Cosimulation-Based Control Synthesis

Adrien Le Coënt(✉), Julien Alexandre dit Sandretto, and Alexandre Chapoutot

U2IS, Institut Polytechnique de Paris, ENSTA Paris, 828 Boulevard des maréchaux, 91762 Palaiseau Cedex, France

adrien.le-coent@ens-cachan.fr, {alexandre,chapoutot}@ensta.fr

Abstract. In this paper, we present a procedure for guaranteed control synthesis for nonlinear sampled switched systems which relies on an adaptive state-space tiling procedure. The computational complexity of the procedure being exponential in the dimension of the system, we explore the use of cosimulation for improving computation times and the scalabity of the method. We apply the procedure on a scalable case study of various dimensions, which is, to our knowledge, a significant step towards the scalabity of formal control synthesis methods with respect to the state of the art.

Keywords: Switched systems · Guaranteed numerical integration · Interval analysis · Symbolic control synthesis · Cosimulation

1 Introduction

Model-based design [20,27] is an effective approach to tackle the increasing complexity of cyber-physical systems. In this approach, physical systems, *e.g.*, plant, are usually modelled by differential equations while computer parts are described by transition systems. Combining these models allows to simulate the behaviour of the whole model of the system in order to predict its behaviour to avoid faults or to synthesize control algorithms.

Safety critical cyber-physical systems require strong guarantees in their execution in order to assess the safety of the mission or the users. Formal methods can produce rigorous evidence for the safety of cyber-physical systems, *i.e.*, based on mathematical reasoning. For example, reachability analysis is an efficient technique to compute the set of reachable states of cyber-physical systems. Once, knowing the set of reachable states, the avoidance of bad states can be formally proved. The main feature of reachability analysis is its ability to propagate sets of values through dynamical systems instead of performing several numerical simulations.

This work was supported by the Chair Complex Systems Engineering - Ecole polytechnique, THALES, DGA, FX, Dassault Aviation, Naval Group Research, ENSTA Paris, Télécom Paris, and Fondation ParisTech.

One weakness of formal verification methods, in particular, reachability analysis, is the scalability with respect to the dimension (number of states) of cyber-physical systems. Applying a cosimulation approach to reachability analysis is attractive since it could broaden the class of problems which can be solved with this technique. A set-based approach of cosimulation to solve differential equations has been defined [21], we explore here its use in the context of formal control synthesis.

Contribution. We propose an extension of a controller synthesis algorithm for a particular class of cyber-physical systems, a.k.a. *nonlinear sampled switched systems*, it relies on a cosimulation approach for the required reachability analysis. A formal definition of the set-based cosimulation is given and then used in order to compute a safe controller for a model of an apartment with a controlled heating.

Related Work. Most of the recent work on set-valued integration of nonlinear ordinary differential equations is based on the upper bounding of the Lagrange remainders either in the framework of Taylor series or Runge-Kutta schemes [2,3,5,7,9,10,12,24]. Sets of states are generally represented as vectors of intervals, a.k.a. *boxes*, and are manipulated through interval arithmetic [25] or affine arithmetic [11]. Taylor expansions with Lagrange remainders are also used in the work of [3], which uses *polynomial zonotopes* for representing sets of states in addition to interval vectors.

The *guaranteed* or *validated* solution of ODEs using interval arithmetic is studied in the framework of Taylor series in [10,13,22,25,26], and Runge-Kutta schemes in [2,5,6,14,18]. The former is the oldest method used in interval analysis community because the expression of the remainder of Taylor series is simple to obtain. Nevertheless, the family of Runge-Kutta methods is very important in the field of numerical analysis. Indeed, Runge-Kutta methods have several interesting stability properties which make them suitable for an important class of problems. The recent work [1] implements Runge-Kutta based methods which prove their efficiency at low orders and for short simulations.

Cosimulation has been extensively studied in the past years [16,17], and has been reported in a number of industrial applications (see [16] for an extensive list domain applications and associated publications). However, most of the uses and tools developed rely on the FMI/FMU standard [4,8,28], which do not allow guaranteed simulation. To our knowledge, guaranteed cosimulation of systems has never been applied on controller synthesis method.

Organization of the Paper. Section 2 presents the mathematical model of sampled switched systems as well as an algorithm to synthesize a safe controller. Set-based simulation and its extension to cosimulation are presented in Sect. 3. Experimental results are presented in Sect. 4 before concluding in Sect. 5.

2 Control Synthesis of Switched Systems

A presentation of the mathematical model of sampled switched systems is given in Sect. 2.1. An algorithm to synthetize safe controllers is described in Sect. 2.2.

2.1 Switched Systems

Let us consider nonlinear switched systems such that

$$\dot{x}(t) = f_{\sigma(t)}(x(t), d(t)) \tag{1}$$

is defined for all $t \geq 0$, where $x(t) \in \mathbb{R}^n$ is the state of the system, $\sigma(\cdot) : \mathbb{R}^+ \longrightarrow U$ is the switching rule, and $d(t) \in \mathbb{R}^m$ is a bounded perturbation. The finite set $U = \{1, \dots, N\}$ is the set of switching modes of the system. We focus on *sampled switched systems*, given a sampling period $\tau > 0$, switchings will periodically occur at times τ, 2τ, ... Switchings depend only on time, and not on states, this is the main difference with hybrid systems.

The switching rule $\sigma(\cdot)$ is thus piecewise constant, we will consider that $\sigma(\cdot)$ is constant on the time interval $[(k-1)\tau, k\tau)$ for $k \geq 1$. We call "*pattern*" a finite sequence of modes $\pi = (i_1, i_2, \dots, i_k) \in U^k$. With such a control pattern, and under a given perturbation d, we will denote by $\mathbf{x}(t; t_0, x_0, d, \pi)$ the solution at time $t \geq t_0$ of the system

$$\begin{aligned} \dot{x}(t) &= f_{\sigma(t)}(x(t), d(t)), \\ x(t_0) &= x_0, \\ \forall j \in \{1, \dots, k\},\ \sigma(t) &= i_j \in U \text{ for } t \in [t_0 + (j-1)\tau, t_0 + j\tau). \end{aligned} \tag{2}$$

We address the problem of synthesizing a state-dependent switching rule $\tilde{\sigma}(\cdot)$ for Eq. (2) in order to verify some properties. This important problem is formalized as follows:

Problem 1 (Control Synthesis Problem). Let us consider a sampled switched system as defined in Eq. (2). Given three sets R, S, and B, with $R \cup B \subset S$ and $R \cap B = \emptyset$, find a rule $\tilde{\sigma}(\cdot)$ such that, for any $x(0) \in R$

- *τ-stability*[1]: $x(t)$ returns in R infinitely often, at some multiples of sampling time τ.
- *safety*: $x(t)$ always stays in $S \backslash B$.

In this problem, S is a *safety* set in which the state should always stay. The set R is a *recurrence* set, in which the state will return infinitely often, it is used to make the computation of a safety controller easier. The set B is an optional obstacle, or *avoid* set. Under the above-mentioned notation, we propose the main procedure of our approach which solves this problem by constructing a rule $\tilde{\sigma}(\cdot)$,

[1] This definition of stability is different from the stability in the Lyapunov sense.

such that for all $x_0 \in R$, and under the unknown bounded perturbation d, there exists $\pi = \tilde{\sigma}(\cdot) \in U^k$ for some k such that:

$$\begin{cases} \mathbf{x}(t_0 + k\tau; t_0, x_0, d, \pi) \in R \\ \forall t \in [t_0, t_0 + k\tau], \quad \mathbf{x}(t; t_0, x_0, d, \pi) \in S \\ \forall t \in [t_0, t_0 + k\tau], \quad \mathbf{x}(t; t_0, x_0, d, \pi) \notin B. \end{cases}$$

Such a law permits to perform an infinite-time state-dependent control. The synthesis algorithm is described in Sect. 2.2 and involves guaranteed set-based integration presented in Sect. 3, the main underlying tool is interval analysis [25].

2.2 Controller Synthesis Algorithm

Before introducing the algorithms to synthetize controller of sampled switched systems, some preliminary definitions will be introduced.

Definition 1. *Let $X \subset \mathbb{R}^n$ be a box of the state space. Let $\pi = (i_1, i_2, \ldots, i_k) \in U^k$. The* successor set *of X via π, denoted by $Post_\pi(X)$, is the image of X induced by application of the pattern π,* i.e., *the solution at time $t = k\tau$ of*

$$\begin{aligned} \dot{x}(t) &= f_{\sigma(t)}(x(t), d(t)), \\ x(0) &= x_0 \in X, \\ \forall t \geq 0, \quad d(t) &\in [d], \\ \forall j \in \{1, \ldots, k\}, \quad \sigma(t) &= i_j \in U \textit{ for } t \in [(j-1)\tau, j\tau). \end{aligned} \tag{3}$$

Note that $Post_\pi(X)$ is usually hard to compute so an over-approximation will be computed instead in order to guarantee rigourous results.

Definition 2. *Let $X \subset \mathbb{R}^n$ be a box of the state space. Let $\pi = (i_1, i_2, \ldots, i_k) \in U^k$. We denote by $Tube_\pi(X)$ the union of boxes covering the trajectories of IVP* (3)*, which construction is detailed in Sect. 3.*

Principle of the Algorithm. We describe the algorithm solving the control synthesis problem for nonlinear switched systems (see Problem 1, Sect. 2.1). Given the input boxes R, S, B, and given two positive integers P and D, the algorithm provides, when it succeeds, a decomposition Δ of R of the form $\{V_i, \pi_i\}_{i \in I}$ verifying the properties:

- $\bigcup_{i \in I} V_i = R$,
- $\forall i \in I,\ Post_{\pi_i}(V_i) \subseteq R$,
- $\forall i \in I,\ Tube_{\pi_i}(V_i) \subseteq S$,
- $\forall i \in I,\ Tube_{\pi_i}(V_i) \bigcap B = \emptyset$.

Decomposition $\Delta = \{V_i, \pi_i\}_{i \in I}$ is thus a set of boxes (V_i) covering R, each box being associated with a control pattern (π_i), and I is a set of indexes used for listing the covering boxes. The sub-boxes $\{V_i\}_{i \in I}$ are obtained by repeated bisection to produce a paving of R. At first, function *Decomposition* calls sub-function *Find_Pattern* which looks for a pattern π of length at most P such that $Post_\pi(R) \subseteq R$, $Tube_\pi(R) \subseteq S$ and $Tube_\pi(R) \bigcap B = \emptyset$. If such a pattern π is found, then a uniform control over R is found (see Fig. 1(a)). Otherwise, R is divided into two sub-boxes V_1, V_2, by bisecting R w.r.t. its longest dimension. Patterns are then searched to control these sub-boxes (see Fig. 1(b)). If for each V_i, function *Find_Pattern* manages to get a pattern π_i of length at most P verifying $Post_{\pi_i}(V_i) \subseteq R$, $Tube_{\pi_i}(V_i) \subseteq S$ and $Tube_{\pi_i}(V_i) \bigcap B = \emptyset$, then it is a success and algorithm stops. If, for some V_j, no such pattern is found, the procedure is recursively applied to V_j. It ends with success when every sub-box of R has a pattern verifying the latter conditions, or fails when the maximal depth of decomposition D is reached. The algorithmic form of functions *Decomposition* and *Find_Pattern* are given in Algorithm 1 and Algorithm 2 respectively.

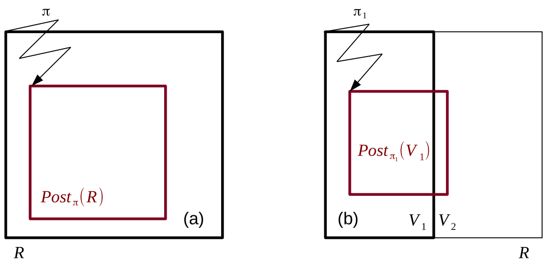

Fig. 1. Principle of the bisection method.

Algorithm 1. Algorithmic form of Function *Decomposition*.

```
Function: $Decomposition(W, R, S, B, D, P)$

   Input: A box $W$, a box $R$, a box $S$, a box $B$, a degree $D$ of bisection, a length $P$
of input pattern
   Output: $\langle\{(V_i, \pi_i)\}_i, True\rangle$ or $\langle\_, False\rangle$

$(\pi, b) := Find\_Pattern(W, R, S, B, P)$
if $b = True$ then
   return $\langle\{(W, \pi)\}, True\rangle$
else
   if $D = 0$ then
      return $\langle\_, False\rangle$
   else
      Divide equally $W$ into $(W_1, W_2)$
      for $i = 1, 2$ do
         $(\Delta_i, b_i) := Decomposition(W_i, R, S, B, D-1, P)$
      end for
      return $(\bigcup_{i=1,2} \Delta_i, \bigwedge_{i=1,2} b_i)$
   end if
end if
```

Our control synthesis method being well defined, we introduce the main algorithm of this paper, stated as follows:

Proposition 1. *Algorithm 1 with input (R, R, S, B, D, P) returns, when it successfully terminates, a decomposition $\{V_i, \pi_i\}_{i \in I}$ of R which solves Problem 1.*

Proof. Let $x_0 = x(t_0 = 0)$ be an initial condition belonging to R. If the decomposition has terminated successfully, we have $\bigcup_{i \in I} V_i = R$, and x_0 thus belongs to V_{i_0} for some $i_0 \in I$. We can thus apply the pattern π_{i_0} associated to V_{i_0}. Let us denote by k_0 the length of π_{i_0}. We have:

- $\mathbf{x}(k_0\tau; 0, x_0, d, \pi_{i_0}) \in R$,
- $\forall t \in [0, k_0\tau], \quad \mathbf{x}(t; 0, x_0, d, \pi_{i_0}) \in S$,
- $\forall t \in [0, k_0\tau], \quad \mathbf{x}(t; 0, x_0, d, \pi_{i_0}) \notin B$.

Let $x_1 = \mathbf{x}(k_0\tau; 0, x_0, d, \pi_{i_0}) \in R$ be the state reached after application of π_{i_0} and let $t_1 = k_0\tau$. State x_1 belongs to R, it thus belongs to V_{i_1} for some $i_1 \in I$, and we can apply the associated pattern π_{i_1} of length k_1, leading to:

- $\mathbf{x}(t_1 + k_1\tau; t_1, x_1, d, \pi_{i_1}) \in R$,
- $\forall t \in [t_1, t_1 + k_1\tau], \quad \mathbf{x}(t; t_1, x_1, d, \pi_{i_1}) \in S$,
- $\forall t \in [t_1, t_1 + k_1\tau], \quad \mathbf{x}(t; t_1, x_1, d, \pi_{i_1}) \notin B$.

We can then iterate this procedure from the new state

$$x_2 = \mathbf{x}(t_1 + k_1\tau; t_1, x_1, d, \pi_{i_1}) \in R.$$

This can be repeated infinitely, yielding a sequence of points belonging to R $x_0, x_1, x_2, \dots$ attained at times $t_0, t_1, t_2, \dots$, when the patterns $\pi_{i_0}, \pi_{i_1}, \pi_{i_2}, \dots$ are applied.

We furthermore have that all the trajectories stay in S and never cross B:

$$\forall t \in \mathbb{R}^+, \exists k \geq 0, \ t \in [t_k, t_{k+1}]$$

and

$$\forall t \in [t_k, t_{k+1}], \ \mathbf{x}(t; t_k, x_k, d, \pi_{i_k}) \in S, \ \mathbf{x}(t; t_k, x_k, d, \pi_{i_k}) \notin B.$$

The trajectories thus return infinitely often in R, while always staying in S and never crossing B. □

Remark 1. Note that it is possible to perform reachability from a set R_1 to another set R_2 by computing $Decomposition(R_1, R_2, S, B, D, P)$. The set R_1 is thus decomposed with the objective to send its sub-boxes into R_2, *i.e.*, for a sub-box V of R_1, patterns π are searched with the objective $Post_\pi(V) \subseteq R_2$.

Remark 2. The search space of control patterns is the set of patterns of length at most P, *i.e.* $U \cup U^2 \cup \dots U^P$. In a practical way, function $Find\_Pattern$ tests control patterns of length 1, then control patterns of length 2, iteratively up to length P. Patterns of length i are generated as combinatorial i-tuples. The set of

Algorithm 2. Algorithmic form of Function *Find_Pattern*.

```
Function: $Find\_Pattern(W, R, S, B, P)$

  Input: A box $W$, a box $R$, a box $S$, a box $B$, a length $P$ of input pattern
  Output: $\langle \pi, True \rangle$ or $\langle \_, False \rangle$

for $i = 1 \dots P$ do
  $\Pi := U^i$ (the set of input patterns of length $i$)
  while $\Pi$ is non empty do
    Select $\pi$ in $\Pi$
    $\Pi := \Pi \setminus \{\pi\}$
    if $Post_\pi(W) \subseteq R$ and $Tube_\pi(W) \subseteq S$ and $Tube_\pi(W) \bigcap B = \emptyset$ then
      return $\langle \pi, True \rangle$
    end if
  end while
end for
return $\langle \_, False \rangle$
```

patterns of length i is U^i, its size is N^i. The complexity of function *Find_Pattern* is thus exponential with the length of control patterns P. The value of P leading to successful decompositions is unknown and depends on each system, but in most cases $P = 4$ leads to successful control synthesis. Longer sequences might be required if the dynamics is slow.

3 Set-Based Cosimulation

In this section, we explain how the *Post* and *Tube* operators can be computed in a distributed way through a cosimulation approach. We first explain the principle of interval analysis and standard guaranteed integration, we then suppose that the system can be written as the composition of components and explain our method for guaranteed cosimulation. In order to ease the reading of this section, we omit the notation of the switched modes σ and control sequences π associated to the *Post* and *Tube* operators.

Before presenting the details of interval analysis and cosimulation, let us introduce the following time periods:

- τ is the switching period,
- H is the communication period,
- h is the simulation period (or integration time-step).

We suppose that $h \le H \le \tau$, H is a multiple of h, and τ is a multiple of H. Consider $H = kh$ and $\tau = KH$ with $k, K \in \mathbb{N}_{>0}$, and an initial time t_0. On time intervals $[t_0, t_0 + \tau)$, the switching mode is constant. In case of cosimulation a model of cyber-physical systems is broken down into different *Simulation Units* (SU). Those SUs will exchange information at periodic rate, *i.e.*, at times $t_0, t_0 + H, \ldots, t_0 + KH$.

3.1 Interval Analysis

In this section, the main set-based tools that are required in this paper are presented.

Interval Artithmetic. The simplest and most common way to represent and manipulate sets of values is with *intervals*, see [25]. An interval $[x_i] = [\underline{x_i}, \overline{x_i}]$ defines the set of reals x_i such that $\underline{x_i} \leq x_i \leq \overline{x_i}$. $\mathbb{IR}$ denotes the set of all intervals over reals. The size or the width of $[x_i]$ is denoted by $w([x_i]) = \overline{x_i} - \underline{x_i}$.

Interval arithmetic extends to $\mathbb{IR}$ elementary functions over $\mathbb{R}$. For instance, the interval sum, *i.e.*, $[x_1]+[x_2] = [\underline{x_1}+\underline{x_2}, \overline{x_1}+\overline{x_2}]$, encloses the image of the sum function over its arguments. In general, an arithmetic operation $\diamond = \{+, -, \times, \div\}$ is associated to its interval extension such that:

$$[a] \diamond [b] \subset [\min\{\underline{a} \diamond \underline{b}, \overline{a} \diamond \underline{b}, \underline{a} \diamond \overline{b}, \overline{a} \diamond \overline{b}\}, \max\{\underline{a} \diamond \underline{b}, \overline{a} \diamond \underline{b}, \underline{a} \diamond \overline{b}, \overline{a} \diamond \overline{b}\}].$$

An interval vector or a *box* $[\mathbf{x}] \in \mathbb{IR}^n$, is a Cartesian product of n intervals. The enclosing property basically defines what is called an *interval extension* or an *inclusion function*.

Definition 3 (Inclusion function). *Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$, then $[f]: \mathbb{IR}^n \to \mathbb{IR}^m$ is said to be an extension of f to intervals if*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \quad [f]([\mathbf{x}]) \supseteq \{f(\mathbf{x}), \mathbf{x} \in [\mathbf{x}]\}.$$

It is possible to define inclusion functions for all elementary functions such as $\times$, $\div$, sin, cos, exp, etc. The *natural inclusion function* is the simplest to obtain: all occurrences of the real variables are replaced by their interval counterpart and all arithmetic operations are evaluated using interval arithmetic. More sophisticated inclusion functions such as the centered form, or the Taylor inclusion function may also be used (see [19] for more details).

Combining the inclusion function and the rectangle rule, integral can be bounded following:

$$\int_a^b f(x)\, dx \in (b-a).[f]([a,b]).$$

Set-Based Simulation. Also named validated simulation or reachability, set-based simulation aims to compute the reachable tube of an Initial Value Problem with Ordinary Differential Equation (IVP-ODE) with a set-based approach and validated computations.

When dealing with validated computation, mathematical representation of an IVP-ODE is as follows:

$$\begin{cases} \dot{y}(t) = f(t, y(t), d(t)) \\ y(0) \in [y_0] \subseteq \mathbb{R}^n \\ d(t) \in [d] \subseteq \mathbb{R}^m. \end{cases} \tag{4}$$

We assume that $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is continuous in t and globally Lipschitz in y, so Eq. (4) admits a unique solution for a given continuous perturbation trajectory. We furthermore suppose that d is bounded in the box $[d]$.

The set (expressed as a box) $[y_0]$ of initial conditions is usually used to model some (bounded) uncertainties. The set $[d]$ is used to model (bounded) perturbations. For a given initial condition $y_0 \in [y_0]$, and a given perturbation $d \in [d]$, the solution at time $t > 0$ when it exists is denoted $y(t; y_0, d)$. The goal, for *validated numerical integration* methods, is then to compute the set of solutions of Eq. (4), *i.e.*, the set of possible solutions at time t given the initial condition in the set of initial conditions $[y_0]$ and the perturbation lying in $[d]$:

$$y(t; [y_0], [d]) = \{y(t; y_0) \mid y_0 \in [y_0],\ d(t) \in [d]\}. \tag{5}$$

Validated numerical integration schemes, exploiting set-membership framework, aims at producing the solution of the IVP-ODE that is the set defined in Eq. (5). It results in the computation of an over-approximation of $y(t; [y_0], [d])$.

The use of set-membership computation for the problem described above makes possible the design of an inclusion function for $[y](t; [y_0], [d])$, which is an over-approximation of $y(t; [y_0], [d])$ defined in Eq. (5). To do so, let us consider a sequence of time instants $t_1, \ \dots, t_K$ with $t_{i+1} = t_i + h$ and a sequences of boxes $[y_1], \ \dots, [y_K]$ such that $y(t_{i+1}; [y_i], [d]) \subseteq [y_{i+1}]$, $\forall i \in [0, K-1]$ are computed. From $[y_i]$, computing the box $[y_{i+1}]$ is a classical 2-step method (see [23]):

- *Phase 1:* compute an a priori enclosure $\mathcal{P}^h([y_i], [d])$ of the set $\{y(t_k; y_i, d) \mid t_k \in [t_i, t_i + h], y_i \in [y_i], d \in [d]\}$, such that $y(t_k; [y_i], [d])$ is guaranteed to exist;
- *Phase 2:* compute a tight enclosure of the solution $[y_{i+1}]$ at time t_{i+1}.

The a priori enclosure $\mathcal{P}^h([y_i], [d])$ computed in Phase 1 is referred to as a Picard box, since its computation relies on the Picard-Lindelöf operator and the Picard theorem (see [2,21] for more details). We omit the theoretical details, but a successful computation of this box ensures the existence and uniqueness of solutions over the time interval $[t_i, t_i + h]$ for the given box of initial conditions $[y_i]$ and perturbation box $[d]$. Two main approaches can be used to compute the tight enclosure in *Phase 2*. The first one, and the most used, is the Taylor method [25,26]. The second one, more recently studied, is the validated Runge-Kutta method [2]. *Guaranteed integration* or *reachability analysis* consists in computing a sequence of boxes that enclose the state of the system on a given time interval. For a given switched mode (the notation being omitted) and perturbation set $[d]$ on time interval $[t, t + \tau]$, given a time integration period h such that $\tau = Kh$, $(k = 1)$, the *Tube* operator is computed as the union of enclosures $\mathcal{P}^h([y_i], [d])$:

$$Tube([y_0]) = \bigcup_{i=1,\dots,K} \mathcal{P}^h([y_i], [d]).$$

The post operator is the tight enclosure given at the final time:

$$Post([y_0]) = [y_K].$$

3.2 Cosimulation of Reachable Sets

The complexity of the computation of the Picard boxes, as well as the tightening of the solutions, is exponential in the dimension of the differential equation considered. As a result, reachability analysis lacks scalability with respect to the dimension of the system. In order to break the exponential complexity of those computations, a cosimulation approach can be used with the aim of computing these objects only on parts of the system.

Cosimulation aims at simulating components of a coupled system separately. In brief, the principle is to enable simulation of the coupled system through the composition of simulators, or simulation units (SUs) [17], each SU being dedicated to only a component of the system. SUs exchange information at some given communication times in order to ensure the simulation error does not grow uncontrollably.

Let us suppose that the dynamics can be decomposed as follows:

$$\begin{aligned}
&\dot{x}_1 \in f_1(t, x_1, u_1) \quad \text{with} \quad x_1(0) \in [x_1^0],\ u_1 \in [u_1],\\
&\dot{x}_2 \in f_2(t, x_2, u_2) \quad \text{with} \quad x_2(0) \in [x_2^0],\ u_2 \in [u_2],\\
&\dots\\
&\dot{x}_m \in f_m(t, x_m, u_m) \quad \text{with} \quad x_m(0) \in [x_m^0],\ u_m \in [u_m],\\
&L(x_1, \dots, x_m, u_1, \dots, u_m) = 0,
\end{aligned}$$

where the state x is decomposed in m components $x = (x_1, \dots, x_m)$, for all $j \in \{1, \dots, m\}$, $x_j \in X_j$, $X_1 \times \cdots \times X_m = \mathbb{R}^d$, and L is a coupling function between the components. The coupling condition $L(x_1, \dots, x_m, u_1, \dots, u_m) = 0$ should hold at all time. From now on, we use index $j \in \{1, \dots, m\}$ to denote subsystem j, and index i to denote a time interval starting at t_i. Note that, in order to increase the accuracy of the method, the decomposition should be made so as to minimize the number of shared variables between sub-systems.

In the most general case, coupling L is an algebraic condition. For our applications, the coupling is supposed to be given explicitly, *i.e*, u_j is given as function of the other state variables: $u_j = K_j(x_1, \dots, x_m)$. Cosimulation then consists in computing *Post* operators for each sub-system separately, and doing a cross product to obtain the global state. To ensure a guaranteed computation, the inputs u_j can be considered as bounded perturbations. The difficulty lies in the determination of the size of the set in which the perturbations evolve, since it has to be determined before performing the simulation of the other sub-systems.

This can be done using the *cross-Picard operator*, introduced in [21]. The purpose of the cross-Picard operator is to over-approximate the solutions of all the sub-systems over a given time interval (the communication period, also called macro-step), using only local computations.

To compute these sets, we start by guessing a rough over-approximation $[p_j]$ of the solutions x_j over the next macro-step. This gives some rough over-approximations $[r_j] = K_j([p_1], \dots, [p_m])$ of the perturbations u_j. We then compute local Picard boxes iteratively, until the proof of validity of the approximations is obtained for all sub-systems.

More precisely, let us denote by $\mathcal{P}_j^H([x_j], [u_j])$ the enclosure of the set of solutions of subsystem j over the time-interval $[t, t+H]$: $\{x_j(t_k; x_j, u_j) \mid t_k \in [t, t+H], x_j \in [x_j], u_j \in [u_j]\}$, where $[x_j]$ and $[u_j]$ are the boxes of initial conditions and perturbation for sub-system j. If we can prove that for all sub-systems $j \in \{1, \dots, m\}$, $\mathcal{P}_j^H([x_j], K_j([p_1], \dots, [p_m])) \subsetneq [p_j]$, then, by application of the Picard theorem, existence and uniqueness of global solutions is ensured for the time interval $[t, t+H]$. Fortunately, this condition is in practice easily met by application of a fixed point algorithm that tightens the rough initial guesses $[p_j]$ (see [21]). Once the Picard boxes are computed and proved safe, each sub-system j can, in parallel, compute its own solution safely on the time interval $[t, t+H]$ by considering u_j as a perturbation lying in $K_j([p_1], \dots, [p_m])$. We denote the cross-Picard operator as the computation of the validated Picard boxes, the result being given as the cross-product of the Picard boxes.

Our approach for guaranteed cosimulation of the *Post* operator over the interval $[t, t+\tau]$ is thus summarized as follows:

1. Compute an over-approximation of the solutions on time interval $[t, t+H]$ (compute the cross-Picard operator),
2. Advance simulation of all subsystems in parallel (using a time step h) until time $t+H$, the inputs are considered as bounded perturbations in the sets returned in Step 1,
3. Update initial conditions and input values,
4. Repeat on interval $[t+H, t+2H]$ until $[t+\tau-H, t+\tau]$.

3.3 Discussion on Meta-parameters

The different time periods involved in the synthesis and cosimulation procedures (h, H, and τ) play a crucial role in the accuracy of the reachability analysis, and thus in the success of the control synthesis. In mere words, a reachability analysis is performed each time a control sequence is tested. Improving the speed of the reachability analysis drastically improves control synthesis computation times, provided that the accuracy is high enough to allow control synthesis.

One of the key aspects is that the frequency at which we update the initial conditions and perturbation sets (the communication frequency $1/H$) should be as small as possible in order to increase the speed of the reachability analysis, but at the cost of the accuracy of cosimulation. The speed increase when using fewer communications is due to the fact that each communication time involves the application of a fixed-point algorithm to validate the perturbation sets, thus taking a non negligible amount of computation. However, using shorter communication periods means that the perturbation sets are smaller, and the cosimulation thus leads to tighter reachable tubes, making the synthesis easier. The largest communication period we can consider is actually the switching period $H = \tau$. If such a large communication period allows enough accuracy for control synthesis purposes, then this would lead to the best computation time gains. However, in practice, the switching period can be too large to avoid communication between switching times. If a communication is necessary between switchings, then, in order to maximize the use of the data exchange, communication frequency should be a multiple of the switching frequency.

We would like to point out that the integration time step h can actually be different for each simulation unit (for reachability analysis of separate components, once the perturbation sets are validated). The integration methods can be essentially different since we can even consider implicit and explicit methods in parallel. The only requirement is that the perturbation sets are proved safe (with the use of Picard operators). This means that complex systems involving stiff and nonstiff dynamics, or linear and nonlinear dynamics, can be divided in such a way that the computation power is dedicated to the more difficult parts to integrate. In our applications, we illustrate the scalability property of the proposed method, but industrial applications involving more complex dynamics could show even better improvements.

4 Experiments

4.1 Case Study

This case study is based on a simple model of a two-room apartment, heated by a heater in one of the rooms (adapted from [15]). Initially of dimension of the state space is 2, the case study is made scalable by concatenating two-room apartments in line, so that each room exchanges heat with its neighbouring rooms, and every other room is equipped with a heater.

In this example, the objective is to control the temperature of all rooms. There is heat exchange between neighbouring rooms and with the environment. The *continuous* dynamics of the system, *the temperature*, is given by

$$\dot{\begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{pmatrix}} = A \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{pmatrix} + B_u.$$

The dimension n is supposed to be even $n = 2m$. The non null coefficients $a_{i,j}$ of matrix A are:

$$\begin{aligned}
a_{1,1} &= -\alpha_r - \alpha_e - \alpha_f u_1 && \\
a_{2i+1,2i+1} &= -2\alpha_r - \alpha_e && i = 1, \ldots, m-1 \\
a_{2i,2i} &= -2\alpha_r - \alpha_e - \alpha_f u_i && i = 1, \ldots, m-1 \\
a_{2m,2m} &= -\alpha_r - \alpha_e && \\
a_{i,i+1} &= a_{i+1,i} = \alpha_r && i = 1, \ldots, 2m-1.
\end{aligned}$$

The non null coefficients $b_{i,j}$ of the input matrix B_u are:

$$\begin{aligned}
b_{2i-1} &= \alpha_e T_e + \alpha_f T_f u_{i+1} && i = 1, \ldots, m \\
b_{2i} &= \alpha_e T_e && i = 1, \ldots, m.
\end{aligned}$$

Here T_i for $i = 1, \ldots, 2m$ is the temperature of room i, and the state of the system corresponds to $T = (T_1, \ldots, T_n)$. The control modes are given by variables u_j for $j = 1, \ldots, m$, each can take the values 0 or 1, depending on whether the heater in room $2j-1$ (for $j = 1, \ldots, m$) is switched off or on. Hence, the number of switched modes is 2^m. Temperature T_e corresponds to the temperature of the environment, and T_f to the temperature of the heaters. The values of the different parameters are as follows: $\alpha_r = 5 \times 10^{-2}$, $\alpha_e = 5 \times 10^{-3}$, $\alpha_f = 8.3 \times 10^{-3}$, $T_e = 10$ and $T_f = 50$.

The control objective is to ensure τ-stability of the temperature in $R = [19, 21] \times \cdots \times [19, 21]$, while ensuring safety in $S = [18, 22] \times \cdots \times [18, 22]$, with a switching period $\tau = 10$. We don't consider any obstacle B in this example, the maximal length of patterns is set to $P = 4$, and the maximum depth of decomposition is $D = 2$.

4.2 Experimental Results

In order to validate our approach, we synthetize the control rule for the problem given in Sect. 4.1 for different number of rooms $n = 2, 4, 6, 8$. The results are gathered in Table 1. All the simulations are performed with the classical method RK4, an explicit Runge-Kutta method with four stages at the fourth order. Our choice for this method is based on its fame and on the fact that to find a control for the case study, an order greater than two is needed. Cosimulation consists in m simulations of systems of dimension two (three with an additional dimension for time). More precisely, if $m \geq 3$, system

$$\dot{\begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{pmatrix}} = A \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{pmatrix} + B_u,$$

for $n = 2m$, is rewritten as m systems:

$$\dot{\begin{pmatrix} T_1 \\ T_2 \end{pmatrix}} = A^1 \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} + B_u^1 + D^1 \left(T_3\right),$$

$$\dot{\begin{pmatrix} T_3 \\ T_4 \end{pmatrix}} = A^2 \begin{pmatrix} T_3 \\ T_4 \end{pmatrix} + B_u^2 + D^2 \begin{pmatrix} T_2 \\ T_5 \end{pmatrix},$$

$$\cdots$$

$$\dot{\begin{pmatrix} T_{2m-1} \\ T_{2m} \end{pmatrix}} = A^m \begin{pmatrix} T_{2m-1} \\ T_{2m} \end{pmatrix} + B_u^m + D^m \left(T_{2m-2}\right),$$

where D^i is a disturbance matrix composed of coefficients of A. One can see that each subsystem is perturbed by the adjacent rooms (there is one adjacent room for the first and last system, and two adjacent rooms for the others). There is one communication per switching period, meaning that $H = 5$ for $\tau = 10$.

These simulations are also performed in parallel using Open Multi-Processing API for Linux. Experiments are done on a bi-processor Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40 GHz with 12 cores each. A time out (T.O.) is fixed at three days, *i.e.*, 4320 min. Computation times seem important but the results are guaranteed and have to be computed only one time and offline.

Table 1. Results of synthesis for problem given in Sect. 4.1: computation times, in minutes, for centralized dynamics, with cosimulation and with parallelized cosimulation.

Number of rooms (n)	Centralized	Cosimulation	Cosimulation in parallel
2	0m43	–	–
4	2m28	2m30	1m58
6	185m	80m	42m
8	T.O	3606m	2072m

4.3 Discussion

Our method shows its efficiency, even with only 4 rooms if parallelization is used. A control rule can be synthetized for 8 rooms with cosimulation while no result can be obtained without our approach before time out. Cosimulation allows a very straightforward parallelization which reduces significantly computation time. Our experiments revealed the necessity of using one communication per switching period for this case study. Using none (communicating only at the beginning of a switching period) led to sets too wide for ensuring τ-stability.

5 Conclusion

In this paper, we presented a procedure for control synthesis that relies heavily on (guaranteed) reachability computations, its scalability being limited by the complexity of set-based integration. We proposed to use a guaranteed cosimulation to improve the control synthesis computation times. We illustrate the scalability of our method on a scalable case-study that shows the efficiency of our approach. As of now, some expertise is required for choosing a communication frequency that allows computation time gains as well as successful control synthesis. We would like to explore the possibility of automating the determination of a good communication frequency in the context of switching systems.

The current implementation of the procedure allows to simulate subsystems in parallel, but the cross-Picard computation (involving repeated applications of Picard operators) is still sequential due to memory management issues. Our future work will be devoted to the development of a parallel implementation of the cross-Picard computation. Such an implementation would hopefully mitigate the cost of communication times in the present procedure.

References

1. Alexandre dit Sandretto, J., Chapoutot, A.: DynIbex. https://perso.ensta-paris.fr/~chapoutot/dynibex/
2. Alexandre dit Sandretto, J., Chapoutot, A.: Validated explicit and implicit Runge-Kutta methods. Reliable Comput. **22**, 79 (2016)
3. Althoff, M.: Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In: Hybrid Systems: Computation and Control, pp. 173–182 (2013)
4. Arnold, M., Clauß, C., Schierz, T.: Error analysis and error estimates for co-simulation in FMI for model exchange and co-simulation v2.0. In: Schöps, S., Bartel, A., Günther, M., ter Maten, E.J.W., Müller, P.C. (eds.) Progress in Differential-Algebraic Equations. DEF, pp. 107–125. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44926-4_6
5. Bouissou, O., Chapoutot, A., Djoudi, A.: Enclosing temporal evolution of dynamical systems using numerical methods. In: Brat, G., Rungta, N., Venet, A. (eds.) NFM 2013. LNCS, vol. 7871, pp. 108–123. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38088-4_8
6. Bouissou, O., Martel, M.: GRKLib: a guaranteed Runge Kutta library. In: Scientific Computing, Computer Arithmetic and Validated Numerics (2006)
7. Bouissou, O., Mimram, S., Chapoutot, A.: HySon: set-based simulation of hybrid systems. In: Rapid System Prototyping. IEEE (2012)
8. Broman, D., et al.: Determinate composition of FMUs for co-simulation. In: 2013 Proceedings of the International Conference on Embedded Software (EMSOFT), pp. 1–12. IEEE (2013)
9. Chen, X., Abraham, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: IEEE 33rd Real-Time Systems Symposium, pp. 183–192. IEEE Computer Society (2012)
10. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_18
11. de Figueiredo, L.H., Stolfi, J.: Self-validated numerical methods and applications. In: Brazilian Mathematics Colloquium Monographs, IMPA/CNPq (1997)

12. Alexandre dit Sandretto, J., Chapoutot, A.: Validated simulation of differential algebraic equations with Runge-Kutta methods. Reliable Comput. **22**, 57 (2016)
13. Dzetkulič, T.: Rigorous integration of non-linear ordinary differential equations in Chebyshev basis. Numer. Algorithms **69**(1), 183–205 (2015). https://doi.org/10.1007/s11075-014-9889-x
14. Gajda, K., Jankowska, M., Marciniak, A., Szyszka, B.: A survey of interval Runge–Kutta and multistep methods for solving the initial value problem. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1361–1371. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68111-3_144
15. Girard, A.: Low-complexity switching controllers for safety using symbolic models. IFAC Proc. Vol. **45**(9), 82–87 (2012)
16. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: state of the art. arXiv preprint arXiv:1702.00686 (2017)
17. Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H.: Co-simulation: a survey. ACM Comput. Surv. (CSUR) **51**(3), 1–33 (2018)
18. Immler, F.: Verified reachability analysis of continuous systems. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 37–51. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_3
19. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, London (2001). https://doi.org/10.1007/978-1-4471-0249-6
20. Jensen, J.C., Chang, D.H., Lee, E.A.: A model-based design methodology for cyber-physical systems. In: 2011 7th International Wireless Communications and Mobile Computing Conference, pp. 1666–1671. IEEE (2011)
21. Le Coënt, A., Alexandre dit Sandretto, J., Chapoutot, A.: Guaranteed cosimulation of cyber-physical systems. hal-02505237 https://hal.archives-ouvertes.fr/hal-02505237 (2020)
22. Lin, Y., Stadtherr, M.A.: Validated solutions of initial value problems for parametric odes. Appl. Numer. Math. **57**(10), 1145–1162 (2007)
23. Lohner, R.J.: Enclosing the solutions of ordinary initial and boundary value problems. In: Computer Arithmetic, pp. 255–286 (1987)
24. Makino, K., Berz, M.: Rigorous integration of flows and ODEs using Taylor models. In: Proceedings of the 2009 Conference on Symbolic Numeric Computation, SNC 2009, pp. 79–84. ACM, New York (2009)
25. Moore, R.E.: Interval Analysis. Series in Automatic Computation. Prentice Hall, Englewood Cliffs (1966)
26. Nedialkov, N.S., Jackson, K.R., Corliss, G.F.: Validated solutions of initial value problems for ordinary differential equations. Appl. Math. Comput. **105**(1), 21–68 (1999)
27. Nielsen, C.B., Larsen, P.G., Fitzgerald, J., Woodcock, J., Peleska, J.: Systems of systems engineering: basic concepts, model-based techniques, and research directions. ACM Comput. Surv. (CSUR) **48**(2), 1–41 (2015)
28. Schierz, T., Arnold, M., Clauß, C.: Co-simulation with communication step size control in an FMI compatible master algorithm. In: Proceedings of the 9th International MODELICA Conference, Munich, Germany, 3–5 September 2012, no. 076, pp. 205–214. Linköping University Electronic Press (2012)