CoSyMA: A Tool for Controller Synthesis Using Multi-scale Abstractions

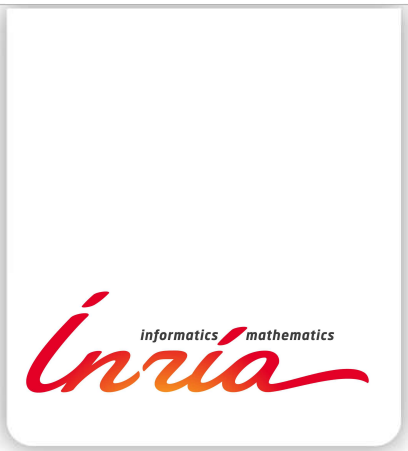
Sebti Mouelhi, Antoine Girard, Gregor Gössler

CoSyMA: A Tool for Controller Synthesis Using Multi-scale Abstractions

Sebti Mouelhi , Antoine Girard, Gregor Gössler

RESEARCH REPORT
N° 8108
Octobre 2012
Project-Teams POP ART

ISSN 0249-6399 ISRN INRIA/RR--8108--FR+ENG



CoSyMA: A Tool for Controller Synthesis Using Multi-scale Abstractions

Sebti Mouelhi [*], Antoine Girard[†], Gregor Gössler [*]

Project-Teams POP ART

Research Report n° 8108 — Octobre 2012 — 23 pages

Abstract: In this report we introduce CoSyMA, a tool for automatic controller synthesis for incrementally stable switched systems based on multi-scale discrete abstractions. The tool accepts a description of a switched system represented by a set of differential equations and the sampling parameters used to define an approximation of the state-space on which discrete abstractions are computed. The tool generates a controller — if it exists — for the system that enforces a given safety or time-bounded reachability specification. We illustrate by examples the synthesized controllers and the significant performance gains during their computation.

Key-words: Switched systems, Multi-scale abstractions, Controller synthesis, Symbolic Algorithms

[*] INRIA Grenoble
[†] LJK, University of Joseph Fourier

RESEARCH CENTRE GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

CoSyMA: un outil pour la synthèse de contrôleurs à base d'abstractions multi-échelles

Résumé : Ce rapport décrit l'outil CoSyMA pour la synthèse automatique de contrôleurs à base d'abstractions multi-échelles pour systèmes commutés incrémentalement stables.

Mots-clés : Systèmes commutés, Abstraction multi-échelles, Synthèse de contrôleurs, Algorithmes symboliques

1 Introduction

Controller synthesis for hybrid systems using discrete abstractions has become an established approach (see [11] and the references therein). In [6], it has been demonstrated that the (sampled) continuous behavior of incrementally stable [1, 9] switched systems are *approximately bisimilar* to some discrete abstractions. The states of these abstractions are elements of lattices that approximate the continuous state-space. Time and space sampling parameters are chosen to achieve a desired precision; the smaller the time sampling parameter, the finer the lattice used for approximating the state-space, and consequently, the larger the number of states in the abstraction. The approach detailed in [4, 3] based on *multi-scale* discrete abstractions can be used to cope with this problem. These abstractions are defined over a set of embedded lattices. The finer lattices are only explored when the specification cannot be met at the coarsest level.

In this paper, we present CoSyMA (*COntroller SYnthesis using Multi-scale Abstractions*), a tool implementing symbolic approaches based on multi-scale abstractions to synthesize controllers for incrementally stable switched systems. CoSyMA accepts as input a switched system defined by differential equations indexed by a set of modes, time and space sampling parameters used to set an approximation of the continuous state-space, and a safety or a time-bounded reachability specification. If it exists, it computes a controller satisfying the specification. The tool is implemented using OCaml [8] and it is available for download at `multiscale-dcs.gforge.inria.fr`.

2 Theoretical background

2.1 Incrementally stable switched systems

Definition 1. *A* switched system *is a quadruple* $\Sigma = \langle \mathbb{R}^n, P, \mathcal{P}, F \rangle$ *where* $\mathbb{R}^n$ *is the state-space;* $P = \{1, \ldots, m\}$ *is a finite set of modes;* $\mathcal{P}$ *is the set of piecewise constant functions from* $\mathbb{R}^+$ *to* P*, continuous from the right and with a finite number of discontinuities on every bounded interval of* $\mathbb{R}^+$*;* $F = \{f_1, \ldots, f_m\}$ *is a collection of smooth vector fields indexed by* P*. For all mode* $p \in P$*,* $f_p : \mathbb{R}^n \to \mathbb{R}^n$ *is a locally Lipschitz continuous map.*

A *switching signal* of Σ is a function $\mathbf{p} \in \mathcal{P}$. A piecewise $\mathcal{C}^1$ function $\mathbf{x} : \mathbb{R}^+ \to \mathbb{R}^n$ is said to be a *trajectory* of Σ if it is continuous and there exists a switching signal $\mathbf{p} \in \mathcal{P}$ such that, at each $t \in \mathbb{R}^+$ where the function $\mathbf{p}$ is continuous, $\mathbf{x}$ is continuously differentiable and satisfies $\dot{\mathbf{x}}(t) = f_{\mathbf{p}(t)}(\mathbf{x}(t))$. We denote by $\mathbf{x}(t, x, p)$ the point reached at time t, starting from the state x and applying the constant switching signal $\mathbf{p}(s) = p$, for all $s \in [0, t]$. A switched system is δ-GUAS (i.e. globally uniformly asymptotically incrementally stable [1, 6]) if all the trajectories associated with the same switching signal converge asymptotically to the same reference trajectory independently of their initial states.

2.2 Approximate bisimulation

In this section we provide a brief introduction of the notion of approximate bisimulation that relates a switched system to the specific discrete abstraction we construct. Let us consider a class of transition systems of the form $T = \langle Q, L, r, O, H, I \rangle$ consisting of a set of states Q; a set of labels L; a transition relation $r \subseteq Q \times L \times Q$; an output set O;

an output function $H : Q \to O$; a set of initial states $I \subseteq Q$. T is said to be *metric* if the output set O is equipped with a metric d, *discrete* if Q and L are finite or countable sets. For $q \in Q$ and $l \in L$ let $succs_l(q) = \{q' \in Q \mid (q,l,q') \in r\}$. An action $l \in L$ belongs to the set of *enabled actions* at the state q, denoted $Enab(q)$, if $succs_l(q) \neq \emptyset$. The transition system is said to be *deterministic* if for all $q \in Q$ and $l \in Enab(q)$, $succs_l(q)$ has only one element denoted by $succ_l(q)$. A *trajectory* of the transition system is a finite or infinite sequence of transitions $\sigma = q_0 l_0 q_1 l_1 q_2 l_2 \ldots$, it is *initialized* if $q_0 \in I$. A state $q \in Q$ is *reachable* if there exists an initialized trajectory reaching q. We denote by $\Phi(T)$ the set of all the trajectories of T.

Transition systems can described the dynamics of switched systems. Given a switched system $\Sigma = \langle \mathbb{R}^n, P, \mathcal{P}, F \rangle$, let $T(\Sigma) = \langle Q, L, r, O, H, I \rangle$ be the transition system where the set of states is $Q = \mathbb{R}^n$; the set of labels is $L = P \times \mathbb{R}^+$; the transition relation is given by $(x,(p,\tau),x') \in r$ iff $\mathbf{x}(\tau,x,p) = x'$, i.e. the switched system Σ goes from state x to state x' by applying the constant mode p for a duration τ; the set of outputs is $O = \mathbb{R}^n$; the observation map H is the identity map over $\mathbb{R}^n$; the set of initial states is $I = \mathbb{R}^n$. $T(\Sigma)$ is deterministic and metric when the set of outputs $O = \mathbb{R}^n$ is equipped with the metric $d(x,x') = \|x - x'\|$. The relation between the discrete abstractions and $T(\Sigma)$ can be defined by an approximate bisimulation [5].

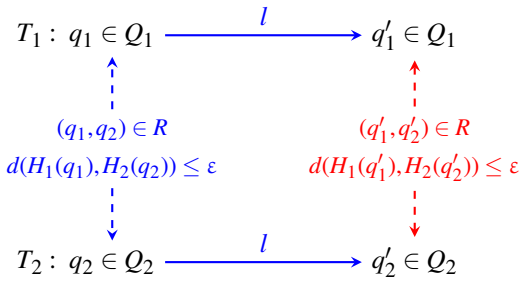


Figure 1: The principal of ε-approximate bisimulation relation

Definition 2. *Let $T_i = \langle Q_i, L_i, r_i, O_i, H_i, I_i \rangle$, for $i \in \{1,2\}$, be metric transition systems where $L_1 = L_2$ and $O_1 = O_2$ equipped with the metric d, and a precision $\varepsilon \geq 0$. A relation $R \subseteq Q_1 \times Q_2$ is said to be an* ε-approximate bisimulation relation *between T_1 and T_2 if for all $(q_1,q_2) \in R$:*

- $d(H_1(q_1),H_2(q_2)) \leq \varepsilon$;
- $\forall (q_1,l,q_1') \in r_1$, $\exists (q_2,l,q_2') \in r_2$, *such that* $(q_1',q_2') \in R$;
- $\forall (q_2,l,q_2') \in r_2$, $\exists (q_1,l,q_1') \in r_1$, *such that* $(q_1',q_2') \in R$.

T_1 and T_2 are said to be approximately bisimilar with precision ε, denoted $T_1 \sim_\varepsilon T_2$, if for all $q_1 \in I_1$, there exists $q_2 \in I_2$, such that $(q_1,q_2) \in R$, and for all $q_2 \in I_2$, there exists $q_1 \in I_1$, such that $(q_1,q_2) \in R$.

The diagram shown in Figure 1 illustrates the principle of the ε-approximate bisimulation relation.

2.3 Multi-scale abstractions

In applications where the switching has to be fast, uniform approximately bisimilar abstractions, as defined in [6], approximate the state-space using fine lattices that results

in a a huge number of abstract states. In practice, fast switching is generally necessary only on a restricted part of the state space. For instance, for safety specifications, fast switching is needed only when the system gets closer to unsafe regions. In order to enable fast switching while using abstractions with a reasonable number of states, we consider discrete abstractions enabling transitions of different durations. For transitions of long duration, it is sufficient to consider abstract states on the coarse lattice. The finer ones are reached by the shorter transitions only when the specification cannot be met at the coarsest level.

Let us consider a switched system Σ whose switching is determined by a time-triggered controller with time-periods in the finite set $\Theta^N_\tau = \{2^{-s}\tau \mid s = 0,\ldots,N\}$ that consists of dyadic fractions of a time sampling parameter $\tau > 0$ up to some scale parameter $N \in \mathbb{N}$. The dynamics of a switched system Σ is then described by the transition system $T^N_\tau(\Sigma) = \langle Q_1, P \times \Theta^N_\tau, r_1, O, H_1, I_1\rangle$ where $Q_1 = O = I_1 = \mathbb{R}^n$, H_1 is the identity map over $\mathbb{R}^n$, and $(x,(p,2^{-s}\tau),x') \in r_1$ iff $\mathbf{x}(2^{-s}\tau,x,p) = x'$.

The discrete abstraction of $T^N_\tau(\Sigma)$ is defined on an approximation of $Q_1 = \mathbb{R}^n$ by a set of embedded lattices $[\mathbb{R}^n]^s_\eta$ defined by

$$[\mathbb{R}^n]^s_\eta = \left\{ q \in \mathbb{R}^n \;\middle|\; q[i] = k_i \frac{2^{-s+1}\eta}{\sqrt{n}},\ k_i \in \mathbb{Z},\ i = 1,...,n \right\}$$

where $s = 0,\ldots,N$, $q[i]$ is the i-th coordinate of q and $\eta > 0$ is a state space discretization parameter. By simple geometrical considerations, we can check that for all $x \in \mathbb{R}^n$ and $s = 0,\ldots,N$, there exists $q \in [\mathbb{R}^n]^s_\eta$ such that $\|x - q\| \leq 2^{-s}\eta$. Then, we can define the abstraction of $T^N_\tau(\Sigma)$ as the transition system $T^N_{\tau,\eta}(\Sigma) = \langle Q_2, P \times \Theta^N_\tau, r_2, O, H_2, I_2\rangle$, where the set of states is $Q_2 = [\mathbb{R}^n]^N_\eta$; the set of actions remains $L = P \times \Theta^N_\tau$; r_2 is defined such that $(q,(p,2^{-s}\tau),q') \in r_2$ iff $q' = \arg\min_{m \in [\mathbb{R}^n]^s_\eta}(\|\mathbf{x}(2^{-s}\tau,q,p) - m\|)$. The approximation principle is illustrated in Figure 2.3. The observation map H_2 is the natural inclusion map from $[\mathbb{R}^n]^N_\eta$ to $\mathbb{R}^n$; the set of initial states is $I_2 = [\mathbb{R}^n]^0_\eta$.

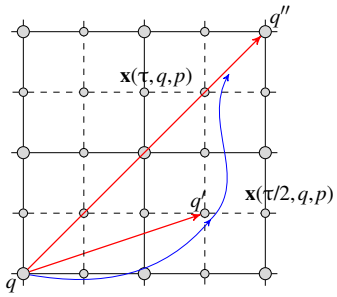


Figure 2: Computation of the discrete abstraction: $q' = succ_2(q,(p,\frac{\tau}{2})) = \arg\min_{m \in [\mathbb{R}^n]^1_\eta}(\|\mathbf{x}(\frac{\tau}{2},q,p) - m\|)$ and $q'' = succ_2(q,(p,\tau)) = \arg\min_{m \in [\mathbb{R}^n]^0_\eta}(\|\mathbf{x}(\tau,q,p) - m\|)$

The resulting abstraction $T^N_{\tau,\eta}(\Sigma)$ is discrete and deterministic, its set of states and its set of actions are respectively countable and finite. For $N = 0$, we recover the "uniform" abstractions introduced in [6]. In [4], it was proved that for a switched system Σ admitting a common δ-GUAS Lyapunov function, $T^N_\tau(\Sigma) \sim_\varepsilon T^N_{\tau,\eta}(\Sigma)$ where the precision ε can be made arbitrarily small by reducing the state sampling parameter η.

2.4 Controller synthesis using multi-scale abstractions

Before presenting the tool details and our experimental results, we explain briefly how we use multi-scale abstractions for synthesizing safety and time-bounded reachability controllers. Let $T = \langle Q, L, r, O, H, I\rangle$ be a deterministic transition system, a *controller* for T is a map $\mathcal{S} : Q \to 2^L$ such that for all $q \in Q$, $\mathcal{S}(q) \subseteq Enab(q)$. The system T controlled by $\mathcal{S}$ is $T/\mathcal{S} = \langle Q, L, r_{\mathcal{S}}, O, H, I\rangle$ where the transition relation is given by $(q,l,q') \in r_{\mathcal{S}}$ iff $(l \in \mathcal{S}(q) \wedge (q,l,q') \in r)$. The *support* of $\mathcal{S}$ is defined by $\mathsf{supp}(\mathcal{S}) = \{q \in Q \mid \mathcal{S}(q) \neq \emptyset\}$.

Safety controller synthesis

Given a safety specification $Q_S \subseteq Q$ (obtained from a subset $O_S \subseteq O$ of safe outputs), a state q of T is *controllable with respect to a safety specification* Q_S if $q \in Q_S$ and there exists an infinite trajectory $\sigma \in \Phi(T)$ starting from q and remaining in Q_S. We denote the set of controllable states of T with respect to the safety specification Q_S by $SCont(T, Q_S)$. A *safety controller* $\mathcal{S}$ for T and Q_S is defined such that $\mathsf{supp}(\mathcal{S}) \subseteq SCont(T, Q_S)$ and for all $q \in \mathsf{supp}(\mathcal{S})$: (1) $q \in Q_S$ (safety) and (2) $\forall l \in \mathcal{S}(q)$, $succ_l(q) \in \mathsf{supp}(\mathcal{S})$ (deadend freedom). The set $SCont(T, Q_S)$ is computable for discrete abstractions. However, the larger the number of states, the more expensive the computation. For that reason, we want to capitalize on multi-scale abstractions to propose an efficient algorithm for safety controller synthesis.

The lazy safety synthesis problem consists in controlling a system so as to keep any trajectory starting from some initial state in I within the safe subset of states Q_S, while applying at each state transitions of the longest possible duration for which safety can be guaranteed. For that purpose we define a priority relation on the set of labels $L = P \times \Theta_\tau^N$ giving priority to transitions of longer duration: for $l, l' \in L$ with $l = (p, \tau)$, $l' = (p', \tau')$, $l \preceq l'$ iff $\tau \leq \tau'$, $l \prec l'$ iff $\tau < \tau'$ and $l \cong l'$ iff $\tau = \tau'$. Given a subset of labels $L' \subseteq L$, we define $\max_{\preceq}(L') = \{l' \in L' \mid \forall l \in L', l \preceq l'\}$.

Definition 3. *A* maximal lazy safety (MLS) controller $\mathcal{S} : Q \to 2^L$ *for* T *and* Q_S *is a safety controller such that* $I \cap SCont(T, Q_S) \subseteq \mathsf{supp}(\mathcal{S})$ *and for all states* $q \in \mathsf{supp}(\mathcal{S})$*, we have: (1) if* $l \in \mathcal{S}(q)$*, then for any* $l \prec l'$*,* $succ_{l'}(q) \notin SCont(T, Q_S)$ *(laziness), and (2) if* $l \in \mathcal{S}(q)$*, then for any* $l \cong l'$*,* $l' \in \mathcal{S}(q)$ *iff* $succ_{l'}(q) \in SCont(T, Q_S)$ *(maximality).*

The MLS controller exists and is unique as proved in [3].

Time-bounded reachability controller synthesis

Given a transition system $T = \langle Q, L, r, O, H, I\rangle$, for all transitions $(q,l,q') \in r$, let $\delta(l)$ be the time needed by T to reach q' from q by action l. For all finite trajectories $\sigma = q_0 l_0 q_1 l_1 \ldots l_{n-1} q_n$ in $\Phi(T)$, we define its *duration* by $\Delta(\sigma) = \delta(l_0) + \delta(l_1) + \cdots + \delta(l_{n-1})$. For instance, for the transition systems $T_\tau^N(\Sigma)$ and $T_{\tau,\eta}^N(\Sigma)$, we have $L = P \times \Theta_\tau^N$ and for all $l = (p, 2^{-s}\tau) \in L$, we have $\delta(l) = 2^{-s}\tau$.

To formally define time-bounded reachability controller, we define $C(T) = \langle Q_c, L, r_c, O, H_c, I_c\rangle$ the *transition system with clock* of T where $Q_c = Q \times \mathbb{R}^+$ is the set of states Q extended by a clock; for all $((q,c), l, (q', c')) \in r_c$, $(q,l,q') \in r$ and $c' = c + \delta(l)$; $H_c((q,c)) = H(q)$ for all $(q,c) \in Q_c$; $I_c = I \times \{0\}$. The set of reachable states of $C\big(T_{\tau,\eta}^N(\Sigma)\big)$ is countable and defined by $Q \times 2^{-N}\tau\mathbb{N}$. Given a maximal time bound $B \in \mathbb{R}^+$, the state (q,c) of $C(T)$ is controllable with respect to a time-bounded reachability specification

(Q_S, Q_T, B), where $Q_T \subseteq Q_S$ if (1) $q \in Q_S$ and there exists a finite trajectory σ of T starting from q, eventually reaching Q_T, and remaining in Q_S until reaching Q_T, such that $c + \Delta(\sigma) \leq B$. The set of these states is denoted by $RCont(C(T), Q_S, Q_T, B)$.

Next we define time-bounded reachability controllers using the notion of safety controllers. We start by defining the notion of *stuttering* ($\circ$) actions. An outgoing transition from a state q labeled by a stuttering action loops on the same state ($succ_\circ(q) = q$ and $\delta(\circ) = 0$).

Let us now define $T_{\circlearrowleft Q'} = \langle Q, L \cup \{\circ\}, r^{\circlearrowleft}, O, H, I\rangle$ for $Q' \subseteq Q$ such that

$$(q,l,q') \in r^{\circlearrowleft} \text{ iff } \begin{cases} q = q' & \text{if } l = \circ \text{ and } q \in Q'; \\ (q,l,q') \in r & \text{if } l \neq \circ \text{ and } q \in Q \setminus Q'. \end{cases}$$

$T_{\circlearrowleft Q'}$ is the transition system derived from T where the only actions enabled from a state in Q' are stuttering.

Since we are not concerned with the evolution of the system after reaching the target Q_T, we will use the transition system $C(T_{\circlearrowleft Q_T}) = \langle Q_c, L, r_c^{\circlearrowleft}, O, H_c, I_c\rangle$ rather than $C(T)$. We easily show that $RCont(C(T_{\circlearrowleft Q_T}), Q_S, Q_T, B) = RCont(C(T), Q_S, Q_T, B) = SCont(C(T_{\circlearrowleft Q_T}), Q_S \times [0, B])$.

A *time-bounded reachability controller* for the transition system $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B) is a safety controller for $C(T_{\circlearrowleft Q_T})$ and $Q_S \times [0, B]$. We define the maximal lazy time-bounded reachability controller based on Definition 3 as follows.

Definition 4. *The* maximal lazy time-bounded reachability (MLBR) controller $\mathcal{R}^m : Q \times \mathbb{R}^+ \to 2^L$ *for* $C(T_{\circlearrowleft Q_T})$ *and* (Q_S, Q_T, B) *is the MLS controller for* $C(T_{\circlearrowleft Q_T})$ *and* $Q_S \times [0, B]$.

It is clear, based on the previous definition, that $\mathcal{R}^m$ is unique and that Q_T is reachable within the specified time bound starting from controllable initial states.

Example 2.1. We shall give a simple example to describe the principle of MLBR controller. Given a time and space sampling parameters τ and η, we consider the transition system $T^1_{\tau,\eta}(\Sigma)$ of a switched system Σ with a unique mode p. The set of labels is defined by $L = P \times \Theta^1_\tau$ where $P = \{p\}$. In Figure 3 (left), we show only a part of the transition system $T^1_{\tau,\eta}(\Sigma)$ where, from initial states $\{1, 5, 7, 9\}$, we take only paths that enter the set of target states Q_T (the set $\{3, 4, 6, 8\}$ is included in Q_T) within a time lesser or equals to $B = 2\tau$. We suppose that all states $\{1, \ldots, 9\}$ are in the set of safe states Q_S. The system controlled by the MLBR controller according to the specification $(Q_S, Q_T, 2\tau)$ is described in Figure 3 (right). The reader can easily check that $\mathcal{R}^m$ is a time-bounded reachability controller for $T^1_{\tau,\eta}(\Sigma)$ and $(Q_S, Q_T, 2\tau)$ such that the maximality and laziness properties are satisfied.

We can use the algorithm synthesizing MLS controllers proposed in [3] to synthesize MLBR controllers. However, their computation is expensive because dealing with problems of n dimensions amounts to handle their equivalents of $n+1$ dimensions by adding a clock. To avoid this constraint, we can settle for a sub-controller $\mathcal{V}$ of $\mathcal{R}^m$ such that all controllable initial states of $\mathcal{R}^m$ are also controllable by $\mathcal{V}$.

Let $\mathcal{R}^m$ be the MLBR controller for $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B). A *sub-controller* $\mathcal{V}$ of $\mathcal{R}^m$ is a time-bounded reachability controller for $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B) such that for all $(q, c) \in \mathsf{supp}(\mathcal{V})$, $\mathcal{V}((q, c)) \subseteq \mathcal{R}^m((q, c))$. The sub-controller $\mathcal{V}$ is *complete* if $\{i \in I \mid (i, 0) \in \mathsf{supp}(\mathcal{R}^m)\} = \{i \in I \mid \exists 0 \leq c \leq B \mid (i, c) \in \mathsf{supp}(\mathcal{V})\}$. We can now define static reachability controllers based on the previous definition.

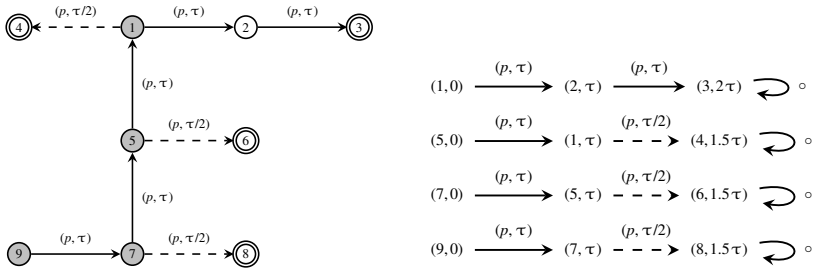


Figure 3: A part T of $T^1_{\tau,\eta}(\Sigma)$ (above); The transition system $C(T_{\circlearrowleft Q_T})/\mathcal{R}^m$ where $\mathcal{R}^m$ is the MLBR controller for T and (Q_S, Q_T, B) (below)

Definition 5. *Consider a complete sub-controller $\mathcal{V}$ of $\mathcal{R}^m$. The* static reachability controller $\mathcal{R}^{ls}_{\mathcal{V}} : Q \to 2^L$ *for $T_{\circlearrowleft Q_T}$ and (Q_S, Q_T, B) obtained from $\mathcal{V}$ is the controller such that for all $(q,c) \in \mathsf{supp}(\mathcal{V})$, $q \in \mathsf{supp}(\mathcal{R}^{ls}_{\mathcal{V}})$ and for all $q \in \mathsf{supp}(\mathcal{R}^{ls}_{\mathcal{V}})$, $\mathcal{R}^{ls}_{\mathcal{V}}(q) = \mathcal{V}\big((q, c_{max}(q))\big)$ where $c_{max}(q) = max\{c' \mid (q,c') \in \mathsf{supp}(\mathcal{V})\}$.*

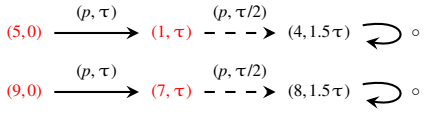


Figure 4: The transition system $C(T_{\circlearrowleft Q_T})/\mathcal{V}$ where $\mathcal{V}$ is a complete sub-controller of the MLBR controller $\mathcal{R}^m$ shown in Figure 3 (right)

Example 2.2. Let us consider the transition system $C(T_{\circlearrowleft Q_T})/\mathcal{R}^m$ of Fig. 3 (right), we can define the controller $\mathcal{V}$ such that $C(T_{\circlearrowleft Q_T})/\mathcal{V}$ is the one shown in Fig. 4. It is clear that $\mathcal{V}$ is a sub-controller for $\mathcal{R}^m$. The sub-controller $\mathcal{V}$ is complete because all the initial states $\{i \in I \mid \exists (i,0) \in \mathsf{supp}(\mathcal{R}^m)\}$ (states $\{1,5,7,9\}$) remain controllable in $C(T_{\circlearrowleft Q_T})/\mathcal{V}$. The states 1 and 7 are controllable with clocks greater than 0.

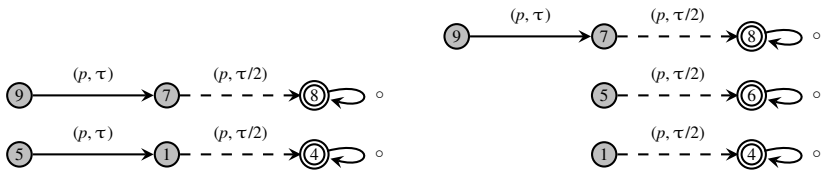


Figure 5: The transition system $T_{\circlearrowleft Q_T}/\mathcal{R}^{ls}_{\mathcal{V}}$ controlled by $\mathcal{R}^{ls}_{\mathcal{V}}$ where $\mathcal{V}$ is the controller of $C(T_{\circlearrowleft Q_T})/\mathcal{V}$ shown in Figure4 (above); the transition system $T_{\circlearrowleft Q_T}/\mathcal{R}^{ls}_{\mathcal{V}}$ where $\mathcal{V} = \mathcal{R}^m$ (below)

Example 2.3. Consider the complete sub-controller $\mathcal{V}$ where $C(T_{\circlearrowleft Q_T})/\mathcal{V}$ is shown in Figure 4. The static reachability controller $\mathcal{R}^{ls}_{\mathcal{V}}$ derived from $\mathcal{V}$ is defined by the transition system shown in Figure 5 (left). If $\mathcal{V} = \mathcal{R}^m$, the correspondent static reachability controller $\mathcal{R}^{ls}_{\mathcal{V}}$ is shown in Figure 5 (right).

Theorem 1. *Any trajectory σ of $T_{\circlearrowleft Q_T}/\mathcal{R}^{ls}_{\mathcal{V}}$ starting from $q \in \mathsf{supp}(\mathcal{R}^{ls}_{\mathcal{V}})$ eventually reaches Q_T and $\Delta(\sigma) \leq B$.*

Proof. For all states $q \in \mathsf{supp}(\mathcal{R}^{ls}_{\mathcal{V}})$, the state (q,c), where $c = max\{d \mid (q,d) \in \mathsf{supp}(\mathcal{V})\}$, is controllable with respect to (Q_S, Q_T, B). This means that any path σ in $C(T_{\circlearrowleft Q_T})/\mathcal{V}$ starting from (q,c) eventually reach Q_T such that $c + \Delta(\sigma) \leq B$. Lets take (q',c') the first state reachable by σ such that $c' < max\{d \mid (q',d) \in \mathsf{supp}(\mathcal{V})\}$, if we replace the rest of σ starting from (q',c') by a path σ' of $C(T_{\circlearrowleft Q_T})/\mathcal{V}$ starting from $(q',c^m_{q'})$ such that $c^m_{q'} = max\{d \mid (q',d) \in \mathsf{supp}(\mathcal{V})\}$, σ eventually reach Q_T and $c' + \Delta(\sigma') < c^m_{q'} + \Delta(\sigma') \leq B$. If we apply the same operations in such way, for each state (q_i,c_i) reachable by σ, $c_i = max\{d \mid (q_i,d) \in \mathsf{supp}(\mathcal{V})\}$, σ eventually reach Q_T and $c + \Delta(\sigma) \leq B$. These paths without clocks are those of $T_{\circlearrowleft Q_T}/\mathcal{R}^{ls}_{\mathcal{V}}$. □

2.4.1 Static reachability controller synthesis

In this section, we present an algorithm computing the static reachability controller for a given complete sub-controller $\mathcal{V}$ for $\mathcal{R}^m$ where $\mathcal{R}^m$ is a MLBR controller for $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B).

Algorithm 1 guards the system at its original dimension and explore states of $T_{\circlearrowleft Q_T}$ rather then states of $C(T_{\circlearrowleft Q_T})$ to reduce the synthesis complexity caused by growing the system by the supplementary clock parameter. For each explored trajectory σ, it stores for each reachable state q by σ three clocks: (i) the current encountered clock *tmp_clock*(q) for q, (ii) the effective maximal encountered *clock*(q) such that the state $(q, clock(q)) \in RCont(C(T), Q_S, Q_T, B)$ and reachable in $C(T_{\circlearrowleft Q_T})/\mathcal{V}$, and (iii) the minimal encountered clock *cutoff*(q) for q such that $(q, cutoff(q))$ is unreachable in $C(T_{\circlearrowleft Q_T})/\mathcal{V}$. A state q is controllable and added to the abstraction reachable states if *clock*(q) is set at least one time. Given a map $f : X \to Y$, we denote by $\mathsf{dom}(f) = X$ the *domain* of f.

The algorithm uses a first-depth traversal starting from initial states $q_0 \in I$ with a temporary clock *tmp_clock*$(q_0) = 0$. For each initialized trajectory $\sigma = q_0 l_0 q_1 l_1 \ldots l_{i-1} q_i \ldots$ of $T_{\circlearrowleft Q_T}$, we use a stack *path* containing tuples $(-, q_0), (l_0, q_1), \ldots, (l_{i-1}, q_i)$ from the bottom to the top. Since q_0 has no predecessor in σ, the tuple corresponding to q_0, added to *path*, is $(-, q_0)$. The stack *path* is grown until reaching a state $q \in Q_T$ or a previously controlled state q such that *tmp_clock*$(q) \leq$ *clock*(q) (lines 31 to 35) if there is no tuple (l, q') such that $q' = q$ was inserted before in *path* to avoid the treatment of cyclic paths looping on q. In other words, a state q is pushed on *path* and explored if *clock*(q) was not set before during the exploration of another initialized trajectory σ' or *tmp_clock*$(q) >$ *clock*(q) because, in the latter case, we can find traces reaching the target from $(q, tmp\_clock(q))$ in a shorter time than those starting from $(q, clock(q))$. For a top element (l, q), transitions of longer duration starting from q are explored first (laziness). We use for that two maps *lexp* and *rexp* associating, respectively, to each state the set of explored labels and remaining labels to explore.

Algorithm 1: STATIC REACHABILITY CONTROLLER SYNTHESIS

Input: Time-bounded specification (Q_S, Q_T, B); $T_{\circlearrowleft Q_T} = \langle Q, L, r_\circ, O, H, I\rangle$ where $I \subseteq [\mathbb{R}^n]^0_\eta \cap Q_S$; priority order $\preceq$.
Output: An LSB reachability controller $\mathcal{R} : Q \to 2^L$.
Data: *initials*: stack($\{-\} \times I$); *path*: stack($L \cup \{-\} \times Q$); *clock*,*tmp_clock*,*cutoff*: $Q \to \mathbb{N}$; *rexp*: $Q \to 2^L$; *lexp*: $Q \to 2^L$; $\to_\mathcal{R}$ is the set of transitions.

```
1  begin
2    Invariants: dom(clock ∪ cutoff) ⊆ dom(tmp_clock); dom(cutoff) ∩ dom(clock) = ∅;
3    Initialization: initials contains tuples in {−} × I;
4    while initials ≠ ∅ do
5      (−, q0) := pop(initials) ;
6      if q0 ∉ dom(clock) then //if q0 was set controllable then it is not explored
7        if q0 ∈ QT then clock(q0) := 0;
8        else push(path,(−, q0)) ; rexp(q0) := max≺(L); lexp(q0) := ∅; tmp_clock(q0) := 0;
9      while path ≠ ∅ do
10       (l, q) := top(path) ;
11       if rexp(q) = ∅ ∧ ∃(q, l′, q′) ∈→R then //if q is fully explored and has successors
12         pop(path) ;
13         if q ∉ dom(clock) then clock(q) := tmp_clock(q);
14         if q ≠ q0 then //while q is not initial
15           (k, p) := top(path) ;
16           call CopyTransition((p, l, q));
17       else
18         if lexp(q) ⊂ L then
19           //update the remaining labels to explore for q
20           if rexp(q) = ∅ then rexp(q) := max≺(L \ lexp(q));
21           choose l′ from rexp(q); q′ := succ_l′(q);
22           clk := tmp_clock(q) + δ(l′);
23           if q′ ∉ dom(cutoff) ∨ cutoff(q′) > clk then
24             if clk ≤ clock(q′) then
25               call CopyTransition((q, l′, q′));
26             if q′ ∉ dom(clock) ∨ clk > clock(q′) then
27               if q′ ∈ QS ∧ clk ≤ B then //if q′ is safe
28                 if q′ ∈ QT then //if q′ within the target
29                   clock(q′) := clk;
30                   call CopyTransition((q, l′, q′));
31                 else if q′ was not pushed before in path then
32                   tmp_clock(q′) := clk;
33                   //initialize or reinitialize labels to explore for q′
34                   rexp(q′) := max≺(L); lexp(q′) := ∅;
35                   push(path,(l′, q′));
36           //update the set of explored labels for q
37           lexp(q) := lexp(q) ∪ {l′}; rexp(q) := rexp(q) \ {l′};
38         else//all labels in L are explored for q without finding successors
39           cutoff(q) := tmp_clock(q);
40           pop(path);
41   return {(q, l) | ∃(q, l, q′) ∈→R}
42 end
```

A tuple (l, q) is popped from the head of *path* if (1) $(tmp\_clock(q), q)$ is uncontrollable (lines 38 to 40), or (2) it is controllable and labels $l = (m, 2^{-s}\tau)$, for all $m \in P$ by incrementing s from 0 to N, are all explored until finding controllable successors for q. In the case (2), q is set to be controllable if not the case (lines 11 to 13) and the function CopyTransition with the input (p, l, q) is called. If $tmp\_clock(p) > clock(p)$, $clock(p)$ is updated to $tmp\_clock(p)$ and all the transitions starting from p previously

added to $\rightarrow_{\mathcal{R}}$ are removed and replaced by (p,l,q). The update of $tmp\_clock(p)$ have to be made (line 4 of the function COPYTRANSITION only one time to replace transitions starting from p with the old value of $clock(p)$ by the first computed transition starting from p with the new value of $tmp\_clock(p)$. The rest of outgoing transitions from p with the new clock are added after the update and lines 2 to 5 are not executed when COPYTRANSITION is called. The function COPYTRANSITION is also called with input (q,l',q') where $q' = succ_{l'}(q)$ if $tmp\_clock(q') \leq clock(q')$ (line 24) or $q' \in Q_T$ (line 28).

Procedure COPYTRANSITION

Input: a transition (p,l,q).

```
1 begin
2   if tmp_clock(p) > clock(p) then
3     Invariant (∃(p,l',q') ∈ →_R);
4     clock(p) := tmp_clock(p) ;
5     →_R := →_R \{(x,l,y) | x = p} ; //remove the outgoing transitions from p
6   →_R := →_R ∪{(p,l,q)}
7 end
```

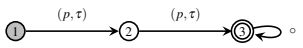
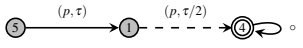
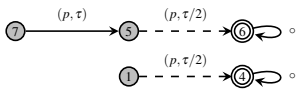
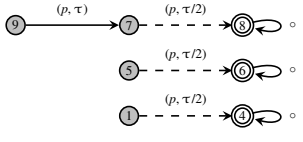
$\gg_1$	*path*	$\rightarrow_{\mathcal{R}}$	*clock*
1	$(3,(p,\tau))$ $(2,(p,\tau))$ $(1,-)$		$1 \mapsto 0$ $2 \mapsto \tau$ $3 \mapsto 2\tau$
5	$(4,(p,\tau/2))$ $(1,(p,\tau))$ $(5,-)$		$1 \mapsto \tau$ $5 \mapsto 0$ $4 \mapsto 1.5\tau$
7	$(6,(p,\tau/2))$ $(5,(p,\tau))$ $(7,-)$		$1 \mapsto \tau$ $5 \mapsto \tau$ $7 \mapsto 0$ $4 \mapsto 1.5\tau$ $6 \mapsto 1.5\tau$
9	$(8,(p,\tau/2))$ $(7,(p,\tau))$ $(9,-)$		$1 \mapsto \tau$ $5 \mapsto 1$ $7 \mapsto \tau$ $9 \mapsto 0$ $4 \mapsto 1.5\tau$ $6 \mapsto 1.5\tau$ $8 \mapsto 1.5\tau$

Table 1: The LSB controller synthesis according to $\gg_1$

Algorithm 1 terminates: for each reachable state q with a temporary clock c, a tuple (l,q') with $q' = q$ is inserted at most once in *path* because if q was set controllable ($clock(q) = c$), it cannot be reinserted in path except in the case where c is grater than $clock(q_i)$ (line 24). In the case where (q,c) is uncontrollable, $cutoff(q)$ is set to c which

prevent adding the state with the same temporary clock a second time in *path* (line 23). Also, eventually, q will be popped from *path* (lines 12 and 40).

From an operational view point, the complete sub-controller $\mathcal{V} \subseteq \mathcal{R}^m$ is selected according to the order of exploration $\gg_I$ of initial states $(q_0, 0) \in I \times \{0\}$. More clearly, let's take the transition system shown in Fig. 3 (above), according to the order $\gg_1 = 1 \triangleright 5 \triangleright 7 \triangleright 9$, the complete sub-controller $\mathcal{V}_{\gg_1}$ is the MLBR controller $\mathcal{R}^m$ shown in Figure 3 (below) (cf. Table. 1). However, if we consider the order $\gg_2 = 9 \triangleright 7 \triangleright 5 \triangleright 1$, the complete sub-controller $\mathcal{V}_{\gg_2}$ is shown in Figure 4 (cf. Table 2). Algorithm 1 iterates the stack *initials* containing tuples $(q_0, -)$ where $q_0 \in I$ according to the order $\gg_I$ from top to bottom.

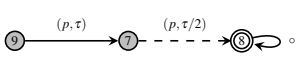
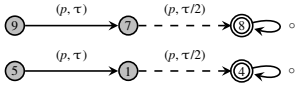
$\gg_2$	*path*	$\rightarrow_{\mathcal{R}}$	*clock*
9	$(8,(p,\tau/2))$ $(7,(p,\tau))$ $(9,-)$		$9 \mapsto 0$ $7 \mapsto \tau$ $8 \mapsto 1.5\tau$
7	Nothing is done because *tmp_clock*(7) $\leq$ *clock*(7), i.e. $0 \leq \tau$		
5	$(4,(p,\tau/2))$ $(1,(p,\tau))$ $(5,-)$		$9 \mapsto 0$ $7 \mapsto \tau$ $8 \mapsto 1.5\tau$ $1 \mapsto \tau$ $5 \mapsto 0$ $4 \mapsto 1.5\tau$
1	Nothing is done because *tmp_clock*(1) $\leq$ *clock*(1), i.e. $0 \leq \tau$		

Table 2: The LSB controller synthesis according to $\gg_2$

3 Tool details

In this section, we present the internal architecture of CoSyMA, its description language, and our implementation choices. CoSyMA accepts a configuration (`.conf`) file describing the system and the synthesis parameters. It contains in the order: the description of a switched system Σ in terms of differential equations $\dot{\mathbf{x}}(t) = f_{\mathbf{p}(t)}(\mathbf{x}(t))$; time τ and space η sampling parameters, and the scale N used to compute the finer lattice $[\mathbb{R}]^N_\eta$ that approximate the continuous state-space; a safety specification Q_S or a time-bounded reachability specification (Q_S, Q_T, B); the plot parameters. The grammar is detailed in Appendix A).

The tool architecture, shown in Figure 6, represents the different steps by which the tool synthesizes the controllers of the described system according to the given safety or a time-bounded reachability specification. After the parsing of the configuration file, the tool represents the vector fields f_p for each switching mode p by an OCaml function of type `float -> float array -> float array`. It computes the successor of a state q under a label $l = (p, 2^{-s}\tau)$ by solving $\dot{\mathbf{x}}(t) = f_p(\mathbf{x}(t))$ for $t \in [0, \delta(l)]$ and $\mathbf{x}(0) = q$ using the common fourth-order Runge-Kutta method. The tool synthesizes the controllers based on $T^N_{\tau,\eta}$ approximately bisimilar to T^N_τ (cf. Section 2.3). For safety specifications Q_S, the tool synthesizes the MLS controller based on the algorithm presented in [3] which computes the abstraction $T^N_{\tau,\eta}$ on the fly.

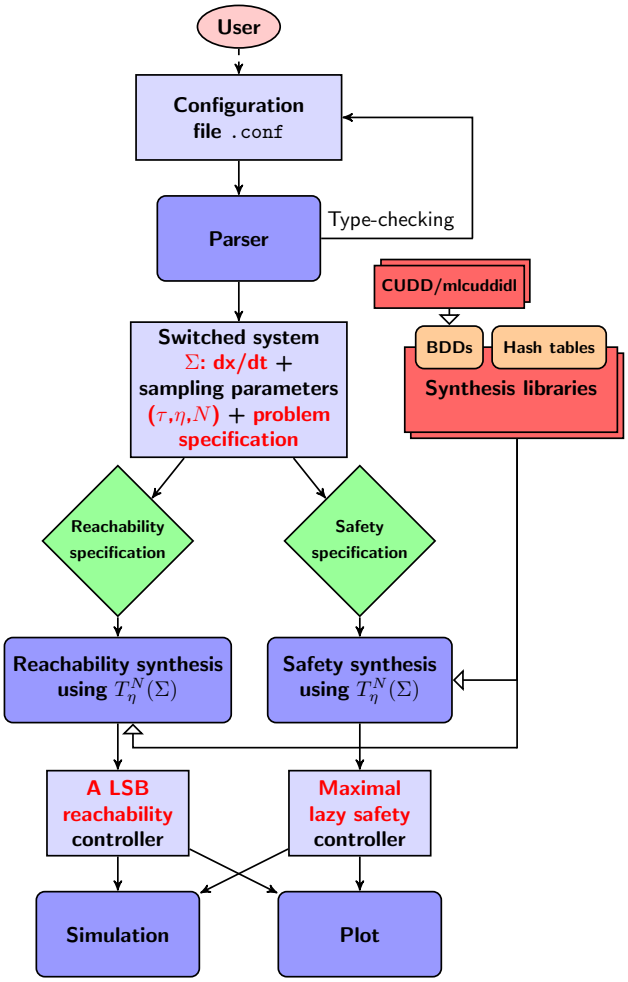


Figure 6: Architecture of CoSyMA

For time-bounded reachability specifications (Q_S, Q_T, B), it implements Algorithm 1 computing the static reachability controller $\mathcal{R}^{ls}_{\mathcal{V}}$ where $\mathcal{V}$ is a complete sub-controller of the MLBR controller $\mathcal{R}^m$ for $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B) (with $T = T^N_{\tau,\eta}(\Sigma)$). The algorithm is a depth-first traversal of paths $\sigma \in \Phi(T^N_{\tau,\eta}(\Sigma))$ starting from initial states I until reaching Q_T to keep trace of the clocks of the states reached by σ. From an operational point of view, the complete sub-controller $\mathcal{V} \subseteq \mathcal{R}^m$, according to which the static reachability controller $\mathcal{R}^{ls}_{\mathcal{V}}$ is defined, is computed according to the order of exploration of initial states I. By keeping the problem at its original dimension, the synthesis complexity is significantly reduced.

3.1 Configuration file

A configuration must be encapsulated in a preamble declared as follows

Switched *id* { ... }

where `id` is the name of the configuration. The dimension of the system is defined by the statement "Dimension" = `n` where `n` is a strictly positive natural number. The modes are defined by the statement

Modes = { ‘p1 , ... , ‘pk };

where ‘p1, ..., ‘pk for $k >= 1$ are the different modes. The keyword "Constants" is followed by constants and their values used to define the differential equation systems. The variable declaration are defined after the keyword "Variables". The variables may be dependent or independent of time. Time-dependent variables are declared as follows:

v1, *v2* : real -> real;
v3 : real -> [5.2 .. 8.9];

where *v1* and *v2* is a time-dependent variable representing a function belonging to $\mathbb{R}^+ \rightarrow \mathbb{R}$ and *v3* is a time-dependent variable representing a function belonging to $\mathbb{R}^+ \rightarrow [5.2, 8.9]$. By cons, the time-independent variables are declared as follows:

v4 : real ;
v5 : [5.2 .. 8.9] ;

The variables are used to describe the functions used to represent the trajectory fields of a system $\mathbf{x}(t)$. After the keyword "Trajectory", we define the piecewise function $\mathbf{x}(t)$ representing the trajectory in terms of time-dependent variables. Many trajectories can be defined. For example, we define a trajectory $\mathbf{x}(t)$ by the following statement.

x = [*v1*,*v2*] ;

We mention that the special variable "@t" is used for the time parameter. Under the keyword "Coefficients", given a linear differential equation systems $\dot{\mathbf{x}}(t) = a(t)\mathbf{x}(t) + b(t)$, we define the coefficients a and b. Two types of coefficients are allowed : time-dependent matrices and time-dependent vectors.

a : Matrix (2);
b : Vector (2);

The type Matrix (n) represents square matrices of dimension n and Vector (n) represents vectors of dimension n. For example, we can define the matrix a by the following statement

a = [[*c1* * @t, sin(2 * @t) + 1],[5, 2 * @t];

and the vector b by the following one.

b = [5 * @t, *c2* + #t];

where *c1* and *c2* are constants. The keyword "VectorFields" is used to define the differential equation system $\dot{\mathbf{x}}(t) = f_p(\mathbf{x}(t))$ of the declared modes p. The differential equation systems may be *simplified* or *expanded*. The statement used to define a linear differential system for a given mode p is the following

simplified when ‘p :: x_dot = a * x + b;

where x is the system trajectory. The statement used to define an expanded differential equation system for a given mode q is the following

nonlinear when ‘q ::
$v1$_dot = $v1$ + cos($v2$) - @t,
$v2$_dot = ln($v1$);

where $v1$ and $v2$ are the elements of x. Under the keyword “SamplingParameters”, we declare the time and space sampling parameters, and a desired finer refinement scale.

time = 0.5 ;
space = 1 / (40 * sqrt (2));
finer = 1;

CoSyMA allows also the declaration of state-spaces (see the grammar in Appendix A), and the specification of safety and time-bounded reachability problems. The command for the controller synthesis for safety problems is written as follows

SafetySynthesis (*x*,*safe*,*init*,*ds*,*rk4s*,*hash_size*) ;

where x is the trajectory identifier, *safe* is the identifier of safe state-space, *init* is the identifier of initial state-space, *ds* equals to “Enum” or “Bdd” according to the desired abstraction representation, *rk4s* indicates the precision (number of iterations) of the fourth-order Runge-Kutta method, and finally *hash_size* is the initial size of the hash table used to represent the abstraction (it is considered only when *ds* = “Enum”). The command time-bounded reachability problems is written as follows

ReachabilitySynthesis (*x*,*safe*,*init*,*target*,*b*,*ds*,*rk4s*,*hash_size*) ;

where *target* is the target state-space and b is a natural such that the maximal time bound $B = b\tau$.

Finally, the tool CoSyMA also allows to generate TikZ [12] plots to visualize controllers and simulate them. It allows the generation of plots for systems of two, three, and four dimensions. The plot configuration for two dimensional systems is defined as follows.

Plot2D (*dora*,*mors*,*modes_colors*,*scales_colors*,*x_res*,*y_res*) ;

where *dora* equals to “dots” or “arrows” according to the desired plot with dots (printing only states) or arrows (printing states and enabled transitions), *mors* equals to “modes” if we want to print states according to the modes of their outgoing transitions or “scales” if we want to print states according to the durations of their outgoing transitions. The lists *modes_colors* and *scales_colors* represents respectively the colors used for modes and durations. The reals *x_res* and *y_res* represents respectively the resolution of the x-axis and the y-axis. The plot configurations Plot2Dof3D and Plot2Dof4D are similar to Plot2D except we have to add after the parameter *y_res* parameter the variables of the trajectory corresponding to the x-axis and y-axis by indicating the values of the rest of variables.

3.2 Abstraction representation and manipulation

The user has the choice to represent the system abstractions either by *enumerated types* or *boolean functions*. The tool uses *hash tables* as an enumerated type to operate on the system abstractions. Hash tables turn out to be more efficient than search trees or other table lookup structures, especially for large numbers of entries.

Alternatively, the tool can use BDDs (*Binary Decision Diagrams*) [2] to represent the system abstractions. They are able to represent sets and relations compactly in memory as boolean functions. We implement BDDs using the modules `Cudd.Man` and `Cudd.Bdd` of the OCaml IDL interface MLCUDDIDL [7] of the CUDD (*CU Decision Diagram*) package [10]. However, using BDDs makes the controller synthesis algorithms presented above more costly than hash tables since the symbolic abstraction is constructed by enumerating the states and computing their successors using the Runge-Kutta method.

4 Experimental results

DC-DC Converter

As a first case study, we apply our approach to a boost DC-DC converter. It is a switched system with two modes, the two dimensional dynamics associated with both modes are affine of the form $\dot{\mathbf{x}}(t) = a_p\mathbf{x}(t) + b$ for $p \in \{1,2\}$ (see [6] for numerical values). It can be shown that it is incrementally stable and thus approximately bisimilar discrete abstractions can be computed. We consider the problem of keeping the state of the system in a desired region of operation given by the safe set $O_S = [1.15, 1.55] \times [5.45, 5.85]$.

We use approximately bisimilar abstractions to synthesize MLS controllers for the DC-DC converter. We compare the cost of controller synthesis for the uniform abstraction $T^0_{\tau_1,\eta_1}$ for parameters $\tau_1 = 0.5s$ and $\eta_1 = 10^{-3}\sqrt{2}/4$ (containing transitions of duration 0.5s) and the multi-scale abstractions $T^2_{\tau_2,\eta_2}$ for parameters $\tau_2 = 4\tau_1$ and $\eta_2 = 4\eta_1$ (containing transitions of durations in $\Theta^2_\tau = \{2s, 1s, 0.5s\}$). These two abstractions have the same precision. Table 3 details the experimental results obtained for the synthesis of the $T^0_{\tau_1,\eta_1}$ and $T^2_{\tau_2,\eta_2}$. We can see that there is a noteworthy reduction of the time used to compute the controller using multi-scale abstractions instead of using uniform ones (up to a 86% improvement between $T^0_{\tau_1,\eta_1}$ and $T^2_{\tau_2,\eta_2}$). This is due to the fact that the size of uniform abstractions grows exponentially with higher resolutions, whereas using multi-scale abstractions are refined only when we get closer to unsafe regions (reduction of more than 91% between $T^0_{\tau_1,\eta_1}$ and $T^2_{\tau_2,\eta_2}$). Interestingly, this reduction in computation time and size does not affect the performance of the multi-scale controllers, which yield a ratio of controllable initial states[1] (CR) over the safety specification comparable to that of their its uniform counterparts. It is worth emphasizing that using CoSyMA there is a remarkable reduction of the computation times compared to those reported in in [3] obtained by a prototype implementation of the algorithm. Figure 7 depicts the maximal lazy safety controller for $T^2_{\tau_2,\eta_2}$ and Q_S and the trace of its simulation starting from the state (1.15,5.6).

Now, we consider the time-bounded reachability specification (Q_S, Q_T, B) where $Q_S = [0.65, 1.65] \times [4.95, 5.95]$, $Q_T = [1.1, 1.6] \times [5.4, 5.9]$ and $B = 20s$. We synthe-

[1] The ratio of controllable initial states for a controller $\mathcal{S} : Q \to 2^L$ and a system $T = (Q, L, \to, O, H, I)$ is computed as $|\{q \in I | \mathcal{S}(q) \neq \emptyset\}|/|I|$.

	Abstractions $T^N_{\tau,\eta}$	
	$N=0,\tau=0.5s$, $\eta=10^{-3}\sqrt{2}/4$	$N=2,\tau=2s$, $\eta=10^{-3}\sqrt{2}$
Time	8.32s	1.10s
Size	599 294	53 479
$\delta(l)$	0.5s (100%)	2s (63.61%) 1s (31.67%) 0.5s (4.72%)
CR	93.52%	93.51%

Table 3: Experimental results for the MLS controller synthesis for the boost DC-DC converter

size static reachability controllers respectively for $T^0_{\tau_1,\eta_1}$ where $\tau_1 = 0.25$ and $\eta_1 = 10^{-3}\sqrt{2}/4$, and $T^2_{\tau_2,\eta_2}$ where $\tau_2 = 4\tau_1$ and $\eta_2 = 4\eta_1$, and the specification (Q_S, Q_T, B).

As shown in Section 2.4, the MLBR controller for the abstraction $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B) where T equal to $T^0_{\tau_1,\eta_1}$ or $T^2_{\tau_2,\eta_2}$ is the maximal lazy safety controller for $C(T_{\circlearrowleft Q_T})$ and the safety specification $Q_S \times [0, B]$. Its computation is costly because the problem is grown form 2 to 3 dimensions by considering the supplementary clock parameter. Synthesizing a static reachability controller rather than an MLBR controller significantly reduces complexity. In Table 4, we observe a considerable reduction of the size of the controlled abstraction of $T^2_{\tau_2,\eta_2}$ of more than 91.46% compared to that of $T^0_{\tau_1,\eta_1}$ with comparable controllability ratios of initial states. Also, the computation time of the controller of $T^2_{\tau_2,\eta_2}$ is significantly shorter than that of $T^0_{\tau_1,\eta_1}$.

	Abstractions $T^N_{\tau,\eta}$	
	$N=0,\tau=0.5s$, $\eta=10^{-3}\sqrt{2}/4$	$N=2,\tau=2s$, $\eta=10^{-3}\sqrt{2}$
Time	658 s	223 s
Size	3 149 538	262 593
$\delta(l)$	0.5s (100%)	2s (72.26%) 1s (15.77%) 0.5s (11.97%)
CR	89.97%	90.30%

Table 4: Experimental results for the static reachability controller for the boost DC-DC converter

Now, we consider the time-bounded reachability specification (Q_S, Q_T, B) where $Q_S = [0.65, 1.65] \times [4.95, 5.95]$, $Q_T = [1.1, 1.6] \times [5.4, 5.9]$ and $B = 20s$. We synthesize static reachability controllers respectively for $T^0_{\tau_1,\eta_1}$ where $\tau_1 = 0.25$ and $\eta_1 = 10^{-3}\sqrt{2}/4$, and $T^2_{\tau_2,\eta_2}$ where $\tau_2 = 4\tau_1$ and $\eta_2 = 4\eta_1$, and the specification (Q_S, Q_T, B).

As shown in Section 2.4, the MLBR controller for the abstraction $C(T_{\circlearrowleft Q_T})$ and (Q_S, Q_T, B) where T equal to $T^0_{\tau_1,\eta_1}$ or $T^2_{\tau_2,\eta_2}$ is the maximal lazy safety controller for $C(T_{\circlearrowleft Q_T})$ and the safety specification $Q_S \times [0, B]$. Its computation is costly because the problem is grown form 2 to 3 dimensions by considering the supplementary clock parameter. Synthesizing static reachability controllers rather than MLBR controllers, reduces widely the synthesis complexity. In Table 4, we observe a considerable reduction of the size of the controlled abstraction of $T^2_{\tau_2,\eta_2}$ of more than 91.46% compared to that of $T^0_{\tau_1,\eta_1}$ with comparable controllability ratios of initial states. Also, the com-

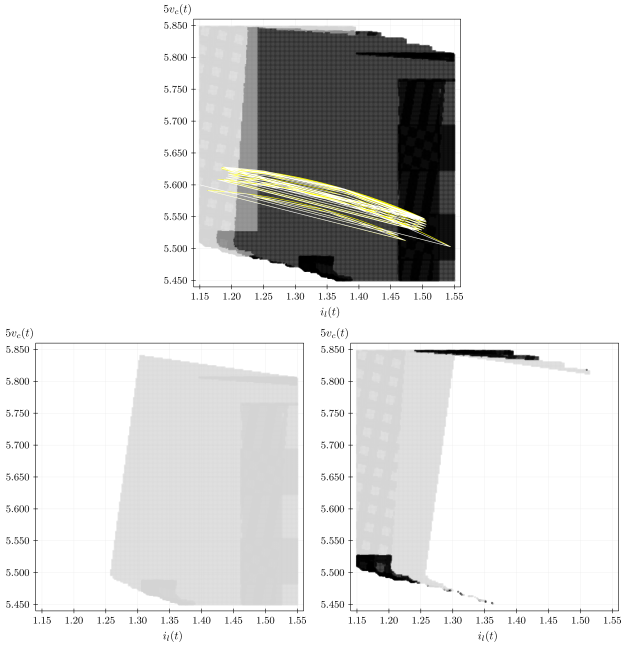


Figure 7: The MLS controller for $T^2_{\tau_2,\eta_2}$ and Q_S. Top: mode 1 is activated (light gray); mode 2 is activated (black); modes 1 and 2 (gray); Bottom (Left): actions of 2s are enabled (light gray); Bottom (right): actions of 1s are enabled (light gray), actions of 0.5s are enabled (black).

putation time of the controller of $T^2_{\tau_2,\eta_2}$ is widely shorter than that of $T^0_{\tau_1,\eta_1}$. Figure 8 depicts a static reachability controller for $T^2_{\tau_2,\eta_2}$ and (Q_S, Q_T, B). The trace, shown on the figure, is the simulation of the controller from the state (0.65,5.4) until reaching the target.

Building Temperature Regulation

The second case study deals with temperature regulation in a circular building. Each room is equipped with a heater and at a given instant at most one heater is switched on. The temperature t_i of the room i is defined by the differential equation $\dot{t}_i = \alpha(t_{i+1} + t_{i-1} - 2t_i) + \beta(t_e - t_i) + \gamma(t_h - t_i)u_i(t)$ where t_{i-1} is the temperature of the room $i-1$; t_{i+1} the temperature of the room $i+1$; t_e is the temperature of the external environment of the building; t_h is the temperature of the heater; α is the temperature transfer ratio between the rooms $i \pm 1$ and the room i; β is the temperature transfer ratio between the external environment and the room i; γ is the temperature transfer ratio between the heater and the room i; $u_i(t)$ equals to 1 if the room i is heated, or 0 otherwise. Given a number $n \geq 2$ of rooms, we distinguish $n+1$ switching modes. For $1 \leq i \leq n$, the mode p_i represents the mode of activating the heater of room i. The mode p_{n+1} represents that no heater is activated. The values of α, β, γ, t_e, and t_h are respectively 1/20, 1/200, 1/100, 10, and 50. We will increase the system dimension to test the limits of the tool in terms of memory usage and computation time. Given the safety specification $Q_S = [20.0, 22.0]^n$ for $n \in \{3,4,5\}$, we synthesize safety controllers for buildings of three, four, and five rooms. The values of τ and η are given in Table 5. By looking

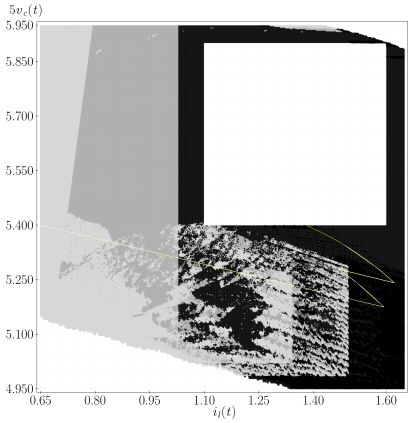


Figure 8: A static reachability controller for $T^2_{\tau_2,\eta_2}$ and (Q_S, Q_T, B): mode p_1 (light gray); mode p_2 (black); both modes (medium gray)

to the results, we can see the combinatorial explosion of the size of abstractions by increasing the system dimension from 3 to 5. Also, it makes sense that the ratio of controllability of initial states decreases by increasing the number of rooms. On our machine equipped with 4GiB of RAM, synthesis fails for the 6-dimensional instance due to to memory overflow.

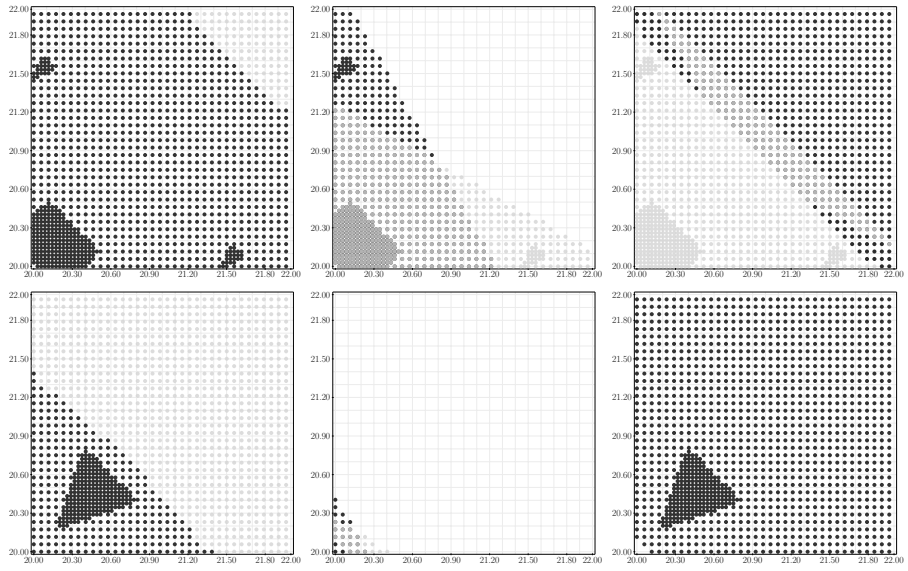
	Abstractions $T^N_{\tau,\eta}$		
	$n=3, N=2,$ $\eta = 50\times10^{-3}$ $\tau = 20s$	$n=4, N=2,$ $\eta = 50\times10^{-3}$ $\tau = 20s$	$n=5, N=1$ $\eta = 0.1$ $\tau = 10s$
Time	2.40s	595 s	571 s
Size	55 564	3 927 564	6 135 218
$\delta(l)$	20s (20.06%) 10s (79.94%) 5s (0%)	20s (8.77%) 10s (86.99%) 5s (4.24%)	 10s (86.44%) 5s (13.56%)
CR	99.99%	99.89%	99.79%

Table 5: Comparison of experimental results for the safety synthesis for the temperature regulator system of three, four, and five dimensions

Figure 9 shows the MLS controller for the transition system $T^2_{20,0.05}$ of three dimension and the safety specification $Q_S = [20.0, 22.0]^3$. The plots are slices of the state space in the dimensions (t_1, t_2) for a fixed $t_3 \approx 20^\circ$ (left) and $t_3(t) \approx 22^\circ$ (right), respectively. The plots on the top depicts scales and those in the middle and the bottom depicts modes. We can remark the predominance of the mode p_4 (no heater is activated) by increasing the temperature of third room.

5 Conclusion

In this paper we have introduced CoSyMA, a tool that automatically synthesizes controllers for incrementally stable switched systems based on multi-scale discrete abstractions. We have illustrated by examples the synthesized controllers for safety and



$t_3(t) \approx 20$ $t_3(t) \approx 22$

Figure 9: MLS controller for $T^2_{20,0.05}$ of three dimension and Q_S; Horizontal axis: $t_1(t)$; Vertical axis: $t_2(t)$; Top: $t_3(t) \approx 20$; Bottom: $t_3(t) \approx 22$; Left: actions of 20s are enabled (black); actions of 10s are enabled (gray); Middle: mode p_1 (black); mode p_2 (light gray); p_1 and p_2 (gray); Right: mode p_4 (black); mode p_3 (light gray); p_3 and p_4 (gray).

time-bounded reachability problems. The benchmarks provide evidence that the use of multi-scale abstractions leads to a substantial reduction of synthesis time and size of the obtained controller while maintaining coverage of the state space.

References

[1] D. Angeli. A lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, 47(3):410 –421, March 2002.

[2] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.

[3] J. Cámara, A. Girard, and G. Gössler. Safety Controller Synthesis for Switched Systems Using Multi-Scale Symbolic Models. In *IEEE on Decision and Control and European Control Conference, CDC-ECC-2011*, pages 520–525, Orlando, USA, 2011. IEEE.

[4] J. Cámara, A. Girard, and G. Gössler. Synthesis of switching controllers using approximately bisimilar multiscale abstractions. In *Proc. of the 14th Inter. Conf. on Hybrid systems: computation and control*, HSCC '11, pages 191–200, New York, NY, USA, 2011. ACM.

[5] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782 –798, may 2007.

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

[7] B. Jeannet. MLCUDDIDL: OCaml interface for CUDD library, version 2.2.0, February 2011. `http://pop-art.inrialpes.fr/~bjeannet/mlxxxidl-forge/mlcuddidl/html/Cudd.html`.

[8] X. Leroy, D. Doligez, A. Frisch, J. Garrigue, D. Rémy, and J. Vouillon. The Objective Caml system release 3.12. Documentation and user manual, 2010.

[9] D. Liberzon. *Switching in systems and control*. Birkhäuser Boston, 2003.

[10] F. Somenzi. CUDD: CU Decision Diagram Package. `http://vlsi.colorado.edu/~fabio/CUDD`.

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

[12] T. Tantau. The TikZ and PGF Packages, Manual for version 2.10, 2010.

A The tool Grammar

The grammar of CoSyMA configuration file is defined as follows.

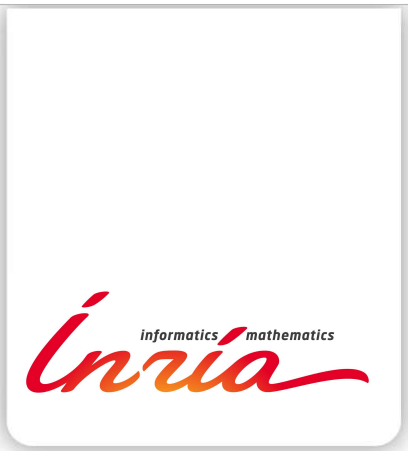
configuration ::=
Switched id { Dimension = natural; Modes = *enum_fileds*; Constants : *constants* Variables : *variables* Trajectories : *trajectories* *maybe_coefs* VectorFields : *vector_fields* SamplingParameters : *sampling_parameters* *spaces* *synthesis_specification* *plot_simulation*}

where

id	::=	string
id_list	::=	(*id*,)* *id*
string_list	::=	(string,)* string
int_list	::=	(int,)* int
real_list	::=	(real,)* real
enum_fileds	::=	(*enum_filed*,)* *enum_filed*
enum_filed	::=	‘string
constants	::=	(*constants_by_value*)* *constants_by_value*
constants_by_value	::=	*id_list* = *xp*;
variables	::=	(*variables_by_type*)* *variables_by_type*
variables_by_type	::=	*id_list* : *variable_type*;
trajectories	::=	(*trajectory*)* *trajectory*
trajectory	::=	*id* = [*id_list*];
maybe_coefs	::=	Coefficients : *coefs_types*; *coefs_values* | <*empty*>
coefs_types	::=	(*coefs_by_type*)* *coefs_by_type*
coefs_by_type	::=	*id_list* : *coef_type*;
coefs_values	::=	(*coefs_by_value*)* *coefs_by_value*
coefs_by_value	::=	*id_list* : *coef_value*
coef_value	::=	*matrix_of_xp* | *vector_of_xp* | *xp*
matrix_of_xp	::=	*xp* array array
vector_of_xp	::=	*xp* array
vector_fields	::=	(*trajectory_vfs*)* *trajectory_vfs*
trajectory_vfs	::=	for *id* : (*trajectory_vf*)* *trajectory_vf*
trajectory_vf	::=	*simplified_vf* | *expanded_vf*
simplified_vf	::=	simplified when *enum_filed* :: *id* `concat` “_dot” = *xp*;
expanded_vf	::=	expanded when *enum_filed* :: (*var_diffeq*)* *var_diffeq*
var_diffeq	::=	*id* `concat` “_dot” = *xp*;
sampling_parameters	::=	time = *xp*; space = *xp*; finer = natural;
spaces	::=	(*space*)* *space*
space	::=	*id* = (*interval*#)* *interval*;
interval	::=	[real .. real]
synthesis_specification	::=	*safety_synthesis* | *reachability_synthesis*
safety_synthesis	::=	SafetySynthesis(*id*,*id*,*id*,*ds*,natural,natural);
reachability_synthesis	::=	ReachabilitySynthesis(*id*,*id*,*id*,*id*,natural,*ds*,natural,natural);
ds	::=	Enum | Bdd
plot_simulation	::=	*plot_2d* | *plot_2d3d* | *plot_2d4d* | *simulation*
plot_2d	::=	Plot2D(*dora*,*mors*,[*string_list*],[*string_list*],real,real,[*int_list*])
plot_2d3d	::=	Plot2Dof3D(*dora*,*mors*,[*string_list*],[*string_list*],real,real,[*int_list*],*id*, *id*,real)
plot_2d4d	::=	Plot2Dof4D(*dora*,*mors*,[*string_list*],[*string_list*],real,real,[*int_list*],*id*, *id*,real,real)
Simulate	::=	Simulate([*real_list*],real,real,int,real,real)

The non-terminals *xp* (expressions), *variable_type* (types of variables), and *coef_type* (type of coefficients), are defined as follows.

xp	::=	@t | *id* | natural | int | real | (*xp*) | + *xp* | - *xp* | *xp* + *xp* | *xp* - *xp* | *xp* * *xp* | *xp* / *xp* | *xp* % *xp* | *xp* ^ *xp* | *unary_fun* (*xp*) | log (*xp*,*xp*)
unary_fun	::=	sin | cos | tg | ctg | sec | csc | asin | acos | atg | sinh | cosh | tanh | sqrt | ln | exp
variable_type	::=	real | *interval* | real -> real | real -> *interval*
coef_type	::=	real -> real | real -> *interval* | Matrix (natural) | Vector (natural)



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399