Automatic Control Software Synthesis for Quantized Discrete Time Hybrid Systems

Vadim Alimguzhin, Federico Mari, Igor Melatti, Ivano Salvo, Enrico Tronci

***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 overapproximate 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.**

I. 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 [16]. 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.

Regarding filtering, if any, as a part of the state sensing process and assuming that the plant state is observable, the typical control loop skeleton for an SBCS is the following. In an endless loop, measure x of the system state from plant *sensors* go through an *analog-to-digital* (AD) conversion, yielding a *quantized* value $\hat{x}$ to the control software. A function `ctrlRegion` checks if $\hat{x}$ belongs to the region in which the control software works correctly. If this is not the case, a *Fault Isolation and Recovery* (FDIR) procedure is triggered, otherwise a function `ctrlLaw` computes a command $\hat{u}$ to be sent to plant *actuators* after a *digital-to-analog* (DA) conversion, in order to guarantee that the closed loop system meets given *safety* and *liveness* specifications (*System Level Formal Specifications*). Basically, the control software design problem for SBCSs consists in designing software implementing functions `ctrlLaw` and `ctrlRegion`.

Traditionally, the control software is designed using a *separation-of-concerns* approach. That is, *Control Engineering* techniques (e.g., see [6]) 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 [19] 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 [18], 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).

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 [19] the linear plant model $\mathcal{L}_{\mathcal{H}}$. Since $\mathcal{L}_{\mathcal{H}}$ overapproximates $\mathcal{H}$, the controllers that we synthesize are inherently *robust*, that is they meet the given closed loop requirements *notwithstanding* nondeterministic small *disturbances*.

As in the linear case, the automatically generated control software has a WCET guaranteed to be linear in the number of bits of the state quantization schema and it implements a (near) *time optimal* strategy [10] to reach the goal for the the closed loop system. We show the effectiveness of our approach by presenting experimental results on the inverted pendulum benchmark [17], a challenging and well studied example in control synthesis.

The authors are with the Computer Science Department, Sapienza University of Rome, Via Salaria 113, 00198 Roma, Italy.

Vadim Alimguzhin is also with the Department of Computer Science and Robotics Ufa State Aviation Technical University 12 Karl Marx Street, Ufa, 450000, Russian Federation

A. Related Work

The paper closer to our is [17] which studies the problem of control synthesis for discrete time (possibly nonlinear) systems. However, while we present an automatic method, the approach in [17] is not automatic since it requires the user to provide a suitable Lyapunov function.

In [21] it is presented an automatic method that, taking as input a continuous time linear system and a goal specification, produces a control law (represented as an OBDD) through PESSOA [20]. In contrast, our contribution focuses on (discrete time) possibly nonlinear *hybrid* systems (DTHS). Furthermore, [21] does not supply an effective method to generate control software and as a consequence it does not give any guarantee on WCET.

Stemming from suitable symbolic models for nonlinear control systems [22], a method to find overapproximations of *switched systems* is presented in [11]. In combination with [21], such results provide a semi-automatic method for finding a control law for nonlinear and switched systems. However, we note that nonlinear systems in [22] are not hybrid, since they cannot handle discrete variables, and in a switched system as in [11] mode transitions can only depend on control inputs, whereas in a hybrid system they can be triggered also by state changes. Moreover, [21] combined with [11] and [22] provides semi-automatic methods since they rely on a Lyapunov function provided by the user, much in the spirit of [17].

Verification and control law synthesis for *Linear Hybrid Automata* (LHA) [1], [2] has been investigated in [2], [12], [15], [24], [8], [23], [5]. Control law synthesis for *Piecewise Affine Discrete Time Hybrid Systems* (PWA-DTHS) has been investigated in [3], [4]. Explicit control synthesis algorithms for discrete time hybrid systems have been studied in [7]. All such approaches do not account for quantization since they all assume *exact* state measures. Thus, they do not offer any formal guarantee about system level correctness of the generated software, which is instead our focus here.

The tool QKS [19] synthesizes control software from system level specifications for DTLHSs. Here, we address control software synthesis for a more general class of discrete time hybrid systems. The overapproximation of hybrid systems with linear hybrid systems has been studied in [14], [13]. Such works consider dense time models, and focus on verification rather than control synthesis.

II. BACKGROUND

We denote with $[n]$ an initial segment $\{1,\dots,n\}$ of the natural numbers. We denote with $X = [x_1,\dots,x_n]$ a finite sequence of distinct variables, that we may regard, when convenient, as a set. Each variable x ranges on a known bounded interval $\mathcal{D}_x$ either of the reals or of the integers (discrete variables). Boolean variables are discrete variables ranging on the set $\mathbb{B} = \{0, 1\}$. We denote with $\mathcal{D}_X$ the set $\prod_{x\in X}\mathcal{D}_x$. To clarify that a variable x is *continuous* (resp. discrete, boolean) we may write x^r (resp. x^d, x^b). Analogously X^r (X^d, X^b) denotes the sequence of real (integer, boolean) variables in X. Finally, if x is a boolean variable we write $\bar{x}$ for $(1-x)$.

A. 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, $3\sin x$, $\frac{3}{2}\log xy$, 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 $x_i \in X$. Observe that our notion of linearity is merely *syntactical* much as arithmetic expressions are in programming languages. For example, for us $x+y$ is a linear expression, while $x+y+\sin x-\sin x$ is not, even though they "semantically" denote the same function. A *constraint* is an expression of the form $E(X) \le b$, 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 $E(X) \ge b$ for $-E(X) \le -b$, $E(X) = b$ for $(E(X) \le b) \wedge (E(X) \ge b)$, and $a \le x \le b$ for $(x \ge a) \wedge (x \le b)$. Given a constraint $C(X)$ and a boolean variable $y \notin X$, the *guarded constraint* $y \to C(X)$ (if y then $C(X)$) denotes the predicate $(y = 0) \vee C(X)$. Similarly, $\bar{y} \to C(X)$ 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.

B. 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 $T : S \times A \times S \to \mathbb{B}$ 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 $\pi = s_0,a_0,s_1,a_1,s_2,a_2,\dots$ of states s_t and actions a_t such that $\forall t \ge 0\ T(s_t,a_t,s_{t+1})$. 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. We formalize such a concept by defining solutions to an LTS control problem. 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 $K : S \times A \to \mathbb{B}$ 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 $(S,A,T^{(K)})$, 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

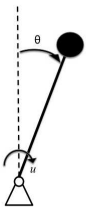

Fig. 1: Inverted Pendulum with Stationary Pivot Point.

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 $\sup \varnothing = +\infty$ 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 $s \in \mathcal{D}_X$ we have $J(\mathcal{S}^{(K^*)},G,s) \leq J(\mathcal{S}^{(K)},G,s)$.

III. Discrete Time Hybrid Systems

In this section we introduce our class of *Discrete Time Hybrid Systems* (DTHS), together with the DTHS representing the inverted pendulum on which our experiments will focus. Moreover, we will define the *Quantized Control Problem*.

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.

Input variables model *controllable inputs*, whereas auxiliary variables model *uncontrollable inputs*, i.e. disturbances. We do not have output variables since we focus on systems whose state is fully observable.

The semantics of a DTHS $\mathcal{H}$ is given in terms of the labeled transition system $\mathrm{LTS}(\mathcal{H}) = (\mathcal{D}_X, \mathcal{D}_U, \tilde{N})$ where: $\tilde{N} : \mathcal{D}_X \times \mathcal{D}_U \times \mathcal{D}_X \to \mathbb{B}$ is a function s.t. $\tilde{N}(x,u,x') \equiv \exists y \in \mathcal{D}_Y : N(x,u,y,x')$ (observe that if Y is empty then $\tilde{N}$ is just N since no existentialization takes place). 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 [17], as shown in Fig. 1. 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 $\dot{\theta}$.

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

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 $Y = \varnothing$. Differently from [17], 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 (1) as the Euler approximation with sampling time T, i.e. the predicate $(x_1' = x_1 + Tx_2) \wedge (x_2' = x_2 + T\frac{g}{l}\sin x_1 + T\frac{1}{ml^2}Fu)$.

Example 2: Disturbances can be modeled by using auxiliary variables as uncontrollable inputs. For example, let us consider the DTLHS $\mathcal{H}_1 = (X,U,Y_1,N_1)$ with: $X = \{x\}$, $U = \{u\}$, $Y_1 = \varnothing$ and $N_1(X,U,Y_1,X') = \{x' = 3x + u\}$. Let $\mathcal{H}_2 = (X,U,Y_2,N_2)$ with: $Y_2 = \{d\}$ (e.g., with $\mathcal{D}_d = [-1,1]$) and $N_2(X,U,Y_2,X') = \{x' = 3x + u + d\}$. Then $\mathcal{H}_2$ models disturbances within $\mathcal{H}_1$ ranging in the real interval $[-1,1]$. Note that $\mathcal{H}_2$ overapproximates $\mathcal{H}_1$ since any trajectory of $\mathcal{H}_1$ is also a trajectory of $\mathcal{H}_2$.

A. 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 (e.g., see [9]). 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 $\gamma : I \mapsto \mathbb{Z}$ 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 $[\gamma_{w_1}(v_1),\ldots\gamma_{w_k}(v_k)]$.

Example 3: In our experiments 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}.

A control problem admits a *quantized* solution if control decisions can be made by just looking at quantized values. This enables a software implementation for a controller.

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 $\hat{K} : \Gamma(\mathcal{D}_X) \times \Gamma(\mathcal{D}_U) \to \mathbb{B}$ such that $K(x,u) = \hat{K}(\Gamma(x),\Gamma(u))$.

Example 4: The typical goal for the inverted pendulum in Ex. 1 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 $G(X) \equiv (-\rho \le x_1 \le \rho) \wedge (-\rho \le x_2 \le \rho)$, where $\rho \in \{0.05, 0.1\}$, and the initial region is defined by the predicate $I(X) \equiv (-\pi \le x_1 \le \pi) \wedge (-4 \le x_2 \le 4)$.

IV. Linear overapproximation of DTHSs

The tool QKS [19], 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 3 ensures that controllers for $\mathcal{L}_\mathcal{H}$ are guaranteed to be controllers for $\mathcal{H}$.

A. 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 $f(R,W) + E(V) \le b$, 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 $\bigwedge_{w \in \mathcal{D}_W} [f_w(R) + E(V) \le b]$. 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 $I_i = \Pi_{j \in [n]} [a^i_j, b^i_j]$. In the following $R \in I_i$ will denote the conjunctive predicate $\bigwedge_{j \in [n]} a^i_j \le r_j \le b^i_j$.

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 $f^-_{w,i}(R) \le f_w(R) \le f^+_{w,i}(R)$. 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 $Z = \{z_i\}_{w \in \mathcal{D}_W, i \in [n]}$, we define the guarded predicate $\overline{C}(V,Y,Z)$:

$$\begin{aligned}
&\bigwedge_{w \in \mathcal{D}_W} \bigwedge_{i \in [m]} [y_{w,i} + E(V) \le b] \\
&\quad \wedge \bigwedge_{w \in \mathcal{D}_W} \bigwedge_{i \in [m]} z_{w,i} \rightarrow f^-_{w,i}(R) \le y_{w,i} \le f^+_{w,i}(R) \\
&\wedge \bigwedge_{w \in \mathcal{D}_W} \bigwedge_{i \in [m]} z_{w,i} \rightarrow R \in I_i \wedge \bigwedge_{w \in \mathcal{D}_W} \sum_{i \in [m]} z_{w,i} \ge 1
\end{aligned}$$

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

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

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 for the LTS control problem $(\mathcal{S}_2,I,G)$. If $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ and for all $s \in S$ $\mathrm{Adm}(\mathcal{S}_1,s) = \mathrm{Adm}(\mathcal{S}_2,s)$, then K is a solution also for $(\mathcal{S}_1,I,G)$.

Corollary 3: Let $\mathcal{H} = (X,U,Y,N)$ be a DTHS and let $\mathcal{L}_\mathcal{H}$ be its linearization. Let K be a solution for the DTLHS control problem $(\mathcal{L}_\mathcal{H},I,G)$. Then K is a solution also for the DTHS control problem $(\mathcal{H},I,G)$.

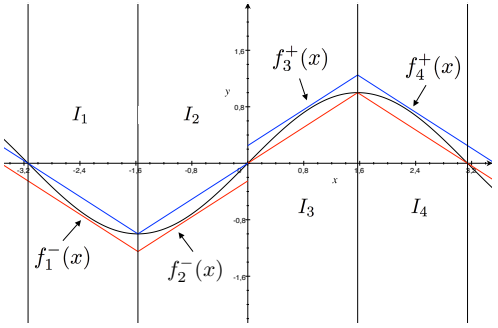

Fig. 2: Linearization of $\sin x$ in $[-\pi,\pi]$.

Example 5: The DTHS model $\mathcal{H}$ for the inverted pendulum in Ex. 1 contains the nonlinear function $\sin x_1$. We define the linearization $\mathcal{L}_\mathcal{H} = (X,U,Y,\tilde{N})$ as follows. In order to exploit sinus periodicity, we consider the equation $x_1 = 2\pi y_k + y_\alpha$, where y_k represents the period in which x_1 lies and $y_\alpha \in [-\pi,\pi]$ represents the actual x_1 inside a given period. This allows us to apply our linearization to $y_\alpha \in [-\pi,\pi]$ only. We partition the interval $[-\pi,\pi]$ into four sub-intervals I_1, I_2, I_3, I_4 as shown in Fig. 2. For $y_\alpha \in I_1 = [-\pi,-\frac{\pi}{2}]$ we define $f_1^+(y_\alpha)$ as the line passing through points $(-\pi,\sin(-\pi))$ and $(-\frac{\pi}{2},\sin(-\frac{\pi}{2}))$, i.e. $f_1^+(y_\alpha) = -0.6369 y_\alpha + 2$. Moreover, we define $f_1^-(y_\alpha)$ as the line which is tangent to the curve $\sin y_\alpha$ at I_1 medium point, i.e. $f_1^-(y_\alpha) = 0.7073(y_\alpha + 0.785) - 0.7068$. Functions $f_2^\pm$, $f_3^\pm$ and $f_4^\pm$ are obtained analogously. Finally, we have that $Y = Y^d \cup Y^r = \{y_k, y_q, z_1, z_2, z_3, z_4\} \cup \{y_\alpha\}$ and $\tilde{N} \equiv (x_1' = x_1 + 2\pi y_q + Tx_2) \wedge (x_2' = x_2 + T\frac{g}{l} y_\alpha + T\frac{1}{ml^2} Fu) \wedge x_1 = 2\pi y_k + y_\alpha \wedge \bigwedge_{i=1}^4 z_i \rightarrow f_i^- \le y_\alpha \le f_i^+ \wedge \bigwedge_{i=1}^4 z_i \rightarrow x_1 \in I_i \wedge \sum_{i=1}^4 z_i \ge 1$.

B. Linearization: a systematic approach

When nonlinear subexpressions are $\mathcal{C}^2$ functions, a systematic approach to compute linear overapproximations of a DTHS makes use of Taylor polynomial of degree 1 as piecewise affine functions that over- and under-approximate the value of a $\mathcal{C}^2$ function. Let $f(x)$ be a $\mathcal{C}^2$ function of n real variables over a given interval I. By Taylor's theorem, we may derive *linear* under- and over-approximations for $f(x)$ around a given point $x_0 \in I$ as follows. Namely, we have that there exists $t \in [0,1]$ such that $f(x) = f(x_0) + \nabla f(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T H(x + t(x - x_0))(x - x_0)$, being H the Hessian matrix of f. If we know two real numbers m and M that are the minimum and the maximum value of $\frac{1}{2}(x - x_0)^T H(x + t(x - x_0))(x - x_0)$, in a given interval around x_0 we can choose $f^+(x) = f(x_0) + \nabla f(x_0)(x - x_0) + M$ and $f^-(x) = f(x_0) + \nabla f(x_0)(x - x_0) + m$.

V. Experimental Results

In this section we present our experiments that aim at evaluating effectiveness of our linearization technique. We present experimental results obtained by using QKS [19] on the inverted pendulum described in Ex. 1. In order to let QKS handle such a case study, we linearize the DTHS $\mathcal{H}$ in Ex. 1 with the DTLHS $\mathcal{L}_\mathcal{H}$ of Ex. 5. In all our experiments, as in [17] we set parameters l and m in such a way that $\frac{g}{l} = 1$ (i.e. $l = g$) and $\frac{1}{ml^2} = 1$ (i.e. $m = \frac{1}{l^2}$). The quantization Γ is as in Ex. 3. The initial region I and goal region G are

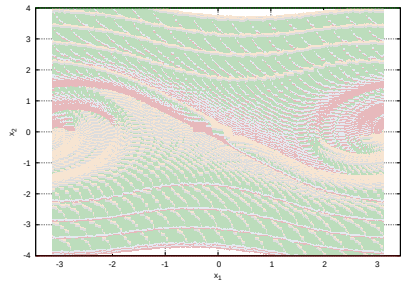

Fig. 3: $\mathrm{dom}(K_{0.5}^{(9)})$ ($T = 0.1$)

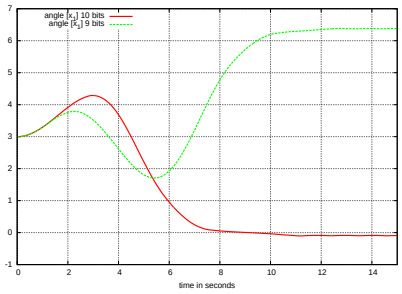

Fig. 4: Trajectories: $\mathcal{H}^{(K_{0.5}^{(9)})}$ and $\mathcal{H}^{(K_{0.5}^{(10)})}$

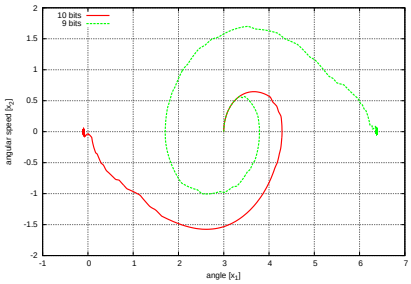

Fig. 5: Traj. of Fig. 4 in the phases space.

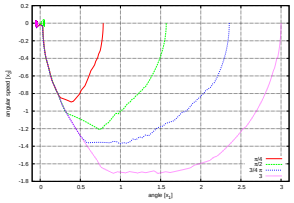

Fig. 6: $\mathcal{H}^{(K_2^{(11)})}$ phases space.

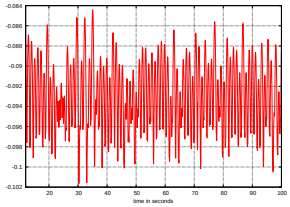

Fig. 7: Ripple: x_1 in $\mathcal{H}^{(K_{0.5}^{(10)})}$

as in Ex. 4, thus the DTHS [DTLHS] control problem we consider is $P = (\mathcal{H}, I, G)$ [$(\mathcal{L}_{\mathcal{H}}, I, G)$].

We run QKS for different values of the remaining parameters, i.e. F (force intensity), ρ (goal tolerance), T (sampling time), and b (number of bits of AD). For each of such experiments, QKS outputs a control software K in C language. In the following, we sometimes make explicit the dependence on F and b by writing $K_F^{(b)}$. In order to evaluate performance of K, we use an *inverted pendulum simulator* written in C. The simulator computes the next state by using Eq. (1) of Ex. 1, thus simulating a path of $\mathcal{H}^{(K)}$. Such simulator also introduces random disturbances (up to 4%) in the next state computation to assess K robustness w.r.t. non-modelled disturbances. Finally, in the simulator Eq. (1) is translated into the discrete time version by means of a simulation time step T_s much smaller than the sampling time T used in $\mathcal{H}$ (and $\mathcal{L}_{\mathcal{H}}$). Namely, $T_s = 10^{-6}$ seconds, whilst $T = 0.01$ or $T = 0.1$ seconds. This allows us to have a more accurate simulation. Accordingly, K is called each 10^4 (or 10^5) simulation steps of $\mathcal{H}$. When K is not called, the last chosen action is selected again (*sampling and holding*).

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).

A. Underactuated Inverted Pendulum ($F = 0.5$)

To stabilize an *underactuated* inverted pendulum (i.e. $F < 1$) from the hanging position to the upright position, a controller needs to find a non obvious strategy that consists of swinging the pendulum once or more times to gain enough momentum. QKS is able to synthesize such a controller taking as input $\mathcal{L}_{\mathcal{H}}$ with $F = 0.5$ (note that in [17] $F = 0.7$). Results are in Tab. I, where each row corresponds to a QKS run, columns b, T and ρ show the corresponding inverted pendulum parameters, column $|K|$ shows the size of the C code for $K_{0.5}^{(b)}$, and columns CPU and MEM show the computation time (in seconds) and RAM usage (in KB) needed by QKS to synthesize $K_{0.5}^{(b)}$.

As for $K_{0.5}^{(b)}$ performance, it is easy to show that by reducing the sampling time T and the quantization step (i.e. increasing b), we increase the quality of $K_{0.5}^{(b)}$ in terms of ripple and set-up time. Fig. 4 and 5 show the simulations of $\mathcal{H}^{(K_{0.5}^{(9)})}$ and $\mathcal{H}^{(K_{0.5}^{(10)})}$. As we can see, $K_{0.5}^{(10)}$ drives the system to the goal with a smarter trajectory, with one swing only. This have a significant impact on the set-up time (the system stabilizes after about 8 seconds when controlled by $K_{0.5}^{(10)}$ instead of about 10 seconds required when controlled by $K_{0.5}^{(9)}$). Fig. 3 shows that $\mathrm{dom}(K_{0.5}^{(9)})$ covers almost all states in the admissible region that we consider. Different colors mean different set of actions enabled by the controller. Finally, Fig. 7 shows the ripple of x_1 for $\mathcal{H}^{(K_{0.5}^{(10)})}$ inside the goal. Note that such ripple is very low (0.018 radiants).

B. Very Underactuated Inverted Pendulum ($F = 0.3$)

We succeeded to find controllers for the inverted pendulum for any value of F down to 0.3, with $T = 0.1$ seconds and $\rho = 0.1$. However, simulations show that the behaviour of the resulting closed loop system is somewhat puzzling. As it is shown in Fig. 8 for $\mathcal{H}^{(K_{0.3}^{(11)})}$, after three swings the pendulum is correctly driven to the goal, but at that point the controller is not able to maintain the plant inside the goal. In fact, the controller let the pendulum fall and makes it do a complete round in order to reach again the upright position. This behaviour is repeated 27 times, before the $K_{0.3}^{(11)}$ makes pendulum stabilize into the goal region.

As already noted in [17], all controllers for underactuated pendulum use two very different strategies to stabilize the system depending on the initial state. When the angle is positive and the speed is negative (and in a suitable range that depends on F), the controller turns directly the pendulum into the upright position. Symmetrically, this also happens when the angle is negative and the speed is positive. Otherwise the controller lets the pendulum fall down to gain enough momentum (or to smoothly slow down it). Therefore, starting from very close states may lead the system to follow very different trajectories. Reducing F squeezes the region of states from which the pendulum is directly turned into the upright position. As Fig. 9 shows, when F is equal to 0.3, we have a rather pathological situation: the frontier between the two strategies lies *inside* the goal region. The controller sometimes is unable to keep the system inside the goal, because disturbances introduced by the simulator make the system cross the frontier between the two strategies. When

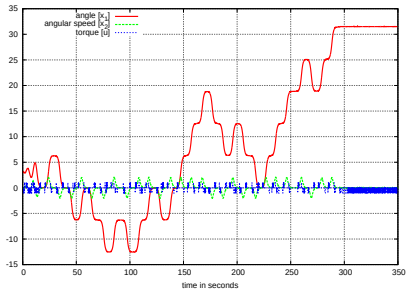

Fig. 8: Simulation for $\mathcal{H}^{(K_{0.3}^{(11)})}$ starting from $(x_1, x_2) = (\pi, 0)$.

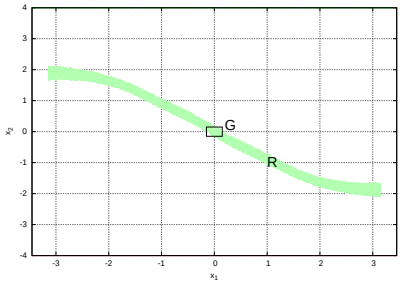

Fig. 9: States turned directly to the goal with $F = 0.3$.

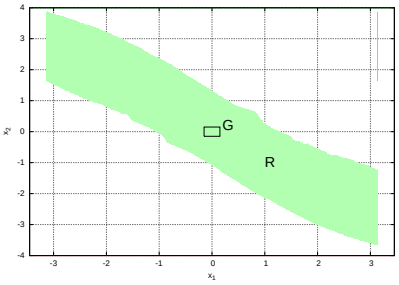

Fig. 10: States turned directly to the goal with $F = 2$.

TABLE I: Experimental Results for inverted pendulum with $F = 0.5$.

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

this frontier lies far enough from the goal (see Fig. 10 for the case $F = 2$), this phenomenon is essentially harmless and leads, at worst, to suboptimal strategies.

C. Overactuated Pendulum ($F = 2$)

When F is greater than 1, finding a control strategy is less challenging. It is worth noting however that, even in this case, our approach allows us to find controllers that hardly can be synthesized by means of traditional analytical methods. In Fig. 6, we show trajectories in the phases space of $\mathcal{H}^{(K_2^{(11)})}$ with $T = 0.01$ seconds, $\rho = 0.05$, and starting values for x_1 are in $\{\frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, 3\}$ and $x_2 = 0$. $\mathcal{H}^{(K_2^{(11)})}$ follows highly non-smooth trajectories: $K_2^{(11)}$ drives the system along an optimal approach to the goal. Before joining this ideal path to the goal, the controller, in order to optimize the set up time, drives the system at the maximum possible "cruising" speed that allows the pendulum to be stopped in the goal. For higher values of F, this cruising speed is even higher.

VI. Conclusions

We presented an automatic methodology to synthesize control software for nonlinear Discrete Time Hybrid Systems. The control software is correct-by-construction with respect both System Level Formal Specifications of the closed loop system and Implementation Specifications, namely the quantization schema. Our experimental results on the inverted pendulum benchmark show the effectiveness of our approach and that we synthesize near optimal controllers that hardly can be designed by using traditional analytical methods of Control Engineering.

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.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *TCS*, 138(1):3 – 34, 1995.

[2] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Softw. Eng.*, 22(3):181–201, 1996.

[3] A. Bemporad. Hybrid Toolbox, 2004. `http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/`.

[4] A. Bemporad and N. Giorgetti. A sat-based hybrid solver for optimal control of hybrid systems. In *HSCC*, LNCS 2993, pp. 126–141, 2004.

[5] M. Benerecetti, M. Faella, and S. Minopoli. Revisiting synthesis of switching controllers for linear hybrid systems. In *CDC-ECC 2011*, pages 4753–4758, dec. 2011.

[6] W. L. Brogan. *Modern control theory (3rd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[7] G. Della Penna, D. Magazzeni, A. Tofani, B. Intrigila, I. Melatti, and E. Tronci. *Automated Generation of Optimal Controllers through Model Checking Techniques*, LNEE 15. Springer, 2008.

[8] G. Frehse. Phaver: algorithmic verification of hybrid systems past hytech. *Int. J. Softw. Tools Technol. Transf.*, 10(3):263–279, 2008.

[9] M. Fu and L. Xie. The sector bound approach to quantized feedback control. *IEEE Trans. on Automatic Control*, 50(11):1698–1711, 2005.

[10] A. Girard. Synthesis using approximately bisimilar abstractions: time-optimal control problems. In *CDC 2010*, pages 5893–5898, dec. 2010.

[11] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2010.

[12] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *STTT*, 1(1):110–122, 1997.

[13] T. A. Henzinger and Pei-Hsin Ho. Algorithmic analysis of nonlinear hybrid systems. In *CAV*, LNCS 939, pages 225–238, 1995.

[14] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond hytech: Hybrid systems analysis using interval numerical methods. In *HSCC*, LNCS 1790, pages 130–144, 2000.

[15] T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. In *ICALP*, pages 582–593, 1997.

[16] Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In *FM*, LNCS 4085, pages 1–15, 2006.

[17] G. Kreisselmeier and T. Birkhölzer. Numerical nonlinear regulator design. *IEEE Trans. on on Automatic Control*, 39(1):33–46, 1994.

[18] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Undecidability of quantized state feedback control for discrete time linear hybrid systems. In *Proc. of ICTAC*, LNCS 7521, pages 243–258. Springer, 2012.

[19] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Synthesis of quantized feedback control software for discrete time linear hybrid systems. In *CAV*, LNCS 6174, pages 180–195, 2010.

[20] M. Mazo, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *CAV*, LNCS 6174, pages 566–569, 2010.

[21] M. Jr Mazo and P. Tabuada. Symbolic approximate time-optimal control. *Systems & Control Letters*, 60(4):256–263, 2011.

[22] G. Pola, A. Girard, and P. Tabuada. Symbolic models for nonlinear control systems using approximate bisimulation. In *Decision and Control, 2007 46th IEEE Conference on*, pages 4656 –4661, dec. 2007.

[23] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[24] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *CDC*, pages 4607–4612 vol. 5. IEEE, 1997.