the set of actions admissible in s, and $\mathrm{Img}(\mathcal{S}, s, a) = \{s' \in S \mid T(s, a, s')\}$ is the set of next states from s via a. A *run* or *path* for an LTS $\mathcal{S}$ is a sequence $\pi = s_0, a_0, s_1, a_1, s_2, a_2, \ldots$ of states s_t and actions a_t such that $\forall t \geq 0\ T(s_t, a_t, s_{t+1})$. The length $|\pi|$ of a finite run π is the number of actions in π. We denote with $\pi^{(S)}(t)$ the $(t+1)$-th state element of π, and with $\pi^{(A)}(t)$ the $(t+1)$-th action element of π. That is $\pi^{(S)}(t) = s_t$, and $\pi^{(A)}(t) = a_t$. Given two LTSs $\mathcal{S}_1 = (S, A, T_1)$ and $\mathcal{S}_2 = (S, A, T_2)$, we say that $\mathcal{S}_2$ *overapproximates* $\mathcal{S}_1$ (notation $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$) when $T_1(s, a, s')$ implies $T_2(s, a, s')$ for all $s, s' \in S$ and $a \in A$. Note that $\sqsubseteq$ defines a partial order over LTSs.

A controller restricts the dynamics of an LTS so that all states in a given initial region will eventually reach a given goal region. In what follows, let $\mathcal{S} = (S, A, T)$ be an LTS, $I, G \subseteq S$ be, respectively, the *initial* and *goal* regions of $\mathcal{S}$ A *controller* for $\mathcal{S}$ is a function $K: S \times A \rightarrow \mathbb{B}$ such that $\forall s \in S$, $\forall a \in A$, if $K(s, a)$ then $\exists s' T(s, a, s')$. The set $\mathrm{dom}(K) = \{\, s \in S \mid \exists a\, K(s, a) \,\}$ is the set of states for which at least a control action is enabled. The *closed loop system* $\mathcal{S}^{(K)}$ is the LTS $\left(S, A, T^{(K)}\right)$, where $T^{(K)}(s, a, s') = T(s, a, s') \wedge K(s, a)$. We call a path π *fullpath* if either it is infinite or its last state $\pi^{(S)}(|\pi|)$ has no successors.

We denote with $\mathrm{Path}(s, a)$ the set of fullpaths starting in state s with action a. Given a path π in $\mathcal{S}$, we define $j(\mathcal{S}, \pi, G)$ as follows. If there exists $n > 0$ s. t. $\pi^{(S)}(n) \in G$, then $j(\mathcal{S}, \pi, G) = \min\{n \mid n > 0 \wedge \pi^{(S)}(n) \in G\}$. Otherwise, $j(\mathcal{S}, \pi, G) = +\infty$. We require $n > 0$ since our systems are non-terminating and each controllable state (including a goal state) must have a path of positive length to a goal state. Taking $\sup \emptyset = +\infty$ the *worst case distance* of a state s from the goal region G is $J(\mathcal{S}, G, s) = \sup\{j(\mathcal{S}, \pi, G) \mid a \in \mathrm{Adm}(\mathcal{S}, s), \pi \in \mathrm{Path}(s, a)\}$. A *control problem* for $\mathcal{S}$ is a triple $\mathcal{P} = (\mathcal{S}, I, G)$. A *solution* to $\mathcal{P}$ is a controller K for $\mathcal{S}$ such that $I \subseteq \mathrm{dom}(K)$ and for all $s \in \mathrm{dom}(K)$, $J(\mathcal{S}^{(K)}, G, s)$ is finite. An *optimal* solution to $\mathcal{P}$ is a solution K^* to $\mathcal{P}$, s.t. for all solutions K to $\mathcal{P}$, for all $\forall s \in S$ we have $J\left(\mathcal{S}^{(K^*)}, G, s\right) \leq J(\mathcal{S}^{(K)}, G, s)$.

3. Discrete Time Hybrid Systems

Definition 1. A *Discrete Time Hybrid System* is a tuple $\mathcal{H} = (X, U, Y, N)$ where:

$X = X^r \cup X^d$ is a finite sequence of real (X^r) and discrete (X^d) *present state* variables. The sequence X' of *next state* variables is obtained by decorating with $'$ all variables in X.

$U = U^r \cup U^d$ is a finite sequence of *input* variables.

$Y = Y^r \cup Y^d$ is a finite sequence of *auxiliary* variables.

$N(X, U, Y, X')$ is a guarded predicate over $X \cup U \cup Y \cup X'$ defining the *transition relation* of the system.

A *Discrete Time Linear Hybrid System* (DTLHS) is a DTHS whose transition relation N is linear.

The semantics of a DTHS $\mathcal{H}$ is given in terms of the labeled transition system $LTS(\mathcal{H}) =$

$(\mathcal{D}_X, \mathcal{D}_U, \widetilde{N})$ where: $\widetilde{N}: \mathcal{D}_X \times \mathcal{D}_U \times \mathcal{D}_X \to \mathbb{B}$ is a function s.t. $\widetilde{N}(x,u,x') \equiv \exists y \in \mathcal{D}_Y: N(x,u,y,x')$.

We say that DTHS $\mathcal{H}_2$ *overapproximates* DTHS $\mathcal{H}_1$ when $\mathrm{LTS}(\mathcal{H}_1) \sqsubseteq \mathrm{LTS}(\mathcal{H}_2)$.

Example 1. Let us consider a simple inverted pendulum. The system is modeled by taking the angle θ and the angular velocity $\dot{\theta}$ as state variables. The input of the system is the torquing force u, that can influence the velocity in both directions. Moreover, the behaviour of the system depends on the pendulum mass m, the length of the pendulum l and the gravitational acceleration g. Given such parameters, the motion of the system is described by the differential equation: $\ddot{\theta} = \frac{g}{l}\sin\theta + \frac{1}{ml^2}u$.

In order to obtain a state space representation, we consider the following normalized system, where x_1 is the angle θ and x_2 is the angular speed $\dot{\theta}$.

$$\dot{x}_1 = x_2 \qquad \dot{x}_2 = \frac{g}{l}\sin x_1 + \frac{1}{ml^2}u \qquad (1)$$

The DTHS model $\mathcal{H}$ for the pendulum is the tuple (X, U, Y, N), where $X = \{x_1, x_2\}$ is the set of continuous state variables, $U = \{u\}$ is the set of input variables, and $Y = \emptyset$. Differently from [5], we consider the problem of finding a discrete controller, whose decisions may be "apply the force clockwise" $(u = 1)$, "apply the force counterclockwise" $(u = -1)$, or "do nothing" $(u = 0)$. The intensity of the force will be given as a constant F. Finally, the discrete time transition relation N is obtained from the equations in Eq. 1. as the Euler approximation with sampling time T, i.e. the predicate $(x_1' = x_1 + Tx_2) \wedge (x_2' = x_2 + T\frac{g}{l}\sin x_1 + T\frac{1}{ml^2}Fu)$.

3.1. Quantized Control Problem for DTHSs

A DTHS control problem $(\mathcal{H}, I, G)$ is defined as the LTS control problem$(\mathrm{LTS}(\mathcal{H}), I, G)$. To manage real variables, in classical control theory the concept of *quantization* is introduced (Quantization is the process of approximating a continuous interval by a set of integer values. A *quantization function* γ for a real interval $I = [a, b]$ is a non-decreasing function $\gamma(I): I \to Z$ s. t. $\gamma(I)$ is a bounded integer interval. We extend quantizations to integer intervals, by stipulating that in such a case the quantization function is the identity function. Given a DTHS $\mathcal{H} = (X, U, Y, N)$, a *quantization* Γ is a set of quantization functions $\Gamma = \{\gamma_w \mid w \in X \cup U\}$. If $W = [w_1, \ldots w_k]$ is a list of variables and $v = [v_1, \ldots, v_k] \in \mathcal{D}_W$, we write $\Gamma(v)$ for the tuple $[\gamma_{w_1}(v_1), \ldots \gamma_{w_k}(v_k)]$.

Definition 2. Let $\mathcal{H} = (X, U, Y, N)$ be a DTHS, Γ be a quantization for $\mathcal{H}$ and $\mathcal{P} = (\mathcal{H}, I, G)$ be a DTHS control problem. A Γ *Quantized Feedback Control* (QFC) solution to $\mathcal{P}$ is a solution $K(x,u)$ to $\mathcal{P}$ s.t. there exists $\widehat{K}: \Gamma(\mathcal{D}_X) \times \Gamma(\mathcal{D}_U) \to \mathbb{B}$ s. t. $K(x,u) = \widehat{\mathrm{K}}(\Gamma(x), \Gamma(u))$.

4. LINEAR OVERAPPROXIMATION OF DTHSS

The tool QKS [1], given a DTLHS control problem $\mathcal{P} = (\mathcal{H}, I, G)$ and a quantization schema as input, yields as output control software implementing an optimal quantized controller for $\mathcal{P}$, whenever a sufficient condition holds. In this section we show how a DTHS $\mathcal{H}$ can be overapproximated by a DTLHS $\mathcal{L}_{\mathcal{H}}$, in such a way that $\mathrm{LTS}(\mathcal{H}) \sqsubseteq \mathrm{LTS}(\mathcal{L}_{\mathcal{H}})$. Corollary 1 ensures that controllers for $\mathcal{L}_{\mathcal{H}}$ are guaranteed to be controllers for $\mathcal{H}$.

4.1. DTHS Linearization

Let $C(V)$, with $V \subseteq X \cup U \cup Y \cup X'$, be a constraint in N that contains a nonlinear function as a subterm. Then $C(V)$ has the shape $f(R,W) + E(V) \leq b$, where $R \subseteq V^r$ is a set of n real variables $\{r_1, \ldots, r_n\}$, and $W \subseteq V^d$ is a set of discrete variables. For each $w \in \mathcal{D}_W$, we define the function $f_w(R)$ obtained from f, by instantiating discrete variables with w, i.e $f_w(R) = f(R,w)$. Then $C(V)$ is equivalent to the predicate $\bigwedge_{w \in \mathcal{D}_W}[f_w(R) + E(V) \leq b]$. In order to make the overapproximation tighter, we partition the domain $\mathcal{D}_R$ of each function $f_w(R)$ into m hyperintervals $I_1, I_2 \ldots I_m$, where $I_i = \prod_{j \in [n]}[a_j^i, b_j^i]$. In the following $R \in I_i$ will denote the conjunctive predicate $\bigwedge_{j \in [n]} a_j^i \leq r_j \leq b_j^i$.

Let $f_{w,i}^+(R)$ and $f_{w,i}^-(R)$ be over- and under-linear approximations of $f_w(R)$ over the hyperinterval I_i, i.e. such that $R \in I_i$ implies $f_{w,i}^-(R) \leq f_w(R) \leq f_{w,i}^+(R)$ (in [4] we show the systematic approach for finding such approximations for $\mathcal{C}^2$ functions using Taylor theorem). Taking $|\mathcal{D}_W| \times n$ fresh continuous variables $Y = \{y_{w,i}\}_{w \in \mathcal{D}_W, i \in [n]}$ and $|\mathcal{D}_W| \times n$ fresh boolean variables $Z = \{z_{w,i}\}_{w \in \mathcal{D}_W, i \in [n]}$, we define the guarded predicate $\bar{\mathrm{C}}(V, Y, Z)$:

$$\bigwedge_{w \in \mathcal{D}_W} \bigwedge_{i \in [m]} \left[y_{w,i} + E(V) \leq b\right]$$
$$\wedge \bigwedge_{w \in \mathcal{D}_W} \bigwedge_{i \in [m]} \left[z_{w,i} \to f_{w,i}^-(R) \leq y_{w,i} \leq f_{w,i}^+(R)\right]$$

$\wedge \bigwedge_{w\in\mathcal{D}_W} \bigwedge_{i\in[m]}[z_{w,i} \to R \in I_i] \wedge$
$\bigwedge_{w\in\mathcal{D}_W} \bigwedge_{i\in[m]}[z_{w,i} \geq 1]$.

This transformation eliminates a nonlinear subexpression of a constraint $C(V)$ and yields a constraint $\bar{C}(V,Y,Z)$ such that $\exists Y,Z\ [\bar{C}(V,Y,Z) \Rightarrow C(V)]$. Given a DTHS $\mathcal{H} = (X,U,Y,N)$, without loss of generality, we may suppose that the transition relation N is a conjunction $\bigwedge_{i\in[m]} C(X,U,Y,X')$ of constraints. By applying the above transformation to each nonlinear subexpression occurring in N, we obtain a conjunction of linear constraints $\bar{N} \equiv \bigwedge_{i\in[m]} \bar{C}(X,U,\bar{Y},X')$, such that $\bar{N} \Rightarrow N$. Hence starting from a DTHS $\mathcal{H}$, we find DTLHS $\mathcal{L}_{\mathcal{H}} = (X,U,\bar{Y},\bar{N})$, whose dynamics overapproximates the dynamics of $\mathcal{H}$.

Theorem 1. Let $\mathcal{H} = (X,U,Y,N)$ be a DTHS and $\mathcal{L}_{\mathcal{H}}$ be its linearization. Then we have $\mathrm{LTS}(\mathcal{H}) \sqsubseteq \mathrm{LTS}(\mathcal{L}_{\mathcal{H}})$.

Theorem 2. Let $\mathcal{S}_1 = (S,A,T_1)$ and $\mathcal{S}_2 = (S,A,T_2)$ be two LTSs and let K be a solution to the LTS Control Problem $(\mathcal{S}_2,I,G)$. If $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ and for all $s \in S$ $\mathrm{Adm}(\mathcal{S}_1,a) = \mathrm{Adm}(\mathcal{S}_2,a)$ then K is also solution to the LTS Control Problem $(\mathcal{S}_1,I,G)$.

Corollary 1. Let $\mathcal{H} = (X,U,Y,N)$ be a DTHS and $\mathcal{L}_{\mathcal{H}}$ be its linearization. Let K be a solution to the DTLHS Control Problem $(\mathcal{L}_{\mathcal{H}},I,G)$. Then K is a solution to the DTHS Control Problem $(\mathcal{H},I,G)$.

5. EXPERIMENTAL RESULTS

We present experimental results obtained by using QKS [1] on the inverted pendulum example described in Ex. 1. In all our experiments as in we set $l = g$ and $m = \frac{1}{l^2}$. We set the force intensity parameter $F = 0.5$.

We use uniform quantization functions dividing the domain of each state variable $\mathcal{D}_{x_1} = [-1.1\pi, 1.1\pi]$ (we write π for a rational approximation of it) and $\mathcal{D}_{x_2} = [-4,4]$ into 2^b equal intervals, where b is the number of bits used by AD conversion. Since we have two quantized variables, each one with b bits, the number of quantized states is exactly 2^{2b}. In the following, we sometimes make explicit the dependence on b by writing $K^{(b)}$.

The typical goal for the inverted pendulum is to turn the pendulum steady to the upright position, starting from any possible initial position, within a given speed interval. In our experiments, the goal region is defined by the predicate $G(X) \equiv (-\rho \leq x_1 \leq \rho) \wedge (-\rho \leq x_2 \leq \rho)$, where $\rho \in \{0.05, 0.1\}$, and the initial region is defined by the predicate $I(X) \equiv (-\pi \leq x_1 \leq \pi) \wedge (-4 \leq x_2 \leq 4)$.

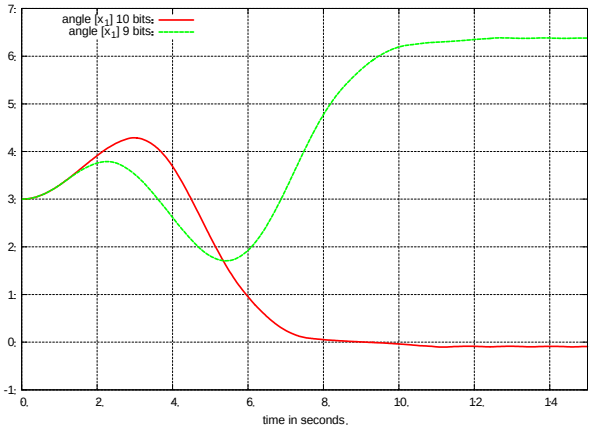

Fig. 1. Trajectories: $\mathcal{H}^{(K^{(9)})}$ and $\mathcal{H}^{(K^{(10)})}$

All experiments have been carried out on an Intel(R) Xeon(R) CPU @ 2.27GHz, with 23GiB of RAM, Debian GNU/Linux 6.0.3 (squeeze).

We run QKS for different values of the remaining parameters, i.e. ρ (goal tolerance) and b (number of bits of AD). In the Tab. 1. each row corresponds to a QKS run, columns b, T, ρ show the corresponding inverted pendulum parameters, column $|K|$ shows the size of the obtained control software, columns CPU and MEM show the computation time (in seconds) and RAM usage (in KiB) needed by QKS to synthesize controller. Fig.1. shows the simulations of $\mathcal{H}^{(K^{(9)})}$ and $\mathcal{H}^{(K^{(10)})}$. As we can see $K^{(10)}$ drives the system to the goal with a smarter trajectory with one swing only.

Table 1. Experimental results for inverted pendulum

b	T	ρ	$\lvert K\rvert$	CPU	MEM
8	0.1	0.1	2.73e+04	2.56e+03	7.72e+04
9	0.1	0.1	5.94e+04	1.13e+04	1.10e+05
10	0.1	0.1	1.27e+05	5.39e+04	1.97e+05
11	0.01	0.05	4.12e+05	1.47e+05	2.94e+05

6. CONCLUSIONS

We presented an automatic methodology to synthesize control software for nonlinear DTHS. The control software is correct-by-construction with respect to both System Level Formal Specifications of the closed loop system and Implementation Specifications, namely the quantization schema. The present work can be extended in several directions. First of all, it would be interesting to consider control synthesis of controllers that are optimal with respect to a cost function given as input of the control problem, rather than simply time-optimal. Second, it would be interesting to extend our approach to CTL specifications, rather than just liveness and safety properties. Finally, a natural possible future research direction is to investigate

DTHS control software synthesis when the state is not fully observable.

ACKNOWLEDGMENTS

This research was supported by Erasmus Mundus MULTIC scholarship from the European Commission (EMA 2 MULTIC 10-837).

REFERENCES

1. Federico Mari, Igor Melatti, Ivano Salvo, and Enrico Tronci, "Synthesis of quantized feedback control software for discrete time linear hybrid systems," *CAV*, LNCS 6174, pp. 180-195, 2010.
2. Federico Mari, Igor Melatti, Ivano Salvo, and Enrico Tronci, "From boolean relations to control software," *ICSEA,* pp. 528–533, 2011.
3. F. Mari, I. Melatti, I. Salvo, and E. Tronci, "Undecidability of quantized state feedback control for discrete time linear hybrid systems," *ICTAC*, LNCS 7521, pp. 243–258, 2012.
4. V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci, "Automatic control software synthesis for quantized discrete time hybrid systems," *CDC*, IEEE, pp. 6120-6125, 2012.
5. V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci, "On model based synthesis of embedded control software," *EMSOFT*, ACM, pp. 227-236, 2012.

ABOUT AUTHORS

ALIMGUZHIN, Vadim, Postgrad. (PhD) Student, Computer Science Department, Sapienza University of Rome. Master of Software and Administration of Information Systems (USATU, 2009).

TRONCI, Enrico, Associate Prof., Computer Science Department, Sapienza University of Rome. Master of Electrical Engineering (Sapienza University of Rome, 1987), PhD in Computer Science (Carnegie Mellon University, 1991).

SALVO, Ivano, Assistant Prof., Computer Science Department, Sapienza University of Rome. Master of Computer Science (University of Udine, 1995), PhD in Computer Science (Sapienza University of Rome, 2000).

MELATTI, Igor, Assistant Prof., Computer Science Department, Sapienza University of Rome. Master of Computer Science (University of L'Aquila, 2001), PhD in Computer Science (University of L'Aquila, 2005).

MARI, Federico, Post Doc, Computer Science Department, Sapienza University of Rome. Master of Computer Science (Sapienza University of Rome, 2006), PhD in Computer Science (Sapienza University of Rome, 2009).