Linearising Discrete Time Hybrid Systems

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

***Abstract*—Model Based Design approaches for embedded systems aim at generating correct-by-construction control software, guaranteeing that the closed loop system (controller and plant) meets given system level formal specifications.This technical note addresses control synthesis for safety and reachability properties of possibly non-linear discrete time hybrid systems. By means of a syntactical transformations that requires non-linear terms to be Lipschitz continuous functions,we over-approximate non-linear dynamics with a linear system whose controllers are guaranteed to be controllers of the original system.We evaluate performance of our approach on meaningful control synthesis benchmarks, also comparing it to a state-of-the-art tool.**

I. Introduction

Many embedded systems are indeed Software Based Control Systems (SBCSs).A SBCS consists of two main subsystems:the controller and the plant.The controller typically consists of control software that, at discrete time instants, reads sensor outputs from the plant and sends commands to plant actuators in order to guarantee that the closed loop system meets given system level formal specifications.Software generation from modelsand formal specifications forms the core of*Model Based Design* of embedded software [21].This approach is particularly interesting for SBCSssince in such a case specifications are much easierto define than the control software itself. The long term goal is to generate correct-by-construction control software from the plant model (as a hybrid system) and from formal specifications for the closed loop system behaviour. Following this approach, several effective tools and techniques for control software synthesis have been introduced for linear and piecewise affine hybrid systems. Here, we address control synthesis for *discrete time*, *hybrid*, *non-linear* systems, as defined in [2]. Synthesised controllers are correct-by-construction with respect to some *safety* and *reachability* specifications of the closed loop system behaviour. Our results apply to continuous time systems on the basis of a right choice of the time discretisation step. This choice depends on upper bounds of errors in computing numerical solutions to Differential Algebraic Equations (DAE) (see e.g. [6]). In particular, as in the context of hybrid systems simulation, the time step must ensure that the solver deals properly with mode jumps [37].

a) Contributions: In this technical note, we present an automatic procedure (implemented in a publicly available software tool)that *over-approximates* (i.e., possibly allowing more behaviours than) a *non-linear* discrete time hybrid system $\mathcal{H}$ by means of a piecewise affine system $\mathcal{L}_{\mathcal{H}}$ 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 the plant model $\mathcal{L}_{\mathcal{H}}$as input to a tool dealing with piecewise affine systems (such as [7], [26]). The system $\mathcal{L}_{\mathcal{H}}$ is automatically computed from $\mathcal{H}$ by *syntactically* transforming the transition relation $N$of $\mathcal{H}$ into a transition relation $\bar{N}$ containing linear constraints only. We eliminate each non-linear function occurring in N in two independent steps: 1) we under- and over- approximate each non-linear sub-term f of N by means of piecewise affine functions f^-, f^+, such that for all x in a bounded domain D, $f^-(x) \leq f(x) \leq f^+(x)$,2) every occurrence of $f(x)$ in N is then syntactically substituted by constraints satisfied by all values in the interval $[f^-(x), f^+(x)]$. Our linearisation algorithm finds arbitrarily precise approximations of a non-linear Lipschitz continuous function f: for all ε it yields as output f^-, f^+ such that for all x in D, $|f^+(x)-f^-(x)|$ is less than or equal to ε. The price to pay in this over-approximation is the increasing of nondeterminism in the system behaviour. Even though our transformation in the worst case is exponential in the number of variables occurring in a non-linear subterm, it comes out to be effective in practical cases, as it can be applied to all non-linear sub-terms separately, and usually, such sub-terms involve a small number of variables.We finally demonstrate, on the inverted pendulum benchmark, feasibility of our approach and we compare its performances to the tool Pessoa [29].

b) Related Work: This work extends [2], in which it is assumed that over-approximations of non-linear sub-terms have to be provided by the user. Over-approximations of non-linear hybrid systems with linear hybrid systems have been considered in [19], [17]. Such works focus on verification rather than control synthesis. Our linearisation algorithm is in the same spirit of interval analysis [1] and global optimisation [22]. To limit non-determinism arising from system linearisation andthe number of segments of over-approximating functions, we follow an approach based on LP optimisation and piecewise affine functions. Building on [28], this work provides automatic control synthesis for Discrete Time Hybrid Systems (DTHSs). We can classify dynamical systems with respect to their time model (discrete or continuous), dynamics (linear, piecewise affine, non-linear), or hybrid class (non-hybrid, switched, or hybrid with inter-sampling mode jumps). With respect to dynamics, DTHSs extends Discrete Time Linear Hybrid Systems. Control synthesis for non-hybrid non-linear systems (discrete [23] or continuous time [33]) has been investigated for a long time in Control Engineering. Verification and control synthesis for timed and linear hybrid automata has been widely investigated (see e.g. [5], [18], [20], [40], [10], [9]).Control synthesis forpiecewise affine discrete time hybrid systems has been investigated in [41], [7], [8].More recently, abstraction based control synthesis has become a popular and fruitful research area [38], [29], [28]. Focusing on switched (or sampled) systems,several control synthesis procedure has been devised [15], [39], [25]. These works have been extended to the non-linear case, usually stemming from

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

some linearisation or hybridisation technique [14], [42], [34], [12]. Focusing on switched systems (both in continuous and discrete time models), further techniques have been devised to make synthesised controllers guarantee some non-functional requirement of the closed-loop system, such as optimality with respect to some cost function[30], [36], [13], [35], [3], or controller robustness [14], [24], [16].Our work is motivated by the fact that such works do not handle systems that are at the same time non-linear and hybrid, as it is the case of DTHSs.

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 ofreals (*continuous* variables) 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 $\prod_{x\in X}\mathcal{D}_x$.A lower case letter x can also denote a vector of n values $x_1,\dots,x_n$. We use $|x|$ to denote the euclidean norm of x. For a matrix x of size $n\times m$, x_i denotes the i^{th} row of x, i.e. the vector $x_{i,1},\dots,x_{i,m}$.Along the same lines, a hyperinterval $[a_1,b_1]\times[a_2,b_2]\times\ldots\times[a_n,b_n]\subseteq\mathbb{R}^n$ will be denoted just by $[a,b]$, where $a = a_1,a_2,\dots,a_n$ and $b = b_1,b_2,\dots,b_n$. As usual, if $a,b\in\mathbb{R}^n$ and $\lambda\in\mathbb{R}$, $a+b$ will denote $a_1+b_1,\dots,a_n+b_n$, λa denotes $\lambda a_1,\dots,\lambda a_n$, and $|[a,b]|$ its diameter $|b-a|$.If x is a $n\times m$ matrix and y is a $p\times m$ matrix, $z = x\oplus y$ is a $(n+p)\times m$ matrix, such that $z_i = x_i$ if $1\le i\le n$, and $z_i = y_i$ for $n+1\le i\le n+p$. A function $f: D\mapsto\mathbb{R}$ is *Lipschitz continuous* if there exists $\lambda\in\mathbb{R}$ such that for all $x,y\in D$, $|f(x)-f(y)|\le\lambda|x-y|$.

a) Predicates: An *expression* $E(X)$ over a set of variables X is an expression of theform $\sum_{i\in[m]}a_i f_i(X)$, where $f_i(X)$ are possibly non-linearfunctions in which all variables occurring in f belong to X and a_i are rational constants.For example $3\sin x$, x^y, x are expressions over$\{x,y\}$.Our notion of linearity is merely *syntactical*: for us x is a linear expression, while$x+\sin x-\sin x$ is not, even though they "semantically" denote thesame function.$E(X)$ is a *linear expression* if it is a linear combination of variables, that is $E(X) = \sum_{i\in[n]}a_i x_i$.A *constraint* is an inequality of the form $E(X)\le b$, where b is arational constant. We will 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 $(x\ge a)\wedge(x\le b)$.A constraint is *linear* if $E(X)$ is a linear expression.A *predicate* (resp. *linear predicate*) is a boolean formula with constraints (resp. linear constraints) as atoms.A *conjunctive predicate* is a conjunction of constraints. 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 thepredicate $(y=0)\vee C(X)$.Similarly, $\bar{y}\to C(X)$ denotes $(y=1)\vee C(X)$.A *guarded predicate* is a conjunction of either constraints or guardedconstraints.A *valuation* over a list of variables X is a function v that maps eachvariable $x\in X$ to a value $v(x)\in\mathcal{D}_x$.Given a valuation v, we denote with $x^*\in\mathcal{D}_X$ the sequenceof values $[v(x_1),v(x_2),\dots,v(x_n)]$.By abuse of language, we call valuation (or assignment) also the sequence of values x^*.

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 set of states,A is a 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 admissible actions in s.The LTS $\mathcal{S}$ is said *non-blocking* if for each state s and action a, there exists a successor state $s'\in S$ such that $T(s,a,s')$.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 actionsa_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 π_t the state s_t. Given two LTSs$\mathcal{S}_1 = (S,A,T_1)$ and $\mathcal{S}_2 = (S,A,T_2)$we say that $\mathcal{S}_2$ *over-approximates* $\mathcal{S}_1$(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 $a\in A$.In what follows, let $\mathcal{S} = (S,A,T)$ be an LTS, $G\subseteq I\subseteq S$be, respectively, the *goal* and the *initial* regions of $\mathcal{S}$.A controller restricts the dynamics of an LTS so that all states in the initial region I will eventually reach the goal region G. We formalize controllers as solutions to an LTS controlproblem.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 $dom(K) = \{s\in S\mid\exists a\in A\ K(s,a)\}$ is the set ofstates for which at least a control action is enabled.The *closed loop system* $\mathcal{S}^{(K)}$ is theLTS $(S,A,T^{(K)})$,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 statehas 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 $j(\mathcal{S},\pi,G)$ asfollows.If there exists $n>0$ such that $\pi_n\in G$, then$j(\mathcal{S},\pi,G) = \min\{n\mid n>0\wedge\pi_n\in G\}$.Otherwise, $j(\mathcal{S},\pi,G) = +\infty$.We require $n>0$ since 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$J(\mathcal{S},G,s) = \sup\{j(\mathcal{S},\pi,G)\mid\exists a\in\mathrm{Adm}(\mathcal{S},s),\pi\in\mathrm{Path}(s,a)\}$.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 $\mathcal{S}$ suchthat $I\subseteq\mathrm{dom}(K)$ and for all $s\in\mathrm{dom}(K)$,$J(\mathcal{S}^{(K)},G,s)$ is finite.

III. Discrete Time Hybrid Systems

Discrete Time Hybrid Systems (DTHSs) is a class of hybrid systems whose discrete time dynamics is defined by a predicate N over state variables, input variables (that model *controllable inputs*), and auxiliary variables (that model *uncontrollable inputs*, i.e. disturbances).

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

- X *is a finite sequence of real and discrete*present state *variables.The sequence* X' *of* next state *variables is obtained by decoratingwith* $'$ *all variables in* X.
- U *is a finite sequence of* input *variables.*
- Y *is a finite sequence of* auxiliary *variables.*
- $N(X,U,Y,X')$ *is a predicate over* $X\cup U\cup Y\cup X'$*defining the* transition relation *of the system.*

The semantics of a DTHS $\mathcal{H}$ *is given in terms of the labelledtransition systemLTS*$(\mathcal{H}) = (\mathcal{D}_X,\mathcal{D}_U,\tilde{N})$*,where:*$\tilde{N}: \mathcal{D}_X\times\mathcal{D}_U\times\mathcal{D}_X\to\mathbb{B}$*is a predicate such that*$\tilde{N}(x,u,x')$ *holds*

if and only if $\exists y \in \mathcal{D}_Y N(x,u,y,x')$.*We say that* Discrete Time Hybrid System *(DTHS)* $\mathcal{H}_2$ over-approximates *DTHS*$\mathcal{H}_1$ *when* *LTS*$(\mathcal{H}_1) \sqsubseteq$ *LTS*$(\mathcal{H}_2)$.

The class of Discrete Time Linear Hybrid Systems (DTLHSs) is the subclass of DTHSsobtained by imposing N to be a *linear* predicate. In [27] DTLHSs have been proved to be expressive enough to faithfully encode the dynamics of continuous time Rectangular Hybrid Automata. Our abstraction procedure relies on MILP solvers, that require conjunctive predicates as input. If all variables in X range over a bounded domain, then each linear predicate can be transformed into an equisatisfiable guarded predicate and then into a conjunctive predicate [28]. As a consequence, we consider linear predicates as modelling language for DTLHSs.

A. From Continuous Models to DTHSs

The transition relation of a DTHS can approximate the dynamics of a continuous time system by considering a fixed step numerical integration method (explicit or implicit). Our formal guarantees depend on upper bounds to errors arising from time discretisation.As an example, if plant dynamics is defined by an ODE system of the form $\dot{x} = f(x,u)$, f is Lipschitz continuous, and assuming u constant in a time interval τ, by applying Taylor expansion, we can over-approximate the plant continuous time dynamics by means of a predicate N_τ of the form $x' = x + f(x,u)\tau + d \wedge |d| \leq \frac{\lambda}{2}\tau^2$, where λ is a Lipschitz constant for f and d is an auxiliary variable. If plant dynamics is defined by a DAE, as it is often the case for hybrid systems, in general it could be far from trivial to find such an over-approximation. Several techniques, mainly based on index-reduction, have been devised to solve specific families of DAEs, providing useful upper-bounds on the time discretisation errors(see, e.g. [6, Chapter 10]).The controller *sampling time* T (that is usually a design requirement) is typically greater than the time step τ. We can always choose τ in such a way that $T = t\tau$, for some $t \in \mathbb{N}$. Building on this, the dynamics of a system with sampling time$T$ can be approximated by iterating t times the transition relation N_τ. We consider the transitionrelation $N^t_\tau(X,U,X') \equiv \exists \tilde{X}^{(0)},\ldots,\tilde{X}^{(t)} \bigwedge_{i=0}^{t-1} N_\tau(\tilde{X}^{(i)},U,\tilde{X}^{(i+1)}) \wedge X = \tilde{X}^{(0)} \wedge X' = \tilde{X}^{(t)}$, being $\tilde{X}^{(0)},\ldots,\tilde{X}^{(t)}$ sets of variables not occurring in N_τ. The predicate $N^t_\tau(x,u,x')$ holds if, by holding action u for$t$ transitions of step τ, the system goes from x to x', without violating any constraint on its dynamics. This allowsus to have a sampling time T, while retaining model accuracy ensured by a time step τ.

Example 1. *Let us consider the ideal inverted pendulum in Fig. 1, with fixed pivot point where a massless but rigid rod with length* l *drives a particle with mass* m *in a rotational motion (we call the angle* θ *which origin is in the upright position).The pivot point is subject to different friction forces, depending on the rod being below or above the pivot centre, e.g. due to different material deterioration. We assume friction to be linear w.r.t. to angular velocity with coefficient* μ_1 *above and* μ_2 *below (*$\mu_1 \neq \mu_2$*). Besides gravitational acceleration* g*, the pendulum is subject to a torquing force* u *applied to its pivot point, that can influence its velocity in both directions. Eq. 1 shows the equation of motion for pendulum in Fig. 1.*

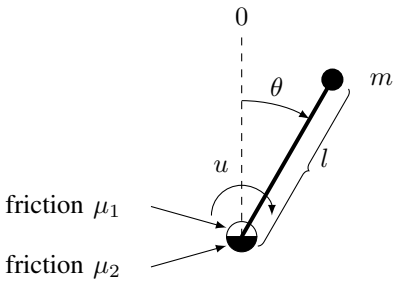

Fig. 1: Inverted Pendulum with Different Frictions.

$$\ddot{\theta} = \frac{g}{l}\sin\theta - \frac{\mu_i}{ml^2}\dot{\theta} + \frac{1}{ml^2}u \tag{1}$$

where i *is* 1 *(resp.* 2*) when the rod is above (resp. below) its pivot centre.By introducing continuous variables* x_1 *for angle* θ *and* x_2 *for angular velocity* $\dot{\theta}$*, we obtain a state space representation equivalent to Eq.* (1)*, defined by first order ODEs in Eq.* (2) *(*$i \in \{1,2\}$*).*

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

Eq. (2) *subsumes two different dynamics due to different friction forces below or above its pivot centre. For this reason, we model it as a hybrid system with two modes.*

Example 2. *Our experimental evaluation considers the discrete time model obtained from the pendulum system in Ex. 1, by applying Euler approximation with time step* τ *to Eq.* (2)*.We introduce the boolean variable* a *for distinguishing between two modes:* $a = 1$ *(resp.* $a = 0$*) means that the pendulum is above (resp. below) its pivot centre. We exploit sinus periodicity by introducing aninteger variable* $q \in \{-1,0,1\}$ *and normalise angle* x_1 *in the range* $[-\pi,\pi]$*. 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* $Y = \{a,q\}$ *is the set of auxiliary variables.Differently from [23], we consider the problem of finding a discretecontroller, whose decisions may be "apply the force clockwise" (*$u = 1$*),"apply the force counterclockwise" (*$u = -1$*), or "do nothing" (*$u = 0$*).Torquing intensity is given as a constant* F*.Finally, the discrete time transition relation* N *is the guarded predicate in Eq.* (3)*.Note that when* $x_1 \in \{-\pi/2,\pi/2\}$*, the transition relation* N*can nondeterministically hit both dynamics (above and below), thus adding more admissible behaviours w.r.t. the original system. This is a safe approach, since a controller for this over-approximation is a fortiori also a controller for the original system.*

$$\begin{array}{l} x_1' = x_1 + \tau x_2 + 2\pi q \ \wedge \\ a \rightarrow \left(\begin{array}{l} x_1 \geq -\frac{\pi}{2} \ \wedge \ x_1 \leq \frac{\pi}{2} \wedge \\ x_2' = x_2 + \tau\frac{g}{l}\sin x_1 - \tau\frac{\mu_1}{ml^2}x_2 + \tau\frac{1}{ml^2}Fu \end{array} \right) \wedge \\ \overline{a} \rightarrow \left(\begin{array}{l} (x_1 \leq -\frac{\pi}{2} \ \vee \ x_1 \geq \frac{\pi}{2}) \wedge \\ x_2' = x_2 + \tau\frac{g}{l}\sin x_1 - \tau\frac{\mu_2}{ml^2}x_2 + \tau\frac{1}{ml^2}Fu \end{array} \right) \end{array} \tag{3}$$

B. Quantized Control Problem for DTHSs

A DTHS control problem $P = (\mathcal{H}, I, G)$ is defined as the LTScontrol problem $(\text{LTS}(\mathcal{H}), I, G)$. The closed loop system is guaranteed to reach G from all initial states, thus we essentially solve what is called a *reachability game* in [38, Chapter 6]. For the sake of simplicity, we do not explicitly consider here safety properties in control problem specifications, as we did in [4]. However, they can be encoded in the transition relation N of $\mathcal{H}$, as forbidden transitions to unsafe states. We search solutions to P among *quantized controllers* (see e.g. [11]). A control problem admits *quantized* solutions if control decisions can bemade by just looking at quantized values.This enables a software implementation for a controller.A *quantization function* γ for a real or integer interval $I = [a, b]$ is anon-decreasing function $\gamma : I \to \mathbb{Z}$, such that $\gamma(I)$ is abounded integer interval.Given a DTHS $\mathcal{H} = (X, U, Y, N)$, a *quantization* Γ isa 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 thetuple $[\gamma_{w_1}(v_1), \ldots, \gamma_{w_2}(v_k)]$.

Example 3. *In our experiments we use uniform quantization functions dividing the domainof 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* Analog-to-Digital *(AD)conversion.Since we have two quantized variables, each one with* b *bits, the number ofquantized states is exactly* 2^{2b}.

Definition 2. *Let* $\mathcal{H} = (X, U, Y, N)$ *be a DTHS,* Γ *be quantization for*$\mathcal{H}$ *and* $\mathcal{P} = (\mathcal{H}, I, G)$ *be a DTHS controlproblem.A* Γ Quantized Feedback Control *(QFC) solution to* $\mathcal{P}$ *is a solution* $K(x, u)$ *to*$\mathcal{P}$ *such that 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. 2 is toturn the pendulum steady to the upright position, starting from any possibleinitial 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* $\rho = 0.1$*,and the initial region is defined by the predicate*$I(X) \equiv (-\pi \leq x_1 \leq \pi) \wedge (-4 \leq x_2 \leq 4)$.

IV. Linear Over-approximation of DTHSs

A DTHS control problem can be reduced to a *Discrete Time Linear Hybrid System* (DTLHS) control problem, by over-approximating the dynamics of $\mathcal{H}$ with the dynamics of a DTLHS $\mathcal{L}_\mathcal{H}$. We present here a fully automatic procedure that *syntactically* transforms the predicate that defines the transition relation of $\mathcal{H}$ into a *linear* predicate, in such a way that $\mathcal{L}_\mathcal{H}$ over-approximates $\mathcal{H}$.This property is the key ingredient of Corollary 4, which ensures that solutions to the control problem $(\mathcal{L}_\mathcal{H}, I, G)$ are guaranteed to be solutions to the originalcontrol problem $(\mathcal{H}, I, G)$.Our procedure over-approximates a Lipschitz continuous non-linear function with a pair of piecewise linear functions. Then we lift this over-approximation to constraints and finally to the predicate that defines the transition relation of a DTHS.

A. Linear Over-approximation of a Non-Linear Function

The building block of our transformation is the elimination of a non-linear sub-term in the predicate N, written as a guarded predicate [28]. We over-approximate a non-linear Lipschitz continuous function $f : D \to \mathbb{R}$ over a bounded region $D \subset \mathbb{R}^n$ with a pair of piecewise affine functions f^-, f^+ such that for all $x \in D$, we have $f^-(x) \leq f(x) \leq f^+(x)$.To make over-approximations tighter, we consider a partition $\{I_1, I_2, \ldots, I_k\}$ that covers D.We then consider a pair of families $(\mathcal{F}^-, \mathcal{F}^+)$ of linear functions ($\mathcal{F}^- = \{f_1^-, \ldots, f_k^-\}$, $\mathcal{F}^+ = \{f_1^+, \ldots, f_k^+\}$), such that for all $i \in [k]$, and for all $x \in I_i$, we have $f_i^-(x) \leq f(x) \leq f_i^+(x)$.

Definition 3. *We say that* $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ *is a* cover *of the closed bounded region* $D \subset \mathbb{R}^n$*, if* $D \subseteq \bigcup_{i=1}^k I_i$*.If all sets* I_i *can be defined by linear predicates,* $\mathcal{I}$ *is a* linear cover*. A set of functions* $\mathcal{F} = \{f_1, \ldots, f_k\}$ *is a* linear family *over* $\mathcal{I}$ *if there exists a matrix* $F \in \mathbb{R}^{k \times (n+1)}$ *such that*$f_i(x) = F_{i,n+1} + \sum_{j=1}^n F_{i,j} x_j$*,for all* $i \in [k]$*.A* linearisation *of* $f : \mathbb{R}^n \to \mathbb{R}$ *on* D *is a pair of linear families*$(\mathcal{F}^-, \mathcal{F}^+)$ *over a linear cover* $\mathcal{I}$ *of* D*,such that for all* $i \in [k]$ *if* $x \in I_i$ *then*$f_i^-(x) \leq f(x) \leq f_i^+(x)$.

By abuse of language, given a matrix F, we will denote with $F_i(x) = f_i(x) = F_{i,n+1} + \sum_{j=1}^n F_{i,j} x_j$. We do not require that $I_i \cap I_j = \varnothing$ for $i \neq j$. All covers we consider have nonempty intersections as they share boundaries with adjacent regions. This is harmless, since if $x \in I_i \cap I_j$, we have that $\max\{f_i^-(x), f_j^-(x)\} \leq f(x) \leq \min\{f_i^+(x), f_j^+(x)\}$, and consequently this defines an over-approximation of f.

B. Linear Over-approximation of a Constraint

We obtain a linear over-approximation of a DTHS $\mathcal{H}$by substituting all non linear sub-terms that appear in the transition relation N of $\mathcal{H}$ with their linearisations.Let $C(V)$ be a constraint in N that contains a non-linear function as a sub-term. Then $C(V)$ has the shape $f(R, W) + E(V) \leq b$, where f is a non-linear function, $R \subseteq V^r$ is aset of n real variables $\{r_1, \ldots, r_n\}$, and $W \subseteq V^d$ is aset of discrete variables.Let $\mathcal{D}_W = \{w_1^*, \ldots, w_m^*\}$ be the set of possible valuations of discrete variables in W. For all $i \in [m]$, we consider the function $f_i(R)$obtained from f, by instantiating discrete variables with w_i^*, i.e. $f_i(R) = f(R, w_i^*)$.Then $C(V)$ is equivalent to the predicate$\bigvee_{i \in [m]} [f_i(R) + E(V) \leq b]$, that can be represented as the following guarded predicate:

$$\bigwedge_{i \in [m]} z_i \to [f_i(R) + E(V) \leq b] \wedge \sum_{i \in [m]} z_i \geq 1$$

by introducing m fresh boolean variables $z_1, \ldots, z_m$.Let $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ be a linear cover of $\mathcal{D}_R$and for each $i \in [k]$, let $(\mathcal{F}_i^-, \mathcal{F}_i^+)$ be a linearisation of f_i on $\mathcal{I}_i$, with $\mathcal{F}_i^- = \{f_{i,1}^-, \ldots, f_{i,k}^-\}$ and $\mathcal{F}_i^+ = \{f_{i,1}^+, \ldots, f_{i,k}^+\}$. Let $I_i(R)$ be a linear predicate such that $r^* \in I_i$ if and only if $I_i(r^*)$.Taking $m \times k$ fresh continuous variables$\tilde{Y} = \{y_{i,j}\}_{i \in [m], j \in [k]}$, and$m \times k$ fresh boolean variables$Z = \{z_{i,j}\}_{i \in [m], j \in [k]}$,we define the guardedpredicate $\bar{C}(V, \tilde{Y}, Z)$ as follows:

$$\bigwedge_{i \in [m], j \in [k]} [z_{i,j} \to [y_{i,j} + E(V) \leq b]] \quad (4)$$

$$\wedge \quad \bigwedge_{i \in [m], j \in [k]} [z_{i,j} \to [f_{i,j}^-(R) \leq y_{i,j} \leq f_{i,j}^+(R)]] \quad (5)$$

$$\wedge \quad \bigwedge_{i \in [m], j \in [k]} [z_{i,j} \to I_j(R)] \wedge \bigwedge_{i \in [m]} \sum_{j \in [k]} z_{i,j} \geq 1 \quad (6)$$

This transformation eliminates the non-linear sub-expression $f(W,R)$ in a constraint $C(V)$, yielding a constraint $\bar{C}(V,\tilde{Y},Z)$ such that if, for some valuation v^* of variables in V, $C(v^*)$ holds, then $\bar{C}(v^*,\tilde{y}^*,z^*)$ holds for some assignment $\tilde{y}^*$ and z^* of variables $\tilde{Y}$ and Z.

Proposition 1. *Let $C(V)$ be a constraint containing a non-linear sub-term. Its linearisation $\bar{C}(V,\tilde{Y},Z)$ is such that $C(V) \Rightarrow \exists \tilde{Y}, Z\ \bar{C}(V,\tilde{Y},Z)$.*

Proof: Let v^* be such that $C(v^*)$ holds. Let r^* be the (induced) assignment to real variables $R = V^r \subseteq V$ and w_i^* be the (induced) assignment to discrete variables $W = V^d \subseteq V$.Hence, $f(r^*, w_i^*) + E(v^*) \leq b$ for some $w_i^* \in \mathcal{D}_W$.Since $\mathcal{I}$ is a cover, $r^* \in I_j$ for some $j \in [k]$, and hence $I_j(r^*)$ holds.Let us consider any assignment to variables in $\tilde{Y}$ and Z such that $z^*_{i,j} = 1$ and $z^*_{p,q} = 0$ for all p,q such that $p \neq i$ or $q \neq j$, $y^*_{i,j} = f(r^*, w^*)$, and arbitrary values to other variables in $\tilde{Y}$. This assignment satisfies constraint (4) because $y^*_{i,j} = f(r^*, w^*)$, and v^* satisfies C. Constraint (5) holds, because $z^*_{p,q} = 0$ for all p,q such that $p \neq i$ or $q \neq j$ and $y^*_{i,j} = f(r^*, w^*)$ and f_i^-, f_i^+ over-approximate f on I_i. Constraint (6) holds because $r^* \in I_i$ and because $z^*_{i,j} = 1$. ■

C. Linear Over-approximation of a DTHS

Given a DTHS $\mathcal{H} = (X,U,Y,N)$, without loss of generality, we can suppose that the transition relation N is a conjunction$\bigwedge_{i\in[n]} C_i(X,U,Y,X')$ of constraints.By applying iteratively the above transformation to each non-linear sub-term occurringin N, we obtain a conjunction of guarded constraints$\bar{N} \equiv \bigwedge_{i\in[n]} \bar{C}_i(X,U,\bar{Y},X')$.By repeatedly applying Prop. 1, it is easy to show that $\bar{N} \Rightarrow N$.Therefore, a *linearisation* $\mathcal{L}_\mathcal{H} = (X,U,\bar{Y},\bar{N})$ of $\mathcal{H}$ is such that its dynamics over-approximates the one of $\mathcal{H}$, thus obtaining the following.

Theorem 2. *Let $\mathcal{H}$ be a DTHS and $\mathcal{L}_\mathcal{H}$ be a linearisation of $\mathcal{H}$.Then LTS($\mathcal{H}$) $\sqsubseteq$ LTS($\mathcal{L}_\mathcal{H}$).*

Theorem 3. *Let $\mathcal{S}_1 = (S,A,T_1)$ and $\mathcal{S}_2 = (S,A,T_2)$ be twoLTSs, 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 $s \in SAdm(\mathcal{S}_1,s) = Adm(\mathcal{S}_2,s)$then K is a solution also to $(\mathcal{S}_1,I,G)$.*

Theorems 2 and 3 (already proven in the context of supervisory control in [32]), guarantee that controllers for the linearised system are controllers for the original system.

Corollary 4. *Let $\mathcal{H}$ be a DTHS and let$\mathcal{L}_\mathcal{H}$ be its linearisation.If K is a solution to $(\mathcal{L}_\mathcal{H},I,G)$ and LTS($\mathcal{H}$) is non-blocking, then K is a solution also to $(\mathcal{H},I,G)$.*

V. Automatic DTHS Linear Over-approximation

The over-approximation of a DTHS $\mathcal{H}$with a DTLHS $\mathcal{L}_\mathcal{H}$ described inSect. IV requires finding linear over- and under-approximations of non-linear sub-expressions occurring in the transition relation of $\mathcal{H}$.This section is devoted to present an algorithm that computes such approximations.

Algorithm 1 Linearisation of a function

Input: A closed bounded region $D = [a,b] \subset \mathbb{R}^n$,Lipschitz continuous function $f : D \to \mathbb{R}$,number of sampling points $m$and maximum linearisation error ε

function *linearize*(D,f,m,ε)
1: $\lambda \leftarrow$ *LipschitzConst*(f,D)
2: $S,\delta \leftarrow$ *sample*(D,m)
3: $C^-(f^-) \leftarrow \bigwedge_{s\in S} f^-(s) \leq f(s) - \lambda|\delta|$
4: $C^+(f^+) \leftarrow \bigwedge_{s\in S} f^+(s) \geq f(s) + \lambda|\delta|$
5: $C^\theta(\theta,f^-,f^+) \leftarrow \bigwedge_{s\in S} f^+(s) - f^-(s) \leq \theta$
6: $err \leftarrow \min\ \theta$ such that $C^- \wedge C^+ \wedge C^\theta$ holds
7: **if** $err \leq \varepsilon$ **then**
8: **return** $\langle err, \{D\}, [f^-], [f^+] \rangle$
9: **else**
10: $\mathcal{I} \leftarrow$ *split*(D)
11: **for all** $i \in [2^n]$ **do**
12: $\langle err_i, \mathcal{I}_i, F_i^-, F_i^+ \rangle \leftarrow$ *linearize*(I_i,f,m,ε)
13: **return** $\langle \max_i err_i, \bigcup_i \mathcal{I}_i, \oplus_i F_i^-, \oplus_i F_i^+ \rangle$

A. Linearisation Algorithm

We aim at finding linearisations that introduce as less-nondeterminism as possible, because nondeterminism makes finding solution to a control problem harder.We start by defining a notion of *linearisation error* to measure how tight a linearisation of a function f is.

Definition 4. *Let $D \subset \mathbb{R}^n$ be a closed region and let $(\mathcal{F}^-,\mathcal{F}^+)$ be a family of linear functionson a cover $\mathcal{I} = \{I_1,I_2,\ldots,I_k\}$ of D.The* linearisation error *ise$(\mathcal{F}^-,\mathcal{F}^+) = \max_{i\in[k]} \max_{x\in I_i} \big(f_i^+(x) - f_i^-(x)\big)$.*

Our *linearize* function in Alg. 1computes a linearisation $(\{f^-\},\{f^+\})$ of a Lipschitz continuous function f over a closed region $D = [a,b] \subseteq \mathbb{R}^n$ (lines 1–6). If $e(\{f^-\},\{f^+\})$ is greater than the required threshold ε, function *split* (line 10) divides D into a finer linear cover $\{I_1,\ldots,I_{2^n}\}$ by dividing each interval $[a_j,b_j]$ into two sub-intervalsof width $(b_j - a_j)/2$ and recursively calls itself on intervals $I_1,\ldots,I_{2^n}$ (line 12). At the end, the resulting linear cover and linearisation is built as union ofcorresponding results for each I_i (line 13).Function *LipschitzConst* (line 1) computes a Lipschitz constant λ for f on D, by using well known techniques (see, e.g. [31]). The choice of λ does not affect algorithm correctness, but the lower λ, the faster *linearize* converges. To speed up convergence, we compute a new value for λ at each recursive call on the hyper-interval D under consideration. Function *sample* (line 2) chooses a set S of m^n points inside D,that are nodes in a uniform n-dimensional grid of size $\delta = |D|/m$. Increasing m of sampling points decreases the linearisation error,at the price of solving harder LP problems.Since constraint $C^- \wedge C^+ \wedge C^\theta$ (line 6)is a conjunction of linear inequalities that involves real variables only, to find a linearisation of f over D, we solve in line 6 a Linear Programming (LP) problem where coefficients of the row vectors f^- and f^+ are decision variables. Constraints$C^-(f^-)$ and $C^+(f^+)$ in lines 3 and

4imply that admissible solutions (f^-, f^+) of the LP problem are coefficients of a linearisation of f.Since λ is a Lipschitz constant for f on D, we have $f(s)-\lambda|\delta| \leq f(x) \leq f(s)+\lambda|\delta|$, for all $s \in S$ and for all $x \in [s, s+\delta]$. The constraint$C^-(f^-)$ implies that $f^-(x) \leq f(s) - \lambda|\delta|andC^+(f^+)$ implies that $f^+(x) \geq f(s) + \lambda|\delta|$.Constraint $C^\theta(f^-, f^+, \theta)$ in line 5 ensuresthat the maximum distance between f^- and f^+, i.e. the linearisationerror, is less or equal than the value of the decision variable θ. Therefore, the optimal value of the LP problem in line 6,is the minimum θ for which there exists a linearisation$(\mathcal{F}^-, \mathcal{F}^+)$(defined by matrices $[f^-]$ and $[f^+]$)of f on the cover of D such that, for all $x \in D$, $f^+(x)-f^-(x) \leq \theta$.

B. Linearisation Algorithm Correctness

Correctness of Alg. 1 is established by Theorem 5 ensuring that function *linearize* eventually terminates by yielding as a result matrices F^- and F^+ that define indeed a linearisation of f over D with linearisation error err.

Theorem 5. *Let $D = [a, b] \subset \mathbb{R}^n$,$f : D \to \mathbb{R}$ be a Lipschitz continuous function,m be the number of sampling points, andε be the maximum tolerable linearisation error. Then:*

1) *function* linearize*(D, f, m, ε) terminates;*
2) *If it returns $\langle err, \mathcal{I}, F^-, F^+\rangle$,then F^- and F^+ define a linearisation $(\mathcal{F}^-, \mathcal{F}^+)$ of f such that $err \leq \varepsilon$.*

Proof: 1) Let us consider$M = \max_{x\in D} f(x)$ and$m = \min_{x\in D} f(x)$. Functions$f^+(x) = M + \lambda|\delta|andf^-(x) = m - \lambda|\delta|$are admissible solutionsto the LP problem solved in line 6.Therefore, linearisation error is bounded by $M - m + 2\lambda|\delta|$. Since λ is a Lipschitz constant for f on D, $M - m$ is bounded by $\lambda|D|$ and hence$err = e(\mathcal{F}^-, \mathcal{F}^+) \leq \lambda|D| + 2\lambda|\delta|$.Denoting with err' (resp. D', δ', λ')the value of the variable err (resp. D, δ, λ)in a recursive call, we have that $err' \leq \lambda'|D'| + 2\lambda|\delta'| < \lambda|D|/2 + \lambda|\delta| = err/2$. Therefore, after k recursive calls, the error err_k, is bounded by $err/2^k$. As a consequence, the maximum nesting k of recursive calls is $\log_2(\lambda|D| + 2\lambda|\delta|)/\varepsilon$. 2)If *linearize* returns $\langle err, \mathcal{I}, F^-, F^+\rangle$, then each $I \in \mathcal{I}$ has been computed in a leaf of its recursion tree, and recursive calls stopped because $err \leq \varepsilon$. Since θ satisfies constraint C^θ, we have that $F^+(x) - F^-(x) \leq \varepsilon$ for all $x \in I$.Now, we show that F^- and F^+ define a linearisation over $\mathcal{I}$. Each $x \in D$ belongs to an interval $J = [s, s + \delta]$, whose vertices arecomputed by function *sample*. A linear function on the closed hyper-interval J has its minimum on a vertex v_0 of J.Since λ is a Lipschitz constant for f on I and the diameter of J is less than $|\delta|$, we have that $f(x) \geq f(v_0) - \lambda|\delta|$.Since f^+ satisfies the constraint C^+ in line 4, we have also $f^+(v_0) \geq f(v_0) + \lambda|\delta|$. By transitivity, $f^+(x) \geq f(x)$.Dually, the same reasoning shows $f^-(x) \leq f(x)$. ∎

As for complexity, in the termination proof, we showed that the nesting k of recursive calls is at worst $(\log_2 \lambda|D| + 2\lambda|\delta|)/\varepsilon$. Since each activation of *linearize* can generate 2^n recursive calls, the maximum number of recursive calls in the worst case is 2^{nk} and hence $((\lambda|D|+2\lambda|\delta|)/\varepsilon)^n$ LP problems have to be solved.

VI. Experimental Results

We implemented our linearisation algorithmin C programming language using IBM ILOG CPLEX for solving LP problems.All experiments have been carried out on anIntel® Xeon® CPU @ 2.83 GHz[1].

TABLE I: Experimental results for$\sum_{i=1}^{t} x_i x_{i+1}$

	$t=2, m=40, x_i \in [-5, 5]$		$t=3, m=10, x_i \in [-2, 2]$	
	n	CPU	n	CPU
monolithic	4768	5d7h49m51s	45436	1d8h40m16s
term-wise	536	1m15s	660	1.1s

A. Evaluation of linearisation algorithm

We start by applying our linearisation algorithm to the function $y \cos x$ (that appears in the dynamics of the inverted pendulum on the cart) on the interval $[0, 2\pi] \times [-2, 2]$.Figs. 2 and 3 show thelinearisation of $y \cos x$ computed using$m = 10$ sampling points and 0.5 as upper bound for linearisation error. Fig. 4 (resp. Fig. 5)shows how the choice of m impacts onthe number of intervals in the computed linear cover (resp. CPU time).Increasing m reduces the number of intervals at the price of solving harderLP problems and this increases CPU time.After a certain threshold, increasing m gives no further improvements(for $y \cos x$ for $m > 40$ the number of intervals does not decreasesignificantly).Function *linearize* is exponential in the number n of dimensions of D.However, our technique allows each non-linear subterm to be linearisedindependently.Let us consider a function $f(x) = \sum_{i=1}^{t} a_i f_i(X_i)$ where for all $i \in [t]$, f_i is non-linear and a constraint $C(X) \equiv f(X) \leq b$. In this case, we can choose to apply function *linearize* to $f$requiring ε as maximum linearisation error (*monolithic* approach) or to iteratively apply the transformationof Sect. IV-B to each subterm $a_i f_i(X)$ requiringε_i as maximum linearisation error, choosing all ε_i insuch a way that $\sum_{i=1}^{t} \varepsilon_i = \varepsilon$ (*term-wise* approach).Let f^-, f^+ (resp. g_i^-, g_i^+) be linearisation computed by the monolithic (resp. term-wise) approach.In both cases, we have that $|f^+(x) - f^-(x)| \leq \varepsilon$ and$|g^+(x) - g^-(x)| \leq \sum_{i=1}^{t} |g_i^+(x) - g_i^-(x)| \leq \sum_{i=1}^{t} \varepsilon_i = \varepsilon$.When the number of variables in each f_i is less than total numberof variables, term-wise approach can greatly reduce both theresulting number of intervals in the computed linear cover and CPU time with respect to monolithic approach.We experimentally show this by considering the function$\sum_{i=1}^{t} x_i x_{i+1}$ for $t = 2, 3$.We apply monolithic approach by requiring $\varepsilon = 1.0$ asupper bound for the linearisation error andterm-wise approach by requiring $\varepsilon_i = \varepsilon/t$ asupper bound for the linearisation error of each f_i.Table I summarizes our results.

B. Control Software Synthesis

We demonstrate how our linearisation technique, combined with thetool *QKS* [28] can be used to automatically generate control software for anon-linear DTHS, by applying this workflow to the pendulum benchmark presented inEx. 2.We set pendulum parameters l and m in such a way that $g/l = 1$ (i.e. $l = g$) and$1/(ml^2) = 1$ (i.e. $m = 1/l^2$). We consider

[1]Software and benchmarks are available as public repositories. linearizer (https://bitbucket.org/mclab/linearizer) is the C library implementing Alg. 1. linearizer-benchmark (https://bitbucket.org/mclab/linearizer-benchmark) contains experimental results. lin4qks (https://bitbucket.org/mclab/lin4qks) transforms output of linearization into *QKS* input.

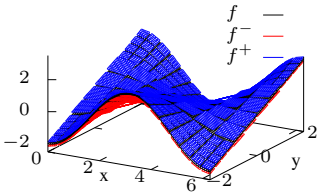

Fig. 2: Linearisation of $y \cos x$

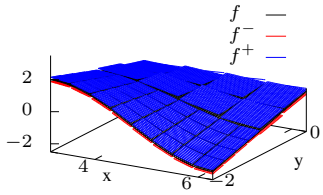

Fig. 3: Linearisation of $y \cos x$ zoomed in $[\pi, 2\pi] \times [-2, 0]$

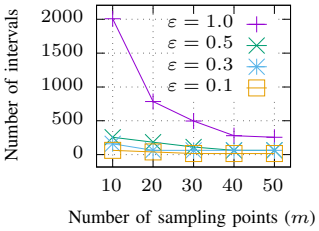

Fig. 4: Experimental results for $y \cos x$ (number of intervals)

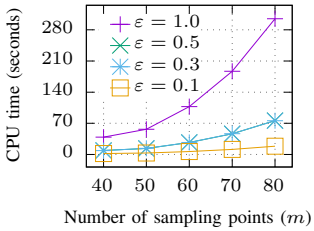

Fig. 5: Experimental results for $y \cos x$ (CPU time)

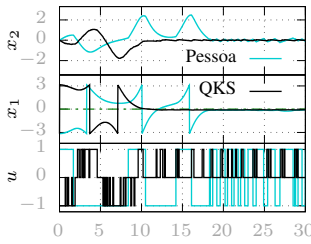

Fig. 6: Switched: Simulation starting from downward position (angle normalised in $[-\pi, \pi]$)

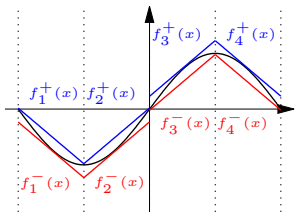

Fig. 7: Linearisation of $\sin x$ with $\varepsilon = 0.5$

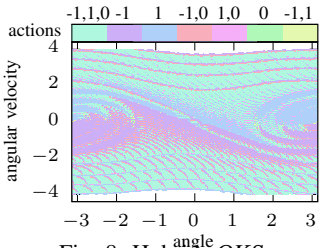

Fig. 8: Hybrid: *QKS* controllable region (actions colour pattern explained)

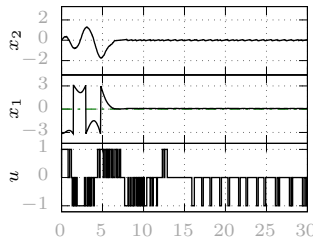

Fig. 9: Hybrid: Simulation starting from downward position (angle normalised in $[-\pi, \pi]$)

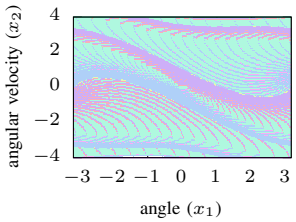

Fig. 10: Switched: Pessoa controllable region, actions colours as in Fig. 8

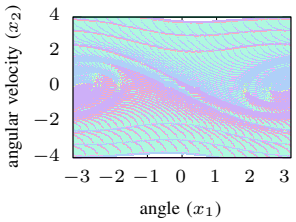

Fig. 11: Switched: *QKS* controllable region ($\tau = 0.01$), actions colours as in Fig. 8

TABLE II: Hybrid: Control software synthesis results

b	ε	n	CPU	MEM	Res.
8	0.5	4	27m40s	76MB	FAIL
9	0.5	4	8d22h13m35s	466MB	PASS
9	1.1	2	1h01m04s	75MB	FAIL

a sampling time $T = 0.1s$. System dynamics is discretised with a time step $\tau = 0.01s$ (i.e., we replicate the transition relation 10 times).The friction coefficients μ_1 isset to 0.01 and μ_2 to 0, and torquing force F to 0.5.We consider the DTHS control problem where the initial and the goal regions I and G are defined as in Ex. 4,and a uniform quantisation as described in Ex. 3.We run *QKS* on the DTLHS controlproblem $(\mathcal{L}^{(\varepsilon)}_{\mathcal{H}}, I, G)$, where$\mathcal{L}^{(\varepsilon)}_{\mathcal{H}}$ is the linearisation of $\mathcal{H}$obtained by applying technique from Sec. V using ε asthe upper bound for the linearisation error of $\sin x$.Fig. 7 shows linearisation of $\sin x$ computed for $\varepsilon = 0.5$.Table II summarises our results for different choices of $b$and ε.Column n shows the number of intervals in the cover, CPU and MEM show the running time (days, hours, minutes, seconds) and RAM usage (MB), and Res. is the result of the control synthesis(PASS or FAIL).Running time in case of FAIL is much smaller than in thecase of PASS thanks to the on-the-fly nature of the *QKS* algorithm [4] thatdetects as soon as possible if a solution cannot be found, without fully computingsystem abstraction. Most of the running time is devoted to compute system abstraction, that in turn depends mainly on b and on the complexity of MILP problems that have to be solved. Data reported in Table III allow us to evaluate the impact on running time of removing modes from the model (from about 9 to less than 1 day) as well as that of removing transition replication (from 23h14m to 1h36m). We simulated the controller found with $b = 9$ and $\varepsilon = 0.5$plugging generated control software into Modelica-based simulator.Fig. 9 shows a simulation starting with the pendulum stationary in the downward position ($x_1 = \pi$, $x_2 = 0$).Controllable region of the synthesised controller is shown in Fig. 8. Different colours mean different sets of actions enabled by thecontroller (see upper stripe of the plot).With $\varepsilon = 1.1$ as linearisation error, *QKS* fails to find a solutionbecause of the nondeterminism introduced in the linearisation process.

C. Control Software Synthesis on Switched Systems

Finally, we compare our approach to the abstraction based control synthesis method implemented in Pessoa [29]. By choosing $\mu_1 = \mu_2 = 0.01$ in the pendulum benchmark (Ex. 1) we get a switched system without mode jumps (other parameters are set as in Sect. VI-B).We run two experiments with *QKS* (with time steps $\tau = 0.1$ and 0.01), both setting $b = 9$ for state quantisation.We run one experiment with Pessoa with the same sampling time and state quantisation step 0.0138 (yielding to 9 bits for x_1 and 10 for x_2), while the time step τ depends on MATLAB integrator[2].Model linearization with $\varepsilon = 0.1$ takes $0.17s$, resulting in 8 intervals for $\sin(x_1)$. Comparison between *QKS* and Pessoa can be summarised as follows.1) *CPU synthesis time:* *QKS* with time step $\tau = 0.1$ computes the controller saving 20% when the *QKS* model uses replication ($\tau = 0.01$), Pessoa saves 91% 2) *MEM:*

[2]Pessoa modeling of the inverted pendulum with fixed pivot point is based on the inverted pendulum on the cart available at https://sites.google.com/a/cyphylab.ee.ucla.edu/pessoa/documentation/examples-1/inverted-pendulum

TABLE III: Switched: *QKS* compared to Pessoa

Tool	τ	CPU	MEM	$\lvert K\rvert$	$h(K)$
Pessoa	ode45	2h00m07s	111MB	19994	40
QKS	0.1	1h36m07s	467MB	21629	38
QKS	0.01	23h14m24s	504MB	21961	38

Pessoa uses much less memory than *QKS* but it requires the MATLAB environment. 3) *Controllable regions*: Figs. 10 and 11 show that the two controllers use different control strategies (colour patterns meaning is the same as in Fig. 8). Understanding different nature of these two control strategies would be interesting, still not in the scope of this paper. 4) *Simulations:* Both controllers are able to reach the goal, even starting from pendulum downright position (Fig. 6 shows Simulink simulations). Here, we used the *QKS* controller with $\tau = 0.01$. 5) *WCET ($h(K)$):* Differences are not significative, as *QKS* uses 38 bits against 40 bits used by Pessoa, but this depends on the Pessoa state quantisation step.

VII. Conclusions

In this technical note, we have presented methods and tools to transform control synthesis problems for discrete time non-linear hybrid system into a controlsynthesis problem for discrete time linear system.Our approach consists of over-approximating a non-linear hybrid system, by means of a syntactic replacement of all non-linear functions occurring in its transition relation with piece-wise affine constraints.We require only Lipschitz continuity of non-linear functions on a bounded region.We implemented our procedureand evaluated performance of the linearisation procedure as well as that of control synthesis. Finally, we have compared our approach to the state-of-the-art tool Pessoa on a switched system.

References

[1] G. Alefelda and G. Mayer. Interval analysis: theory and applications. *Journal of Comput. and Applied Mathematics*, 121:421–464, 2000.

[2] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. Automatic control software synthesis for quantized discrete time hybrid systems. In *Proc. of 51th CDC*, pages 6120–6125. IEEE, 2012.

[3] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. On model based synthesis of embedded control software. In *Proc. of 12th EMSOFT*, pages 227–236. ACM, 2012.

[4] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci. On-the-fly control software synthesis. In *SPIN, LNCS 7976*, pages 61–80, 2013.

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

[6] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Siam, 1998.

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

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

[9] M. Benerecetti, M. Faella, and S. Minopoli. Revisiting synthesis of switching controllers for linear hybrid systems. In *Proc. of 50th CDC-ECC*, pages 4753–4758. IEEE, 2011.

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

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

[12] A. Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.

[13] A. Girard. Low-complexity quantized switching controllers using approximate bisimulation. *Nonlinear Analysis: Hybrid Systems*, 10:34–44, 2013.

[14] A. Girard and S. Martin. Synthesis for constrained nonlinear systems using hybridization and robust controllers on simplices. *IEEE Transactions on Automatic Control*, 57(4):1046–1051, 2012.

[15] 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.

[16] L. Grüne and O. Junge. Global optimal control of perturbed systems. *J. of Optimization Theory and Applic.*, 136(3):411–429, 2007.

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

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

[19] 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.

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

[21] T. A. Henzinger and J. Sifakis. The embedded systems design challenge. In *FM*, LNCS 4085, pages 1–15, 2006.

[22] R. Horst and H. Tuy. *Global Optimization, Deterministic Approaches*. Springer, 1996.

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

[24] J. Liu and N. Ozay. Abstraction, discretization, and robustness in temporal logic control of dynamical systems. In *Proc. of 17th HSCC*, pages 293–302. ACM, 2014.

[25] J. Liu, N. Ozay, U. Topcu, and R. M. Murray. Synthesis of reactive switching protocols from temporal logic specifications. *IEEE Trans. on Automatic Control*, 58(7):1771–1785, 2013.

[26] 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.

[27] 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, 2012.

[28] F. Mari, I. Melatti, I. Salvo, and E. Tronci. Model based synthesis of control software from system level formal specifications. *ACM Trans. Softw. Eng. Methodol.*, 23:Article 6, 2014.

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

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

[31] R. H. Mlandineo. An algorithm for finding the global maximum of a multimodal multivariate function. *Mathematical Programming*, 34(2):188–200, 1986.

[32] T. Moor and J. Raisch. Supervisory control of hybrid systems within a behavioural framework. *Systems & Control Letters*, 38:157–166, 1999.

[33] G. Pola, A. Girard, and P. Tabuada. Symbolic models for nonlinear control systems using approximate bisimulation. In *Proc. of 46th CDC*, pages 4656 –4661. IEEE, 2007.

[34] G. Reissig. Computing abstractions of nonlinear systems. *IEEE Transactions on Automatic Control*, 56(11):2583–2598, 2011.

[35] G. Reissig and M. Rungger. Abstraction-based solution of optimal stopping problems under uncertainty. In *Proc. of 52nd CDC*, pages 3190–3196, 2013.

[36] M. Rungger and O. Stursberg. On-the-fly model abstraction for controller synthesis. In *Proc. of ACC*, pages 2645–2650, 2012.

[37] R. G. Sanfelice and A. R. Teel. Dynamical properties of hybrid systems simulators. *Automatica*, 46:239–248, 2010.

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

[39] D. C. Tarraf, A. Megretski, and Dahleh M. A. Finite approximation of switched homogeneous systems for controller synthesis. *IEEE Trans. on Automatic Control*, 56(5):1140–1145, 2011.

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

[41] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta. Temporal logic control of discrete-time piecewise affine systems. *IEEE Trans. On Automatic Control*, 57:1491–1504, 2012.

[42] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Trans. on Automatic Control*, 57(7):1804–1809, 2012.