# Minimally Constrained Stable Switched Systems and Application to Co-simulation

Cláudio Gomes[1,3], Raphaël M. Jungers[2], Benoît Legat[2], Hans Vangheluwe[1,3,4]

**1 University of Antwerp, Belgium**
**2 Université Catholique de Louvain, Belgium**
**3 Flanders Make vzw, Belgium**
**3 McGill University, Canada**

*** claudio.gomes@uantwerp.be**

**Abstract**

We propose an algorithm to restrict the switching signals of a constrained switched system in order to guarantee its stability, while at the same time attempting to keep the largest possible set of allowed switching signals.

Our work is motivated by applications to (co-)simulation where numerical stability is a hard constraint, but should be attained by restricting as little as possible the allowed behaviours of the simulators.

We apply our results to certify the stability of an adaptive co-simulation orchestration algorithm, which selects the optimal switching signal at run-time, as a function of (varying) performance and accuracy requirements.

## 1 Introduction

A switched system is defined as

$$x_{k+1} = A_{\sigma_k} x_k : \quad \sigma_k \in \{1, \ldots, m\}, \; A_{\sigma_k} \in \mathcal{A} \tag{1}$$

where $\{1, \ldots, m\}$ is the set of modes, $\sigma_k$ is the mode active at time $k$, and $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n \times n}$ is a set of real matrices. We denote the sequence $\sigma_0, \sigma_1, \ldots, \sigma_k$ as the *switching signal*. $A_{\sigma_k} \in \mathcal{A}$ represents the matrix used to compute $x_{k+1}$ from $x_k$ at time $k$.

Switched systems are widely used to model many dynamical systems in modern engineering including viral mutations in a patient's body [25], *trackability* of malicious agents in a sensor network [28], or scheduling of thermostatically controlled loads (TCLs) [37].

In this paper, we are motivated by a new application [16] in the field of co-simulation. It is a numerical technique to couple multiple simulators, each simulating a part of a coupled system, in order to compute the overall behavior more efficiently [7, 8, 17, 20, 22, 30]. One of the objectives is to leverage different mature simulation tools, even when the details of each simulation tool are unknown [3, 4], and it has been applied in fields such as automotive, electricity distribution, maritime, railway, etc. . . See [14, 18, 42, 44] and references thereof, for applications.

As illustrated in Figure 1, in order to run a co-simulation, each simulator approximates the solution of a differential equation, exchanging values with other simulators at agreed-upon communication times. Between communication time instants, the unknown inputs are approximated with extrapolation functions.

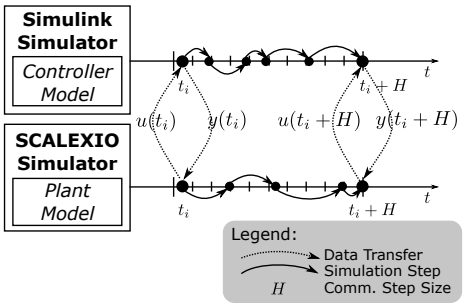



Figure 1: Co-simulation coordination example.

To improve the performance, during a co-simulation each simulator can adapt its policy, by varying:

1. the discretization step size and/or the numerical algorithm (e.g., midpoint method, Runge-Kutta) as a result of error estimates;
2. the input approximation function (e.g., by varying the Lagrange polynomial degree) as a result of the dynamics of the inputs; and
3. the values exchanged and the order in which they are exchanged, as a result of the varying structure of the coupled system being simulated.

We will call *policy sequence* to the sequence of policies taken by all simulators over time.

The ability to adapt allows one to find the best tradeoff between accuracy and computation power, to meet the available time. The applications are not restricted to co-simulation. For example, in Model Predictive Control [12],

the controller needs to be able to simulate the system in real-time.

In the context of (co-) simulation, it is fundamental to ensure that a (co-) simulation method preserves the stability of the system being (co-) simulated. The error dynamics of a (co-) simulation can be modelled as a switched system (Section 5), hence the preservation of stability becomes a problem of deciding the stability of a switched system (System (1)). Furthermore, as the choice of future policies is influenced by the past policies, we consider *constrained switched systems*, a recently developed framework allowing us to model the memory of the system (see Section 2).

If there exist one or more policy sequences that can make the (co-) simulation method unstable, then the simulators have to be forbidden from following these. In the context of constrained switched systems, there are many ways of forbidding a policy sequence, and each way also forbids sequences that do not make the (co-) simulation unstable.

We tackle the problem of how to best forbid policy sequences that make the constrained switched system unstable. We propose that the best solution is to maximize the *entropy* of the stabilized constrained switched system in order to maximize the adaptability of the resulting (co-) simulation method.

In the next section, we formulate the problem of how to forbid "bad" policy sequences (that cause the system to be unstable), while minimizing the number of good sequences that are forbidden as side effect. Then, in Section 3 we propose an algorithm that approximates the solution, and we prove that the it terminates and that the resulting system is stable. Furthermore, we provide a lifting technique that yields better solutions. Section 5 explores applications of our contributions and co-simulation, respectively . Section 6 presents related work and Section 7 concludes.

## 2 Problem Formulation

We introduce the dynamical systems being considered, called *Constrained Switched Systems* [40], and we formulate our research problem.

In practice, some switching signals of System (1) may not be relevant, and a way to represent the sensible ones is required.

For instance, in the switched system described in Example 4, it may not make sense that the Runge-Kutta method is used right after a Forward Euler. This is because the convergence rate of each method is too different to warrant a switch without first taking a step with the Midpoint method.

To model these constraints, we introduce the notion of constrained switched system. When compared to System (1), constrained switched systems incorporate a representation of the allowed switching signals using an automaton [36].

**Definition 1.** *Given a bounded set of matrices $\mathcal{A} = \{A_1, \dots, A_m\}$, we define an* automaton *as a directed and labelled graph $\mathbf{G} = (V, E)$, with nodes $V$ and edges $E$ such that no node has zero ingoing or outgoing degree. Each edge $(v, w, \sigma) \in E$ represents a transition from node $v \in V$ to node $w \in V$, where $\sigma \in \{1, \dots, m\}$ is the* label*, corresponding to $A_\sigma$.*

An example automaton, illustrating possible constraints on the system described in Example 4, is shown in Figure 2.

We say that the switching signal, or word, $s = \sigma_0\sigma_1 \dots \sigma_{k-1}$ is *accepted* by an automaton $\mathbf{G}$ if it corresponds to a path in $\mathbf{G}$, that is, if there exists $v_0, v_1, \dots, v_k \in V$, such that $(v_j, v_{j+1}, \sigma_j) \in E$ for all $j = 0, \dots, k-1$. An accepted word induces an accepted matrix product $A_s = A_{\sigma_{k-1}} \cdots A_{\sigma_1} A_{\sigma_0} \in \mathcal{A}^k$.

For example, the word $(fe, 0.001), (fe, 0.002), (md, 0.002)$ is accepted by the automaton shown in Figure 2. This word induces the matrix product $\tilde{A}_{md,0.002}\tilde{A}_{fe,0.002}\tilde{A}_{fe,0.001}$.

We denote the set of accepted words of length $k$ as $\mathbf{G}_k$, and the set of all words accepted by the automaton as $\mathbf{G}^* = \bigcup_{k=1}^{\infty} \mathbf{G}_k$. Moreover, $\mathbf{G}_k^\circ$ denotes the set of accepted cycles of length $k$.

For example,

$$(fe, 0.001), (fe, 0.002), (md, 0.002) \in \mathbf{G}_3, \text{ and}$$
$$(fe, 0.002), (fe, 0.001) \in \mathbf{G}_2^\circ.$$

One can see that given a word $\sigma(0) \dots \sigma(k-1) \in \mathbf{G}_k$, any sub-word $\sigma(i) \dots \sigma(j)$ for any $0 \le i \le j < k$, satisfies $\sigma(i) \dots \sigma(j) \in \mathbf{G}_{j-i+1}$. Moreover, since every node has at least one outgoing edge in Definition 1, for any $k' > k$, there exists $\sigma(k) \dots \sigma(k'-1)$ such that $\sigma(0) \dots \sigma(k'-1) \in \mathbf{G}_{k'}$.

**Definition 2** (CSS)**.** *Given a set of matrices $\mathcal{A} = \{A_1, \dots, A_m\}$, and an automaton $\mathbf{G} = (V, E)$, we define a* constrained switched system *(CSS) $S = \langle \mathcal{A}, \mathbf{G} \rangle$ as a system where the variable $x_k$ satisfies:*

$$x_{k+1} = A_{\sigma_k} x_k : \quad \sigma_0 \dots \sigma_{k-1} \in \mathbf{G}_k. \tag{2}$$

We say that System (2) is *stable* iff

$$\lim_{k\to\infty} \|x_k\| = \lim_{k\to\infty} \left\|A_{\sigma_{k-1}} \cdots A_{\sigma_0} x_0\right\| = 0,$$

for any word $\sigma_0 \dots \sigma_{k-1} \in \mathbf{G}_k$ and any $x_0 \in \mathbb{R}^n$.

To determine the stability of a CSS, we introduce the *constrained joint spectral radius*.

**Definition 3** ( [11, Definition 1.2])**.** *The* constrained joint spectral radius *is defined as*

$$\hat{\rho}(S) = \lim_{k\to\infty} \hat{\rho}_k(S) \text{ where } \hat{\rho}_k(S) = \sup_{w\in\mathbf{G}_k} \|A_w\|^{\frac{1}{k}},$$

*and $\|\cdot\|$ is any matrix norm that satisfies the sub-multiplicative property.*

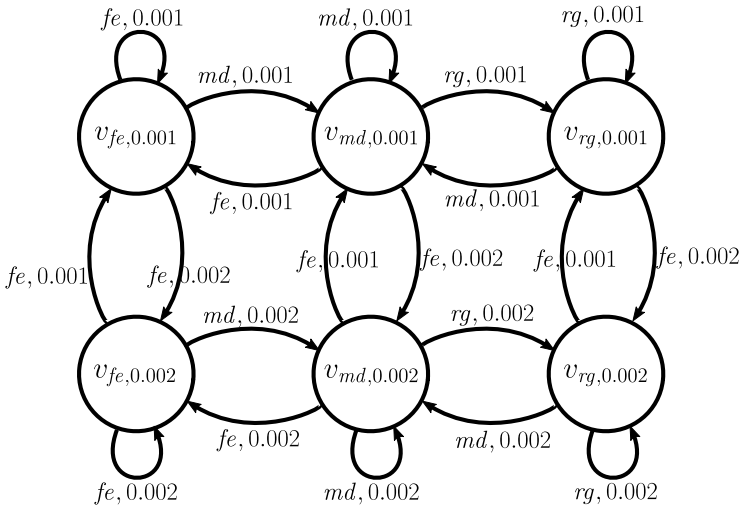



Figure 2: Example automaton for Example 4.

**Proposition 1** ( [11, Lemma 3.1])**.** *If $\hat{\rho}(S) < 1$, then the CSS is stable.*

It can be shown [29, Lemma 1.2] that $\lim_{k\to\infty} \hat{\rho}_k(S) = \inf_{k\geq 1} \hat{\rho}_k(S)$. Therefore, for any $k > 0$, $\hat{\rho}_k(S)$ is an upper bound to $\hat{\rho}(S)$. This fact, together with Proposition 1, gives us a way to check whether a given CSS $S$ is stable:

1. Pick a finite $k > 0$, and compute $\hat{\rho}_k(S)$;
2. If $\hat{\rho}_k(S) < 1$, then $\hat{\rho}(S) < 1$ and $S$ is stable;
3. Otherwise, pick a larger $k$ and try again.

If the CSS is unstable, then the above procedure will never terminate.

A way to prove that a CSS is unstable is to find a switching signal that causes the system to be unstable. For example, by finding a cycle $c \in \mathbf{G}_k^\circ$ with $\rho(A_c) \geq 1$, where $\rho(A_c)$ denotes the spectral radius of the matrix $A_c$ induced by the cycle. In other words, finding a matrix product that, when repeated forever, causes the system to be unstable.

Unstable cycles can be found by brute force or branch-and-bound variants [19, 23, 27]. Naturally, these methods look first for unstable cycles with a small length. However, finding longer cycles becomes prohibitively high (see Section 5).

For example, a simple (and naive) procedure to find a cycle is to pick a finite $k$, enumerate all cycles $c \in \mathbf{G}_k^\circ$, check whether $\rho(A_c) \geq 1$, and stop when one such cycle is found.

The method introduced in [32] works well for finding long cycles. To certify the stability of a given CSS, it solves a semidefinite program to compute polynomial Lyapunov functions of degree $2d$. If the program is infeasible, it uses the dual certificate of infeasibity to generate an infinite switching

signal of guaranteed growth rate. Subwords of this signal can be used to find unstable cycles. As cycles are found along an infinite switching signal, finding *long* unstable cycles is not particularly more difficult. Moreover, if no unstable cycle can be found, one can retry with polynomial Lyapunov functions of degree $2(d+1)$. There is an increased guarantee on the growth rate of the infinite switching signal as the degree increases.

The following definition formalizes the spectral radius of cycle induced matrix products.

**Definition 4** ( [11, Definition 1.2])**.** *The* generalized spectral radius *of a CSS $S$ is defined as:*

$$\rho(S) = \limsup_{k\to\infty} \rho_k(S) \text{ where } \rho_k(S) = \sup_{c\in\mathbf{G}_k^\circ} \rho(A_c)^{\frac{1}{k}} \tag{3}$$

It follows [29, Proposition 1.6] that, for finite $k > 0$,

$$\rho_k(S) \leq \rho(S) \leq \hat{\rho}(S) \leq \hat{\rho}_k(S).$$

Moreover, since $\mathcal{A}$ is bounded, it is shown in [11, Theorem A] that $\rho(S) = \hat{\rho}(S)$.

**Remark 1.** *The above discussion about proving that a CSS is unstable focused on finding finite cycles, as opposed to infinite paths. In fact, there is no guarantee that if a CSS satisfies $\hat{\rho}(S) \geq 1$ (i.e., is unstable), then a cycle $c$ with finite length exists, with $\rho(A_c) \geq 1$ (see [29, Section 2.4] and [5, Theorem 2]). However, the systems we experimented with, either satisfy $\hat{\rho}(S) > 1$, or $\hat{\rho}(S) < 1$. For these, the following result was used.*

**Proposition 2** ( [29, Theorem 2.3])**.** *If $\hat{\rho}(S) > 1$, then there exists a cycle $c \in \mathbf{G}_k^\circ$ of length $k$ that satisfies $\rho(A_c) \geq 1$.*

Our goal is to optimally modify a given CSS, by forbidding unstable switching signal cycles from the language it generates. The problem of finding such cycles is outside the scope of our work (see [33] for the algorithm we used, and references thereof for algorithms with the same goal). As such, we introduce the following definition, which represents any algorithm available for this purpose.

**Definition 5** (Oracle)**.** *Given $\epsilon > 0$, we define a stability oracle $\mathcal{O}_\epsilon : S \to \{\texttt{Stable}\} \cup \bigcup_{k=1}^{\infty} \mathbf{G}_k^\circ$, where $S$ is a CSS. The oracle $\mathcal{O}_\epsilon$ returns either* `Stable` *certifying that $\hat{\rho}(S) < 1$ or a cycle $c \in \mathbf{G}_k^\circ$ such that $\rho(A_c)^{1/k} > 1 - \epsilon$.*

We emphasize that the oracle has a (slightly) imperfect behaviour: in case $1 - \epsilon < \hat{\rho}(S) < 1$, one cannot guarantee what the outcome of the oracle will be. This imperfection is intentional (see Remark 1), as it models the state of the art [38]. Proposition 2 ensures that if $\hat{\rho}(S) > 1 - \epsilon$, there exists a $k$ and a cycle $c \in \mathbf{G}_k^\circ$ such that $\rho(A_c)^{1/k} > 1 - \epsilon$.

We now proceed to define the set of possible different switching signals that are admissible.

**Definition 6** (Admissible Regular Language)**.** *We say that $\mathcal{L} = \mathbf{G}^*$ is the language* recognized *by the automaton $\mathbf{G}$. A language is* regular *if it is recognized by a finite automaton. A language $\mathcal{L}$ recognized by an automaton $\mathbf{G}$ is* admissible *for $\mathcal{A}$ if the constrained switched system $S = \langle \mathcal{A}, \mathbf{G} \rangle$ satisfies $\hat{\rho}(S) < 1$.*

Let $\mathcal{L}_0$ denote the language recognized by the automaton $\mathbf{G}_0$ of a given $S = \langle \mathcal{A}, \mathbf{G}_0 \rangle$. Informally, our goal is to find the "largest" regular language $\mathcal{L}^\star \subseteq \mathcal{L}_0$ that is admissible. For this optimization problem to be well defined we need to find a metric for the objective. This metric should be in accordance to the fact that given $\mathcal{L} \subseteq \mathcal{L}'$, the objective should favor $\mathcal{L}'$. A widely used notion to describe the size of a regular language is that of Entropy.

**Definition 7** (Entropy [35, Definition 4.1.1])**.** *Given a regular language $\mathcal{L}$ recognized by an automaton $\mathbf{G}$, we define the* entropy *as*

$$h(\mathcal{L}) = \lim_{k \to \infty} \frac{1}{k} \log_2 |\mathbf{G}_k|.$$

*In the above, $|\mathbf{G}_k|$ represents the number of words of length $k$ accepted by the automaton $\mathbf{G}$.*

We denote the entropy of the language $\mathbf{G}^*$ recognized by an automaton $\mathbf{G}$ as $h^*(\mathbf{G})$.

If $\mathcal{L} \subseteq \mathcal{L}'$, then $\mathbf{G}_k \subseteq \mathbf{G}'_k$ for any $k$, and so $h(\mathcal{L}) \leq h(\mathcal{L}')$. Our problem can now be formulated.

**Problem 1.** *Given a CSS $\langle \mathcal{A}, \mathbf{G}_0 \rangle$, find the language $\mathcal{L}^\star$ solution of the following optimization problem:*

$$\begin{aligned} \mathcal{L}^\star = \sup_{\mathcal{L} \text{ regular}} \; & h(\mathcal{L}) \text{ s.t.} \\ & \mathcal{L} \subseteq \mathcal{L}_0, \\ & \mathcal{L} \text{ is admissible for } \mathcal{A}. \end{aligned} \tag{4}$$

*where $\mathcal{L}_0$ is the language recognized by $\mathbf{G}_0$.*

**Remark 2.** *In Problem 1, we restrict our attention to regular languages. While there are examples that highlight the benefit of using non-regular languages (see Example 1), in practice, one needs an* efficient *way of generating accepted switching signals. For instance, during a co-simulation, at any step, the simulators need to compute as quickly as possible the set of policies that can be taken (see [16, Section 4.4] for how this can be done). Automata allow the decision procedure to be fast, with little memory. In addition, as hinted in Example 2, regular languages may be constructed to approximate an admissible language with entropy arbitrarily close to the entropy of the optimal solution, even if that optimal solution is a non-regular language.*

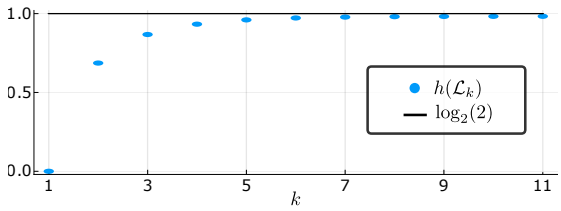
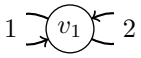



Figure 3: Evolution of $h(\mathcal{L}_k)$ of Example 2 in terms of $k$.

**Example 1.** *Consider $\mathcal{A} = \{A_1, A_2\}$, with $A_1 = 2$ and $A_2 = \frac{1}{2}$, and $\mathbf{G} = (V, E)$, where $V = \{v_1\}$ and $E = \{(v_1, v_1, 1), (v_1, v_1, 2)\}$. That is, $\mathbf{G}$ has the form*




*The optimal solution $\mathcal{L}^\star$ of (relaxed) Problem 1 should include every word that has more $1$s than $2$s. As shown in [47, Example 1.73], no automaton can be built that accepts this language.*

**Example 2.** *Consider $\mathcal{A} = \{A_1, A_2\}$, with $A_1 = 1$ and $A_2 = \frac{1}{2}$. A language is admissible if it does not contain the infinite repetition of the symbol 1. Let $\mathcal{L}_k$ be language of all words that do not contain $k$ consecutive 1's. Figure 3 suggests that that $h(\mathcal{L}_k)$ tends to $\log_2(2)$ when $k$ tends to infinity. The quantity $\log_2(2)$ denotes the entropy of the optimal solution.*

# 3 Lift-and-Constrain Stabilization

## 3.1 Constraining for more stability

Algorithm 1 details an iterative procedure that stabilizes a given CSS $S = \langle \mathcal{A}, \mathbf{G} \rangle$, using the oracle in Definition 5. At each iteration, if the oracle returns a cycle $c = \sigma_k \dots \sigma_k$, then $c$ is eliminated from $\mathbf{G}$. The removal of a cycle can be accomplished by removing an edge of $\mathbf{G}$, thus potentially decreasing $\hat{\rho}(S)$. After removing the cycle $c$, any infinite sequence in $\mathbf{G}^*$ for which $c$ is a subsequence will be eliminated too. This is illustrated in Example 3. The algorithm can produce an empty CSS, which does not imply that the original CSS is impossible to stabilize. An empty CSS is trivially stable.

**Example 3.** *Consider the automaton in Figure 4, and suppose the oracle has returned the cycle $234$. This cycle is highlighted in red, in the figure. Any of the edges in red can be removed to forbid the unstable sequence. If edge $v_1 \xrightarrow{2} v_2$ is removed, the infinite sequences accepted by the resulting automaton end with either an infinite sequence of $2$'s, or an infinite sequence*

*of* $3$*'s. If edge* $v_2 \xrightarrow{3} v_3$ *is removed instead, the resulting automaton accepts infinite sequences comprised of repeating subsequences which include* $2$*, or* $3$*, or* $12$*.*

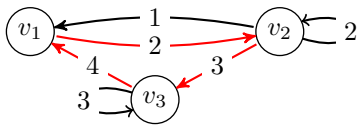



Figure 4: Automaton of Example 3.

As Example 3 shows, the choice of different edges to be removed has a different impact in the entropy of the resulting automaton. Informally, removing the edge $v_2 \xrightarrow{3} v_3$ seems to be the best choice because the resulting automata allows for *more* sequences. This is corroborated by computing the entropy of the resulting automaton alternatives. See Appendix A for how to compute the entropy in this example.

**Data:** A CSS $S = \langle \mathcal{A}, \mathbf{G} \rangle$.
**Result:** A stable CSS $S = \langle \mathcal{A}, \mathbf{G} \rangle$.
**while** $\mathcal{O}_\epsilon(S) \neq$ `Stable` **do**
1. Find $e \in \arg\max\{\, h^*(\mathbf{G} - e) \mid e \in E,\ e$ is an edge of the cycle $\mathcal{O}_\epsilon(S)\,\}$;
2. Set $\mathbf{G} := \mathbf{G} - e$;

**end**

**Algorithm 1:** Stabilization algorithm for a constrained switched system. $h^*(\mathbf{G})$ denotes the entropy of the language recognized by $\mathbf{G}$. The difference $\mathbf{G} - e$ denotes the automaton obtained by removing the edge $e$ from $\mathbf{G}$.

The following result demonstrates that Algorithm 1 always terminates.

**Theorem 1.** *Given a CSS* $S = \langle \mathcal{A}, \mathbf{G} \rangle$ *and an oracle satisfying Definition 5, Algorithm 1 terminates in finite time and the resulting CSS is stable.*

*Proof.* At each iteration of the algorithm, the number of edges of the automaton $\mathbf{G} = (V, E)$ decreases by one. Since at the beginning of the algorithm $|E|$ is finite, the algorithm must terminate after a finite number of iterations. The condition for termination of Algorithm 1 implies that the resulting system is stable. □

**Remark 3.** *In Theorem 1, the assumption that the oracle in Definition 5 always terminates is crucial, as the problem solved by the oracle is undecidable in general (recall Remark 1).*

## 3.2 Lifting for less conservativeness

Algorithm 1 takes a constrained switched system $S = \langle \mathcal{A}, \mathbf{G} \rangle$, and outputs a constrained switched system $S' = \langle \mathcal{A}', \mathbf{G}' \rangle$ that is stable, while attempting to maximize the entropy of the language recognized by $\mathbf{G}'$. If we let $\mathcal{L}'$ denote this language, then, relating this to Problem 1, $\mathcal{L}$ is admissible and regular, and thereby a potential solution. However, it may not be the optimal solution. Similarly, if the algorithm returns an empty CSS, this does not mean that the original CSS is impossible to stabilize . To maximize the entropy of the stabilized CSS's, we propose to take an *$M$-Path-Dependent lift* of the automaton representing the input language $\mathcal{L}_0$.

**Definition 8** ( [40, Definition 3])**.** *Given an automaton $\mathbf{G}$, we define the* lifted *automaton $\mathbf{G}^{[k]}$ of degree $k$ as follows. For each path $v_0, \sigma_0, v_1, \sigma_1, \ldots, \sigma_k, v_{k+1}$ with length $k+1$ of $\mathbf{G}$, $\mathbf{G}^{[k]}$ has a node $u^- = v_0\sigma_0 v_1 \sigma_1 \ldots \sigma_{k-1} v_k$, a node $u^+ = v_1\sigma_1 v_2 \sigma_2 \ldots \sigma_k v_{k+1}$ and an edge $(u^-, u^+, \sigma_k)$.*

Figure 5 shows the second degree ($k = 2$) lift of the automaton in Figure 4.

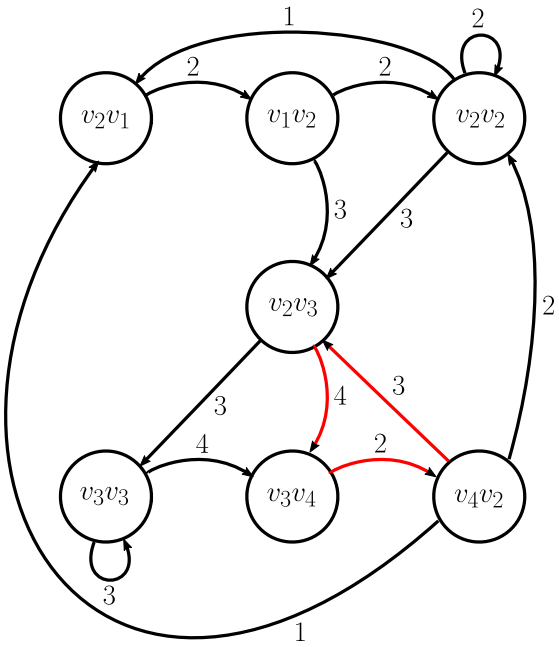



Figure 5: Second degree lifted automaton of Example 3.

The lifted automaton represents the same language, as shown by Proposition 3, but, as suggested by Theorem 2 and illustrated by Example 2 (, lifting the automaton before applying Algorithm 1 allows one to obtain an admissible language with higher entropy.

**Proposition 3.** *Let $\mathcal{L}$ be the language recognized by $\mathbf{G}$ and $\mathcal{L}^{[k]}$ the language recognized by $\mathbf{G}^{[k]}$, where $\mathbf{G}^{[k]}$ is the lift of degree $k$ of $\mathbf{G}$. Then $\mathcal{L} = \mathcal{L}^{[k]}$.*

*Proof.* Consider a sequence $\sigma_0 \dots \sigma_{i-1}$.

If $\sigma_0 \dots \sigma_{i-1} \in \mathbf{G}_i$, there exists nodes $v_0, v_1, \dots, v_i$ of $\mathbf{G}$ such that

$$v_0, \sigma_0, v_1, \sigma_1, \dots, \sigma_{i-1}, v_i$$

is a path of $\mathbf{G}$. As no node has zero ingoing degree, there exists a path of length $k$ that ends in node $v_0$, denoted as $v_{-k}, \sigma_{-k}, \dots, \sigma_{-1}, v_0$ in $\mathbf{G}$. By Definition 8, for any $j = 0, \dots, i$, $u_j = v_{j-k}\sigma_{j-k} \dots \sigma_{j-1}v_j$ is a node of $\mathbf{G}^{[k]}$ and for any $j = 0, \dots, i-1$, there is an edge $(u_j, u_{j+1}, \sigma_j)$ in $\mathbf{G}^{[k]}$. Therefore $\sigma_0 \dots \sigma_{i-1} \in \mathbf{G}_i^{[k]}$.

If $\sigma_0 \dots \sigma_{i-1} \in \mathbf{G}_i^{[k]}$, there exists nodes $u_0, u_1, \dots, u_i$ of $\mathbf{G}^{[k]}$ such that $u_0, \sigma_0, u_1, \sigma_1, \dots, \sigma_{i-1}, u_i$ is a path of $\mathbf{G}^{[k]}$. Let $v_{-k}, \dots, v_i$ be the nodes of $\mathbf{G}$ and $\sigma_{-k}, \dots, \sigma_{-1}$ be the symbols such that for any $j = 0, \dots, i$, $u_j = v_{j-k}\sigma_{j-k} \dots \sigma_{j-1}v_j$. By Definition 8, $v_0, \sigma_0, v_1, \sigma_1, \dots, \sigma_{i-1}, v_i$ is a path of $\mathbf{G}$ hence $\sigma_0 \dots \sigma_{i-1} \in \mathbf{G}_i$.

□

**Theorem 2.** *Consider Algorithm 1 with input $\mathcal{A}, \mathbf{G}_0^{[k]}$ (resp. $\mathcal{A}, \mathbf{G}_0^{[k+1]}$) where $\mathbf{G}_0^{[k]}$ (resp. $\mathbf{G}_0^{[k+1]}$) is the lift of degree $k$ (resp. $k+1$) of a given automaton $\mathbf{G}$. If $\mathcal{O}_\epsilon(\mathcal{A}, \mathbf{G}_0^{[k]})$ and $\mathcal{O}_\epsilon(\mathcal{A}, \mathbf{G}_0^{[k+1]})$ are cycles corresponding to the same word, then $h^*(\mathbf{G}_1^{[k]}) \leq h^*(\mathbf{G}_1^{[k+1]})$.*

*Proof.* Let $e$ be the edge such that $\mathbf{G}_1^{[k]} = \mathbf{G}_0^{[k]} - e$, that is, the edge removed by the algorithm for $\mathbf{G}_0^{[k]}$. Let $\sigma_1\sigma_2 \dots \sigma_k\sigma_{k+1}\sigma_{k+2}$ be a sub-word of the repetition of the cycle $c$ and $v_1, v_2, \dots, v_{k+3}$ be such that

$$e = (v_1\sigma_1v_2\sigma_2 \dots \sigma_kv_{k+1}, v_2\sigma_2 \dots \sigma_kv_{k+1}\sigma_{k+1}v_{k+2}, \sigma_{k+1})$$

and $(v_{k+2}, v_{k+3}, \sigma_{k+2})$ is an edge of $\mathbf{G}$. Let $\mathbf{G}_0^{[k+1]'}$ be the graph obtained by removing the node $v_1\sigma_1v_2\sigma_2 \dots \sigma_{k+1}v_{k+2}$ in $\mathbf{G}_0^{[k+1]}$. The two automata $\mathbf{G}_1^{[k]}$ and $\mathbf{G}_0^{[k+1]'}$ recognize the same language. Let $\mathbf{G}_0^{[k+1]''}$ be the graph obtained by removing the edge $e' = (v_1\sigma_1v_2\sigma_2 \dots \sigma_{k+1}v_{k+2}, v_2\sigma_2v_3\sigma_3 \dots \sigma_{k+2}v_{k+3}, \sigma_{k+2})$ in $\mathbf{G}_0^{[k+1]}$. The language recognized by $\mathbf{G}_0^{[k+1]'}$ is a subset of the language recognized by $\mathbf{G}_0^{[k+1]''}$.

Moreover, as $e'$ is an edge of the cycle $\mathcal{O}_\epsilon(\mathcal{A}, \mathbf{G}_0^{[k+1]})$, $h^*(\mathbf{G}_0^{[k+1]''}) \leq h^*(\mathbf{G}_1^{[k+1]})$. Therefore

$$h^*(\mathbf{G}^{[k]}{}_0) = h^*(\mathbf{G}_0^{[k+1]'}) \leq h^*(\mathbf{G}_0^{[k+1]''}) \leq h^*(\mathbf{G}_1^{[k+1]}).$$

□

In Section 5 we show results corroborating Theorem 2.

# 4 Implementation Details & Optimality

## 4.1 Implementation

The implementation of the stabilization of a CSS is summarized as follows:

1. find all unstable cycles of length from 1 to 3 by iterating over all cycles of these lengths using brute force enumeration;
2. collect the ones that have a spectral radius greater than or equal to one.
3. since several cycles can be disallowed by removing a single edge, select the edge that disallows the largest number of unstable cycles, and use the entropy of the resulting graph to break ties;
4. repeat steps 1–3 until all allowed cycles have a spectral radious below 1;
5. then use the method of [32] to determine whether the resulting CSS is stable or whether there is an unstable cycle.
6. if there is an unstable cycle, select the edge that maximizes the entropy of the resulting system (steps 1–2 of Algorithm 1);
7. repeat steps 5–6 until the resulting system is stable.

It is easy to see that this implementation is a realization of Algorithm 1. Steps 1–4 are an optimization since they execute relatively quickly, and make the execution of [32] take less time.

In Step 6, instead of computing the entropy, we compute the spectral radius of the adjacency matrix of the resulting system. This is equivalent to maximizing the entropy (see Appendix A).

## 4.2 Optimality

The solution attained by Algorithm 1 is not necessarily the optimal solution. For once, applying different lift degrees will yield different optimal solutions. Second, Algorithm 1 removes an edge before finding the next unstable cycle, which means that it misses the chance of optimizing which edge to remove, when more cycles are available (recall Steps 1–4 of the above implementation).

Unfortunately, we found no way of guessing which lift degree yields the optimal solution. However, with small enough constrained switched systems, it is possible to find the optimal solution, for a given lift degree $k$.

To find the optimal solution, suppose that, for a CSS with a lift degree $k$, we know what the unstable cycles are. Now we can iterate over all possible procedures to disallow these cycles in the CSS (each procedure is a sequence of edges to be removed), and compute the entropy of the resulting CSS. The optimal solution is the one that has the maximal entropy.

In order to collect all the unstable cycles, we can perform the following procedure:

1. given a CSS with a lift of degree $k$, apply Algorithm 1 to find an admissible language, and record all the cycles that were removed throughout the procedure;
2. iterate over all possible ways of disallowing the cycles on the original CSS with a lift of degree $k$, and apply the one that results in a language with maximal entropy;
3. the resulting language is not necessarily admissible, because the best procedure is not necessarily that same as the one picked by Algorithm 1 in Step 1, so apply Algorithm 1 to identify and disallow the remaining cycles, adding these to the set of unstable cycles.
4. now repeat Steps 2–3, collecting more and more unstable cycles, until the resulting language is admissible.

The resulting set of unstable cycles represent all possible unstable cycles, and the admissible language found is the optimal solution.

This procedure is applied in Section 5.2.

# 5 Application

We first sketch the application of our algorithm to simulation, and then detail the treatment to co-simulation in Section 5.2. See [29, 46] for other applications of switched systems, where our technique could be applied as well.

## 5.1 Simulation

Consider the problem of approximating the solution $x(t)$ of the system,

$$\dot{x}(t) = \bar{A}x(t), \text{ with } x(0) = x_0, \tag{5}$$

using an adaptive simulation algorithm. These methods are useful in situations where, e.g., the error tolerance, or runtime performance, can vary as a function of $\bar{x}(t)$ and $t$ [10, 13, 45]. In practice, multi-step variable order methods [10, Section 4] are the most commonly used, but for illustrative purposes, we show a single step method. The same analysis can be done for an variable order multi-step method by converting it to a representation in the form of System (1).

**Example 4.** *The approximation $\tilde{x}(t)$ of the solution to System (5), computed by a simulation algorithm that uses different step sizes, and different numerical methods, can be modelled as an unconstrained switched system*

*(System (1)), with*

$$\begin{aligned}\mathcal{A} &= \left\{ \tilde{A}_{fe,h}, \tilde{A}_{md,h}, \tilde{A}_{rg,h} \middle| h \in \{0.001, 0.002\} \right\} \\ \tilde{A}_{fe,h} &\triangleq \mathbf{I} + \bar{A}h \\ \tilde{A}_{md,h} &\triangleq \mathbf{I} + \bar{A}h + (\bar{A}h)^2/2 \\ \tilde{A}_{rg,h} &\triangleq \mathbf{I} + \bar{A}h + (\bar{A}h)^2/12 + (\bar{A}h)^3/6 + (\bar{A}h)^4/24.\end{aligned}$$

*In the above set $\mathcal{A}$, the matrices correspond respectively to the Forward Euler method, the Midpoint method and the Runge-Kutta method.*

In Example 4, if one assumes that System (5) is stable, that is,

$$\lim_{t\to\infty} \|x(t)\| = 0 \text{ for any } x(0),$$

then we need to ensure that the error made by discrete approximation $\tilde{x}(t)$ is dissipated. For this purpose, we introduce the *error switched system*, whose state variable is $e_t = \tilde{x}(t) - x(t)$, and which can be written as

$$\begin{aligned} e_{t+h} &= A_{\sigma(t)} e_t + L_{\sigma(t)}, \text{ with} \\ L_{\sigma(t)} &\triangleq \big(A_{\sigma(t)} - \exp(\bar{A}h)\big)x(t), \end{aligned} \tag{6}$$

where $L_{\sigma(t)}$ is the local error corresponding to $A_{\sigma(t)} \in \mathcal{A}$, and $\mathcal{A}$ is defined in Example 4. Neglecting $L_{\sigma(t)}$ yields a switched system, the stability of which determines the dissipation of error.

**Remark 4.** *To derive Equation* (6)*,*

$$\begin{aligned} e_{t+1} &= \tilde{x}_{t+1} - x(t+h) \\ &= A_{\sigma(t)}\tilde{x}_t - \exp(\bar{A}h)x(t) \\ &= A_{\sigma(t)}(e_t + x(t)) - \exp(\bar{A}h)x(t) \\ &= A_{\sigma(t)}e_t + (A_{\sigma(t)} - \exp(\bar{A}h))x(t) \end{aligned} \tag{7}$$

*where $\exp(h\bar{A}) = \sum_{k=0}^{\infty} \frac{h^k}{k!}\bar{A}^k$ is the matrix exponential.*

As an example, the error of the adaptive simulation algorithm introduced in Example 4 may not be stable for a switching signal $111\ldots$, but may be stable for $2121\ldots$ As a result, the adaptive simulation method may opt for a policy sequence that requires fewer model evaluations (compared to a non-adaptative algorithm), whilst preserving the stability. Another way of stating this is to observe that the spectral radius of $\tilde{A}_1$ is larger than 1, that is, $\rho(\tilde{A}_1) > 1$. This *does not* imply that the product of any switching signal of the form $2121\ldots$ causes the system to be unstable.

To illustrate this fact, Figure 6 shows the domain of numerical stability for multiple "hybrid" solvers applied to the linear system in Equation (5). Each

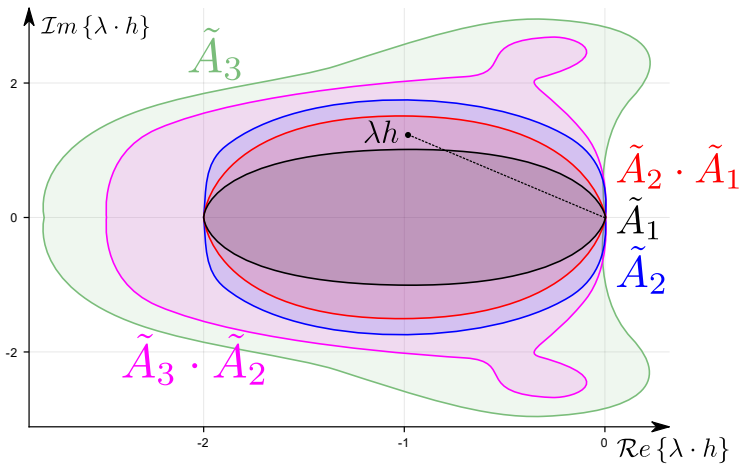



Figure 6: Domain of numerical stability for some hybrid methods in $\mathcal{A}^2$, defined in Example 4.

solver is constructed by combining two matrices in $\mathcal{A}$, where $\mathcal{A}$ is defined in Example 4. As the figure shows, it is possible that, for a given $h$ and Eigenvalue $\lambda$ of $\bar{A}$, $\lambda h$ is located outside the stability domain of $\tilde{A}_1$ (shaded area in black), but inside the stability region of the hybrid method $\tilde{A}_2\tilde{A}_1$ (shaded in red). In that case, forbidding any switching signal of the form $11\ldots$, may ensure that the system introduced in Example 4 is stable.

See [10, Section 2.4] for an example of how to construct the stability domain.

## 5.2 Co-simulation

We apply our algorithm to certify an adaptive orchestration algorithm for the co-simulation of a controlled inverted pendulum. We will consider two simulators.

In the context of co-simulation, time is discretized into a countable set $T = \{t_0, t_1, t_2, \ldots\} \subset \mathbb{R}$, where $t_{i+1} = t_i + H_i$ is the time at step $i$ and $H_i$ is the communication step size at step $i$, with $i = 0, 1, \ldots$. From time $t_i \to t_{i+1}$, the simulator $S_j$, with $j = 1, 2$, is a mapping,

$$\begin{aligned} \tilde{x}_j(t_{i+1}) &= F_j(t_i, \tilde{x}_j(t_i), u_j(t_i)) \\ y_j(t_i) &= G_j(t_i, \tilde{x}_j(t_i), u_j(t_i)) \end{aligned} \tag{8}$$

with state vector $\tilde{x}_j$, input vector $u_j$ and output vector $y_j$.

Simulators exchange outputs only at times $t \in T$. We assume without loss

of generality that the two simulators are coupled in a feedback loop, that is,

$$u_1 = y_2 \text{ and } u_2 = y_1. \tag{9}$$

In the interval $t \in [t_i, t_{i+1}]$, each simulator $S_j$ approximates the solution to a linear ODE,

$$\begin{aligned} \dot{x}_j &= A_j x_j + B_j u_j \\ y_j &= C_j x_j + D_j u_j \end{aligned} \tag{10}$$

where $A_j, B_j, C_j, D_j$ are matrices, and the initial state $x_j(t_i)$ is computed in the most recent co-simulation step. To avoid algebraic loops and keep the exposition short, we assume that either $D_1$ or $D_2$ is the zero matrix. Let $D_2 = \mathbf{0}$.

Since the simulators only exchange outputs at times $t_i, t_{i+1} \in T$, the input $u_j$ has to be extrapolated in the interval $[t_i, t_{i+1})$. In the simplest co-simulation strategy[1], this extrapolation is often implemented as a zero-order hold: $\tilde{u}_j(t) = u_j(t_i)$, for $t \in [t_i, t_{i+1})$. Then, Equation (10) can be re-written to represent the unforced system being integrated by each simulator:

$$\begin{bmatrix} \dot{x}_j \\ \dot{\tilde{u}}_j \end{bmatrix} = \begin{bmatrix} A_j & B_j \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_j \\ \tilde{u}_j \end{bmatrix} \tag{11}$$

We can represent the multiple internal integration steps of System (11), performed by the simulator $S_j$ in the interval $t \in [t_i, t_{i+1}]$, as

$$\begin{bmatrix} \tilde{x}_j(t_{i+1}) \\ \tilde{u}_j(t_{i+1}) \end{bmatrix} = \tilde{A}_j^{k_j} \begin{bmatrix} \tilde{x}_j(t_i) \\ \tilde{u}_j \end{bmatrix} \tag{12}$$

where, e.g., $\tilde{A}_j = \mathbf{I} + h_j \begin{bmatrix} A_j & B_j \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ for the Forward Euler method, $k_j = (t_{i+1} - t_i)/h_j$ is the number of internal steps, and $0 < h_j \leq H_i$ is the internal fixed step size that divides $H_i$. Note that this equation implements the mapping in Equation (8).

At the beginning of the co-simulation step $i$, $u_1(t_i) = y_2(t_i)$ and $u_2(t_i) = y_1(t_i)$, as in Equation (9). This, together with Equation (10), gives,

$$\begin{aligned} u_1(t_i) &= C_2 \tilde{x}_2(t_i) \\ u_2(t_i) &= C_1 \tilde{x}_1(t_i) + D_1 C_2 \tilde{x}_2(t_i). \end{aligned} \tag{13}$$

Equations (11), (12), and (13) can be used to represent each co-simulation step by a linear mapping

$$\begin{bmatrix} \tilde{x}_1(t_{i+1}) \\ \tilde{x}_2(t_{i+1}) \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{A}_1^{k_1} & \mathbf{0} \\ \mathbf{0} & \tilde{A}_2^{k_2} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & C_2 \\ \mathbf{0} & \mathbf{I} \\ C_1 & D_1 C_2 \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} \tilde{x}_1(t_i) \\ \tilde{x}_2(t_i) \end{bmatrix}$$

[1]The derivation presented can be applied to more sophisticated input extrapolation techniques, see [6, Equation (9)].

whose state transition matrix $\tilde{A}$ depends on the following items [18]:

- coupling approach — we used the Jacobi coupling, but Gauss-Seidel, Strong coupling, and others, can be used [21];
- simulator input approximation — we used constant extrapolation, but other techniques can be applied [6];
- internal solver method — we showed the Forward Euler as example, but other methods are available;
- internal simulator step size $h_j$ — this value affects matrix $\tilde{A}_j$ and the value $k_j$;
- communication step size $H_i$ — affects $k_j$.

As evidenced in [6, 9, 15, 42], each configuration of these items has different stability properties. Moreover, as hinted in [16], an adaptive algorithm that changes the configuration at runtime, based on varying tolerance and performance requirements, is beneficial *as long as it does not make the co-simulation unstable* (recall Example 4). Therefore, with the present work we generate a stabilized CSS, that encodes the set of all possible sequences of configurations that make the co-simulation stable, which can then be consulted during the co-simulation, with little computational cost [16].

In order to represent an adaptive co-simulation, let $\mathcal{A}$ be a set of co-simulation state transition matrices, each representing a particular configuration of the above items. An adaptive orchestration algorithm will, at the beginning of each co-simulation step $i$, inspect the state variables, and/or local error estimators [2], and decide which configuration should be used to proceed to step $i+1$. If we assume that any configuration can be chosen at each co-simulation step, then we can represent the adaptive orchestration algorithm as a switched system. On the other hand, if the choice of a configuration for step $i$ depends (in addition to the run-time information) on the configuration chosen for steps $i-1, i-2, \ldots$, then the adaptive orchestration algorithm can be modelled as a constrained switched system.

Consider the co-simulation of an inverted pendulum that is kept at the equilibrium point using a state feedback controller. Simulator $S_1$ represents the controller, and simulator $S_2$ represents the pendulum.

Around the equilibrium point, the pendulum can be approximated as a system of the form of Equation (10), with

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -(I+ml^2)(b/p) & (m^2gl^2)/p & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -(mlb)/p & mgl(M+m)/p & 0 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0 & (I+ml^2)/p & 0 & ml/p \end{bmatrix}^\top$$

$$C_2 = \mathbf{I} \qquad D_2 = \mathbf{0}$$

and parameters $M = 0.5, m = 0.2, b = 0.1, I = 0.006, g = 9.8, l = 0.3$.

The controller is a linear quadratic regulator, which, put in the form of Equation (10), is

$$A_1 = \mathbf{0} \qquad B_1 = \mathbf{0}$$
$$C_1 = \mathbf{0} \qquad D_1 = \begin{bmatrix} 1.0000 & 1.6567 & -18.6854 & -3.4594 \end{bmatrix}.$$

Assume we can use the Forward Euler and Midpoint methods, with internal fixed step sizes in the set $\{0.01, 0.02, 0.1, 0.2\}$. Furthermore, the co-simulation step size can be $H = 0.1$ or $H = 0.2$. Note that the internal step sizes must always divide the communication step size $H$, and that the numerical method and step size used in the controller simulator have no impact in the co-simulation stability, because it has no internal dynamics. Then, applying Equations (11), (12), and (13), we get a switched system over 8 matrices.

The matrices $A_2$ (corresponding to $H = 0.2$, $h_1 = 0.2$, Forward Euler), $A_3$ (corresponding to $H = 0.2$, $h_1 = 0.02$, Midpoint) and $A_4$ (corresponding to $H = 0.2$, $h_1 = 0.02$, Forward Euler) have a spectral radius larger than one. This means that the switching signals $222...$, $333...$ and $444...$ should be forbidden.

Applying Algorithm 1 directly to the unconstrained switched system (which corresponds to a lift of degree 0), leads to removal of the edges with labels $2, 3$ and $4$. This completely disallows the use of the matrices $A_2, A_3$ and $A_4$. The resulting language turns out to be admissible, its entropy is $\log_2(5)$.

Applying Algorithm 1 to a lift of degree 1, we get a constrained switched system with the automaton shown in Figure 7, where the edges in red were removed by the algorithm. We can see that the matrices $A_2$, $A_3$ and $A_4$ are now allowed by the algorithm (only the cyclic application of each one of these matrices is still disallowed). This solution is less conservative than the one with degree 0. Its entropy is $log_2(7.26)$. One allowed cycle is $32645$, where the symbols 5 and 6 seem to play the role of stabilizing the cycle.

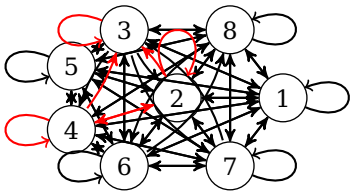



Figure 7: Solution with entropy $\log_2(7.2568898)$.

We applied Algorithm 1 to the lifts of degree 0, 1 and 2. At each application of the algorithm, a stable constrained switched system was produced, with an entropy that increased with the degree of the lift. These results, summarized in Table 1, corroborate Theorem 2.

Table 1: Entropy achieved per lift degree.

| $k$ | Entropy [bit] | CPU time [s] |
|---|---|---|
| 0 | $\log_2(5)$ | 0.13 |
| 1 | $\log_2(7.2568898)$ | 1.8 |
| 2 | $\log_2(7.7083039)$ | 280 |

This application to co-simulation illustrates an important advantage of the method presented in [32]: it is capable of finding large unstable cycles. This method does not find the unstable cycles by iterating through the cycles of some length $K$ but instead extracts them from an infinite switching signal, hence it is not harder for the method to find large unstable cycles. For the lift of degree 2 for example, it found the unstable cycle $542245332$ of length 9, and in a subsequent iteration found the cycle $22453354224533542245335422453254$ of length 33. A brute force method would have to enumerate all $8^{33}$ cycles to achieve the stabilization of the adaptive solver.

Regarding the optimality of the solution found for the lift with degree 1, we have applied the procedure detailed in Section 4.2 to confirm that $\log_2(7.2568898)$ is indeed the maximal entropy for that degree.

# 6 Related Work

There has been a huge effort to understand how to ensure that a (discrete time) switched system is stable. We refer the interested reader to [1, 29, 34, 38, 49] for introductions and surveys on this subject. To the best of our knowledge, the problem we introduce here, i.e. finding the largest set of switching signals that guarantees the stability of the system, has never been studied. In the broader field of stabilization of switched systems, we can highlight the works in [24, 26, 31, 39, 41, 51–54]. The key difference with our work is the goal: we are not satisfied with a single stable switching signal; we want to provide the maximum flexibility to the stabilized CSS, which can make use of this flexibility to choose the most appropriate switching signal. The works in [26, 31, 39, 41, 51, 52] are focused on continuous time systems, and [51, 52, 54] aim at deriving state feedback laws (in addition to switching signals) that make the system stable.

The approach followed in [54] assumes that each mode of the system is stable. In our case, the goal is the same but we tolerate unstable modes.

The approach in [24] is interesting because it allows the combination of stable and unstable modes, in order to ensure stability. However, no algorithm is provided to find these combinations.

The aim of [31] is different as it describes the search for one particular

stable trajectory while we maximize the size of a language of stable switching signals.

[48] describes the stability analysis for continuous switched systems with parametric uncertainties.

[53] focuses on proving that a system is stabilizable, rather than making the system stable. It deals with forced discrete time switched systems, and the stabilization procedure finds a control policy (switching signal, and input) that stabilizes the system. This is in contrast to our goal, which is to find all policies that make the system stable, and maximize this set. In the context of co-simulation, the reader can find stability analysis of traditional orchestration algorithms in [6, 15, 42, 43].

# 7 Conclusion

We introduce a new problem in the context of constrained switched systems: 1) to restrict the switching possibilities of the system, so as to ensure its stability, and 2) to leave as many switching policies as possible (provided that the system becomes stable).

The motivation for leaving as many switching policies as possible lies in the fact that, in adaptive co-simulation, the orchestration algorithm will make the best possible choice as a function of information obtained during the simulation. We restrict the switching possibilities to be representable by an automaton because of their great efficiency.

The problem is interesting in that it transforms a control problem into the problem of building an optimal language, that is, optimizing the construction of an automaton. By combining classical control concepts for switched systems (like the CJSR) , with classical automata-theoretic concepts (like the entropy of shifts), one can design algorithms to solve this problem. Our algorithm takes the form of a hierarchy of sufficient conditions, where increasingly better solutions are found by lifting the automaton (see Figure 3 and Table 1). Essentially, this allows one to control the optimality of the solution, at the cost of processing power and memory.

This work is aimed to be a proof of concept, and we leave many research questions open. We plan to investigate the conservativeness of restricting ourselves to regular languages (see Example 1). Second, we want to understand how our method can be optimized for the particularizes of co-simulation, and apply it to nonlinear systems. Finally, we plan to modify Algorithm 1 so that stronger theoretical results can be proven.

## Acknowledgments

This research was partially supported by a PhD fellowship grant from the Agency for Innovation by Science and Technology in Flanders (IWT, dossier 151067), the Belgian Interuniversity Attraction Poles, the ARC grant 13/18-054 from Communauté française de Belgique, a F.R.S.-FNRS Research Fellowship, and Flanders Make vzw, the strategic research centre for the manufacturing industry.

## References

[1] Amir Ali Ahmadi, Raphaël Jungers, Pablo A Parrilo, and Mardavij Roozbehani. Analysis of the joint spectral radius via lyapunov functions on path-complete graphs. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 13–22. ACM, 2011.

[2] Martin Arnold, Christoph Clauß, and Tom Schierz. Error Analysis and Error Estimates for Co-simulation in FMI for Model Exchange and Co-Simulation v2.0. In Sebastian Schöps, Andreas Bartel, Michael Günther, W. E. Jan ter Maten, and C. Peter Müller, editors, *Progress in Differential-Algebraic Equations*, pages 107–125. Springer Berlin Heidelberg.

[3] Torsten Blochwitz, Martin Otter, Martin Arnold, C. Bausch, Christoph Clauss, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, M. Monteiro, T. Neidhold, Dietmar Neumerkel, Hans Olsson, J.-V. Peetz, and S. Wolf. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In *8th International Modelica Conference*, pages 105–114. Linköping University Electronic Press; Linköpings universitet.

[4] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In *9th International Modelica Conference*, pages 173–184. Linköping University Electronic Press.

[5] Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. 41(2):135–140.

[6] Martin Busch. Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. 96(9):1061–1081.

[7] Martin Busch. Zur effizienten Kopplung von Simulationsprogrammen.

[8] Martin Busch and Bernhard Schweizer. Coupled simulation of multi-body and finite element systems: An efficient and robust semi-implicit coupling approach. 82(6):723–741.

[9] Martin Busch and Bernhard Schweizer. Numerical stability and accuracy of different co-simulation techniques: Analytical investigations based on a 2-DOF test model. In *1st Joint International Conference on Multibody System Dynamics*, pages 25–27.

[10] François Edouard Cellier and Ernesto Kofman. *Continuous System Simulation*. Springer Science & Business Media.

[11] Xiongping Dai. A Gel'fand-type spectral radius formula and stability of linear constrained switching systems. 436(5):1099–1113.

[12] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—A survey. 25(3):335–348.

[13] C. W. Geart and D. S. Watanabe. Stability and Convergence of Variable Order Multistep Methods. 11(5):1044–1058.

[14] Cláudio Gomes. Foundations for Continuous Time Hierarchical Co-simulation. In *ACM Student Research Competition (ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems)*, page to appear. ACM New York, NY, USA.

[15] Cláudio Gomes, Paschalis Karalis, Eva M. Navarro-López, and Hans Vangheluwe. Approximated Stability Analysis of Bi-modal Hybrid Co-simulation Scenarios. In *1st Workshop on Formal Co-Simulation of Cyber-Physical Systems*, pages 345–360. Springer, Cham.

[16] Cláudio Gomes, Benoît Legat, Raphaël M. Jungers, and Hans Vangheluwe. Stable Adaptive Co-simulation : A Switched Systems Approach. In *IUTAM Symposium on Co-Simulation and Solver Coupling*, page to appear.

[17] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: A Survey. 51(3):Article 49.

[18] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art.

[19] Gustaf Gripenberg. Computing the joint spectral radius. *Linear Algebra and its Applications*, 234:43–60, 1996.

[20] Bei Gu. Co-simulation of algebraically coupled dynamic subsystems.

[21] Bei Gu and H. Harry Asada. Co-simulation of algebraically coupled dynamic subsystems. In *American Control Conference*, volume 3, pages 2273–2278. IEEE.

[22] Bei Gu and H. Harry Asada. Co-Simulation of Algebraically Coupled Dynamic Subsystems Without Disclosure of Proprietary Subsystem Models. 126(1):1.

[23] Nicola Guglielmi and Marino Zennaro. An algorithm for finding extremal polytope norms of matrix families. *Linear Algebra and its Applications*, 428(10):2265–2282, 2008.

[24] Guisheng Zhai, Bo Hu, K. Yasuda, and A.N. Michel. Qualitative analysis of discrete-time switched systems. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, pages 1880–1885 vol.3. IEEE.

[25] Esteban Hernandez-Vargas, Patrizio Colaneri, Richard Middleton, and Franco Blanchini. Discrete-time control for switched positive systems with application to mitigating viral escape. *International Journal of Robust and Nonlinear Control*, 21(10):1093–1111, 2011.

[26] L. Hetel, J. Daafouz, and C. Iung. Stabilization of Arbitrary Switched Linear Systems With Unknown Time-Varying Delays. 51(10):1668–1674.

[27] Raphaël M Jungers, Antonio Cicone, and Nicola Guglielmi. Lifted polytope methods for computing the joint spectral radius. *SIAM Journal on Matrix Analysis and Applications*, 35(2):391–410, 2014.

[28] Raphaël M Jungers, Vladimir Protasov, and Vincent D Blondel. Efficient algorithms for deciding the type of growth of products of integer matrices. *Linear Algebra and its Applications*, 428(10):2296–2311, 2008.

[29] Raphaël Jungers. *The Joint Spectral Radius: Theory and Applications*, volume 385. Springer Science & Business Media.

[30] R. Kübler and W. Schiehlen. Two Methods of Simulator Coupling. 6(2):93–113.

[31] Atreyee Kundu and Debasish Chatterjee. Stabilizing switching signals: A transition from point-wise to asymptotic conditions. *Systems & Control Letters*, 106:16–23, 2017.

[32] B. Legat, P. A. Parrilo, and R. M. Jungers. Certifying unstability of Switched Systems using Sum of Squares Programming.

[33] Benoît Legat, Raphaël M. Jungers, and Pablo A. Parrilo. Generating Unstable Trajectories for Switched Systems via Dual Sum-Of-Squares

Techniques. In *19th International Conference on Hybrid Systems: Computation and Control*, pages 51–60. ACM Press.

[34] Hai Lin and Panos J. Antsaklis. Stability and Stabilizability of Switched Linear Systems: A Survey of Recent Results. 54(2):308–322.

[35] Douglas Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge university press.

[36] Peter Linz. *An Introduction to Formal Languages and Automata*. Jones & Bartlett Publishers.

[37] Petter Nilsson and Necmiye Ozay. On a class of maximal invariance inducing control strategies for large collections of switched systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 187–196. ACM, 2017.

[38] Pablo A Parrilo and Ali Jadbabaie. Approximation of the joint spectral radius of a set of matrices using sum of squares. In *HSCC*, pages 444–458. Springer, 2007.

[39] S. Pettersson. Synthesis of switched linear systems. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 5, pages 5283–5288. IEEE.

[40] Matthew Philippe, Ray Essick, Geir E. Dullerud, and Raphaël M. Jungers. Stability of discrete-time switching systems with constrained switching sequences. 72:242–250.

[41] Pavithra Prabhakar and Miriam García Soto. Formal synthesis of stabilizing controllers for switched systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 111–120. ACM, 2017.

[42] Bernhard Schweizer, Pu Li, and Daixing Lu. Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches. 10(5):051007.

[43] Bernhard Schweizer, Pu Li, Daixing Lu, and Tobias Meyer. Stabilized implicit co-simulation methods: Solver coupling based on constitutive laws. 85(11):1559–1594.

[44] Bernhard Schweizer, Daixing Lu, and Pu Li. Co-simulation method for solver coupling with algebraic constraints incorporating relaxation techniques. 36(1):1–36.

[45] Lawrence F Shampine and Mark W Reichelt. The matlab ode suite. 18(1):1–22.

[46] Robert Shorten, Fabian Wirth, Oliver Mason, Kai Wulff, and Christopher King. Stability criteria for switched and hybrid systems. *SIAM review*, 49(4):545–592, 2007.

[47] Michael Sipser. *Introduction to the Theory of Computation*.

[48] Christoffer Sloth and Rafael Wisniewski. Robust stability of switched systems. In *53rd IEEE Conference on Decision and Control*, pages 4685–4690. IEEE.

[49] Zhendong Sun and S.S. Ge. Analysis and synthesis of switched linear control systems. 41(2):181–195.

[50] Douglas Brent West. *Introduction to Graph Theory*, volume 2. Prentice hall Upper Saddle River.

[51] X. Xu and P.J. Antsaklis. Optimal Control of Switched Systems Based on Parameterization of the Switching Instants. 49(1):2–16.

[52] Xuping Xu and Panos J. Antsaklis. Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. 75(16-17):1406–1426.

[53] Wei Zhang, Alessandro Abate, Jianghai Hu, and Michael P. Vitus. Exponential stabilization of discrete-time switched linear systems. 45(11):2526–2536.

[54] Xudong Zhao, Lixian Zhang, Peng Shi, and Ming Liu. Stability and Stabilization of Switched Linear Systems With Mode-Dependent Average Dwell Time. 57(7):1809–1815.

# A Computation of the Entropy

## A.1 Spectral Radius of Adjacency Matrix

Consider a given CSS $S = \langle \mathcal{A}, \mathbf{G} \rangle$, and let $B$ be the adjacency matrix of $\mathbf{G}$.

The matrix element $b_{ij}$ of $B^k$ gives the number of different paths of length $k$ from node $i$ to node $j$ [50]. Hence, $\left\|B^k\right\|$ gives a measure of the size of the matrix comprised by the number of different paths from each node to each other node (see, e.g., [33, Remark 2]), and $\left\|B^k\right\|^{\frac{1}{k}}$ gives the growth rate of this quantity. Taking the limit $k \to \infty$, we have the spectral radius of the adjacency matrix:

$$\rho(B) = \lim_{k\to\infty} \left\|B^k\right\|^{1/k}.$$

**Example 5.** *Recall Example 3, let $B_1$ denote the adjacency matrix of the automata in Figure 4 without the edge $v_1 \xrightarrow{2} v_2$, and let $B_2$ denote the adjacency matrix of the same automata, without the edge $v_2 \xrightarrow{3} v_3$. Then $\rho(B_1) = 1 < \rho(B_2) \approx 1.6180$.*

## A.2 Edge Shift

The logarithm of the spectral radius of the adjacency matrix of an *irreducible* automaton gives the entropy of its *edge shift* [35, Theorem 4.3.1]. An automaton is *irreducible* if for every pair of nodes $u, v$, there exists a path from $u$ to $v$ accepted by the automaton. In other words, the graph consists of a single strong component.

**Definition 9** ( [35, Definition 2.2.5])**.** *The* edge shift *of an automaton $\mathbf{G} = (V, E)$ is the language recognized by the automaton $\mathbf{G}' = (E, E')$ with the transitions $((u, v, \sigma), (v, w, \sigma'), (v, w, \sigma')) \in E'$ for each $(u, v, \sigma), (v, w, \sigma') \in E$.*

An edge shift of automaton Figure 8a is illustrated in Figure 8b.

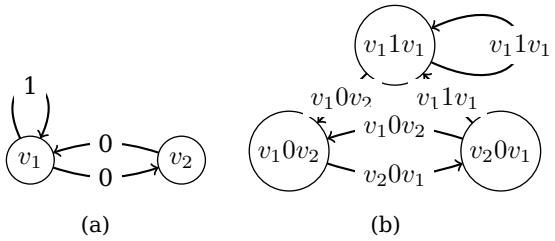



Figure 8: An automaton (a) and its edge shift (b).

It turns out that the entropy of the edge shift is equal to the entropy of the language recognized by the automaton if the automaton is *right-resolving* [35, Proposition 4.1.13].

**Definition 10** ( [35, Definition 3.3.1])**.** *An automaton $\mathbf{G}$ is* right-resolving *if for every vertex $v$, the outgoing edges have different symbols.*

Every regular language is recognized by a right-resolving automaton. Moreover, there are automated ways to obtain such an automaton from a starting representation of a language with an automaton that is not right-resolving [35, Section 3.3].

Since the automata considered here are right resolving and irreducible, the entropy is computed by computing the spectral radius of the adjacency matrix of the CSS.