Memoryless concretization relation

Julien Calbert
julien.calbert@uclouvain.be
ICTEAM Institute, UCLouvain
Louvain-la-Neuve, Belgium

Sébastien Mattenet
sebastien.mattenet@uclouvain.be
ICTEAM Institute, UCLouvain
Louvain-la-Neuve, Belgium

Antoine Girard
antoine.girard@centralesupelec.fr
L2S-CNRS, CentraleSupélec
Université Paris-Saclay
Gif-sur-Yvette, France

Raphaël M. Jungers
raphael.jungers@uclouvain.be
ICTEAM Institute, UCLouvain
Louvain-la-Neuve, Belgium

ABSTRACT

We introduce the concept of memoryless concretization relation (MCR) to describe abstraction within the context of controller synthesis. This relation is a specific instance of alternating simulation relation (ASR), where it is possible to simplify the controller architecture. In the case of ASR, the concretized controller needs to simulate the concurrent evolution of two systems, the original and abstract systems, while for MCR, the designed controllers only need knowledge of the current concrete state. We demonstrate that the distinction between ASR and MCR becomes significant only when a non-deterministic quantizer is involved, such as in cases where the state space discretization consists of overlapping cells. We also show that any abstraction of a system that alternatingly simulates a system can be completed to satisfy MCR at the expense of increasing the non-determinism in the abstraction. We clarify the difference between the MCR and the feedback refinement relation (FRR), showing in particular that the former allows for non-constant controllers within cells. This provides greater flexibility in constructing a practical abstraction, for instance, by reducing non-determinism in the abstraction. Finally, we prove that this relation is not only sufficient, but also necessary, for ensuring the above properties.

CCS CONCEPTS

• **Theory of computation** → **Abstraction**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

KEYWORDS

Symbolic Control, Alternating Simulation, Controller Synthesis, Abstraction, Feedback Refinement.

HSCC '24, May 14–16, 2024, Hong Kong SAR, China

ACM ISBN 979-8-4007-0522-9/24/05
https://doi.org/10.1145/3641513.3650132

ACM Reference Format:
Julien Calbert, Sébastien Mattenet, Antoine Girard, and Raphaël M. Jungers. 2024. Memoryless concretization relation. In *27th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '24), May 14–16, 2024, Hong Kong SAR, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3641513.3650132

1 INTRODUCTION

Abstraction-based control techniques involve synthesizing a correct-by-construction controller through a systematic three-step procedure illustrated on Figure 1. First, both the original system and the specifications are transposed into an abstract domain, resulting in an *abstract system* and corresponding abstract specifications. In this paper, we refer to the original system as the *concrete* system as opposed to the abstract system. Next, an abstract controller is synthesized to solve this abstract control problem. Finally, in the third step, called *concretization*[1] as opposed to *abstraction*, a controller for the original control problem is derived from the abstract controller. The value of this approach lies in the substitution of the concrete system (often a system with an infinite number of states) with a finite state system, which makes it possible to leverage powerful control tools (see [2, 13]), for example from graph theory, such as Dijkstra, the A^* algorithm or dynamic programming, allowing to design correct-by-construction controllers, often with rigorous (safety, or performance) guarantees. Abstraction-based controller design crucially relies on the existence of a *simulation* relation between the concrete and abstract states, allowing to infer the control actions in the concrete system from the ones actuated in the corresponding states of the abstract system. Most approaches are based on the *alternating simulation relation* (ASR) [1], which provides a framework for guaranteeing a concretization step that provably ensures the specifications for the concrete system.

Although the alternating simulation relation offers a safety critical framework, it has several practical drawbacks. The first issue is that this relation provides no complexity guarantee on the concretization procedure, i.e., the concrete controller could contain the entire abstraction (which is typically made up of millions of states and transitions) as a building block, we refer to this problem as the *concretization complexity issue*. See [18, Proposition 8.7] for a description of the algorithmic concretization procedure. The second limitation is that, although ASR guarantees the existence of a controller for the concrete system such that the closed-loop

[1] Note that in other works such as [16], this step is called *refinement procedure*.

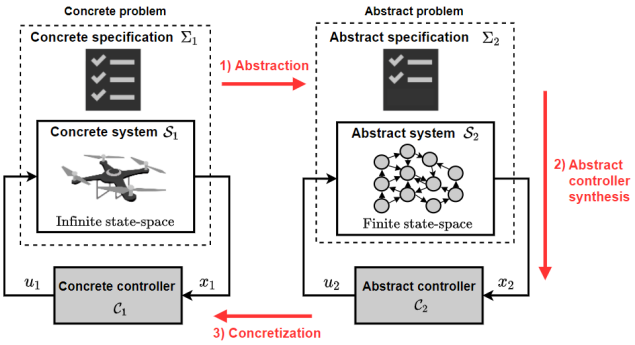

Figure 1: The three steps of abstraction-based control. Our work focuses on the algorithmic burden necessary for the third step, and characterizes the relation between the systems allowing for a memoryless algorithm at step three.

systems share the same behavior, it does not guarantee that some of these properties will be transferred to the concrete controller during the concretization step.

For this reason, various relations have been introduced in the literature, e.g. [3, 6, 15, 17], to tackle specific shortcomings of the alternating simulation relation. Some works propose abstraction-based techniques that do not suffer from this concretization complexity issue, but they are either limited to subsets of specification such as safety or reachability [5, 8, 10, 11], or for a specific class of dynamical systems, e.g. incrementally stable [8], piecewise affine (PWA) dynamics [19]. Other abstraction-based controller synthesis methods circumvent the concretization complexity issue by constructing abstractions without overlapping cells [4, 9, 10, 12, 14, 19]. In particular, in [17], the authors present the *feedback refinement relation* (FRR) which provides a framework for an extremely simple controller architecture, which is not restricted to certain types of specifications or systems, and, as we shall see, forms the basis of our reflection.

In this paper, we study the properties that a simulation relation might require, and establish their implications on controller design characteristics. Specifically, we analyze how the relation established between the concrete and abstract systems during the abstraction phase affects the concretization process, while the methods used to compute abstractions or solve the resulting abstract control problems are beyond the scope of this work. We introduce the *memoryless concretization relation* (MCR) which guarantees a simple control architecture for the concrete system that is not problem-specific. Unlike the general case of alternating simulation relation, the controller only needs information about the current concrete state and does not need to simulate the concurrent evolution of the concrete and abstract systems. We show that in the specific case of a deterministic quantizer (that is, a single-valued map), alternating simulation relation and memoryless concretization relation coincide. As a consequence, the memoryless concretization relation turns out to be essential in the context of an abstraction composed of overlapping cells, a framework we consider crucial for building a non-trivial smart abstraction, see [6]. We prove that MCR is not only sufficient to guarantee the functionality of this memoryless controller architecture, but also necessary for it to work with all abstract controllers, i.e., regardless of the specifications. Finally, we propose an in-depth discussion of feedback refinement relation which turns out to be a special case of memoryless concretization relation, with the additional requirement to use only symbolic state information. As a result, feedback refinement relation is limited to the design of piecewise constant controllers, whereas memoryless concretization relation allows the design of piecewise state-dependent controllers. We illustrate on a simple example the advantage of memoryless concretization relation over feedback refinement relation when concrete state information is available.

Notation: The sets $\mathbb{R}, \mathbb{Z}, \mathbb{Z}_+$ denote respectively the sets of real numbers, integers and non-negative integer numbers. For example, we use $[a, b] \subseteq \mathbb{R}$ to denote a closed continuous interval and $[a; b] = [a, b] \cap \mathbb{Z}$ for discrete intervals. The symbol $\emptyset$ denotes the empty set. Given two sets A, B, we define a *single-valued map* as $f : A \to B$, while a *set-valued map* is defined as $f : A \to 2^B$, where 2^B is the power set of B, i.e., the set of all subsets of B. The image of a subset $\Omega \subseteq A$ under $f : A \to 2^B$ is denoted $f(\Omega)$. We identify a binary *relation* $R \subseteq A \times B$ with set-valued maps, i.e., $R(a) = \{b \mid (a, b) \in R\}$ and $R^{-1}(b) = \{a \mid (a, b) \in R\}$. Given a set A, we denote the binary *identity* relation $Id_A \subseteq A \times A$ such that $(x, y) \in Id_A \Leftrightarrow x = y$. A relation $R \subseteq A \times B$ is *strict* and *single-valued* if for every $a \in A$ the set $R(a) \neq \emptyset$ and $R(a)$ is a singleton, respectively. Given two set-valued maps f, g, we denote by $f \circ g$ their composition $(f \circ g)(x) = f(g(x))$. If $R \subseteq A \times B$ and $Q \subseteq B \times C$, then their composition $R \circ Q = \{(a, c) \in A \times C \mid \exists b \in B$ such that $(a, b) \in R$ and $(b, c) \in Q\}$.

2 PRELIMINARIES

In this section, we start by defining the considered control framework, i.e., the systems, the controllers and the specifications.

We consider dynamical systems of the following form.

Definition 1. *A* transition control system *is a tuple* $\mathcal{S} := (\mathcal{X}, \mathcal{U}, F)$ *where* $\mathcal{X}$ *and* $\mathcal{U}$ *are respectively the set of states and inputs and the set-valued map* $F : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}}$ *where* $F(x, u)$ *give the set of states that can be reached from a given state* x *under a given input* u. ▽

We introduce the set-valued operator of *available inputs*, defined as $\mathcal{U}_F(x) = \{u \in \mathcal{U} \mid F(x, u) \neq \emptyset\}$, which give the set of inputs u available at a given state x. When it is clear from the context which system it refers to, we simply write the available inputs operator $\mathcal{U}(x)$. In this paper, we consider *non-blocking* systems, i.e., $\forall x \in \mathcal{X} : \mathcal{U}(x) \neq \emptyset$.

The use of a set-valued map to describe the transition map of a system allows to model perturbations and any kind of non-determinism in a common formalism. We say that a transition control system is *deterministic* if for every state $x \in \mathcal{X}$ and control input $u \in \mathcal{U}$, $F(x, u)$ is either empty or a singleton. Otherwise, we say that it is *non-deterministic*. A *finite-state* system, in contrast to an *infinite-state* system, refers to a system characterized by finitely many states and inputs.

A tuple $(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}^{[0;T[} \times \mathcal{U}^{[0;T-1[}$ is a *trajectory* of length T of the system $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$ starting at $x(0)$ if $T \in \mathbb{N} \cup \{\infty\}$, $x(0) \in \mathcal{X}$, $\forall k \in [0; T-1[: u(k) \in \mathcal{U}(x(k))$ and $x(k+1) \in F(x(k), u(k))$. The set of trajectories of $\mathcal{S}$ is called the *behavior* of $\mathcal{S}$, denoted $\mathcal{B}(\mathcal{S})$.

Definition 2. *Let* $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$. *The set* $\mathcal{B}(\mathcal{S}) = \{(\boldsymbol{x}, \boldsymbol{u}) \mid \exists T \in \mathbb{N} \cup \{\infty\}$ *such that* $(\boldsymbol{x}, \boldsymbol{u})$ *is a trajectory of* $\mathcal{S}$ *of length* $T\}$, *is called the* behavior *of* $\mathcal{S}$. *We define the* state behavior $\mathcal{B}_x(\mathcal{S})$ *as* $\boldsymbol{x} \in \mathcal{B}_x(\mathcal{S}) \Leftrightarrow \exists \boldsymbol{u} : (\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{B}(\mathcal{S})$.

It is a common practice in the abstraction-based framework to represent systems with an extra set $\mathcal{Y}$ for outputs, and an output mapping function $H : \mathcal{X} \to \mathcal{Y}$. In that case, the states in $\mathcal{X}$ are seen as internal to the system, while the outputs are externally visible. However, as this is not essential to our current discussion, we exclude it here.

We consider systems without *internal variables*, see [16, Definition III.1] and the discussion that follows this definition. This extension is necessary to correctly define the composition of a system with a dynamic controller. For the sake of clarity and to avoid the use of internal variables, we only consider static controllers.

Definition 3. *We define a* static controller *for a system* $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$ *as a set-valued map* $C : \mathcal{X} \to 2^{\mathcal{U}}$ *such that* $\forall x \in \mathcal{X} : C(x) \subseteq \mathcal{U}(x)$, $C(x) \neq \emptyset$. *We define the* controlled system, *denoted as* $C \times \mathcal{S}$, *as the transition system characterized by the tuple* $(\mathcal{X}, \mathcal{U}, F_C)$ *where* $x' \in F_C(x, u) \Leftrightarrow (u \in C(x) \wedge x' \in F(x, u))$.

This controller is *static* since the set of control inputs enabled at a given state only depend on the state. A more general notion of controller (called a *dynamic* controller) would take the entire past trajectory and/or other variables as input. For the sake of clarity we limit ourselves to static controllers although most definitions and proofs extend easily to dynamical controllers.

We now define the control problem.

Definition 4. *Consider a system* $\mathcal{S} = (\mathcal{X}, \mathcal{U}, F)$. *A* specification Σ *for* $\mathcal{S}$ *is defined as any subset* $\Sigma \subseteq (\mathcal{X} \times \mathcal{U})^\infty$. *It is said that system* $\mathcal{S}$ satisfies *the specification* Σ *if* $\mathcal{B}(\mathcal{S}) \subseteq \Sigma$. *A system* $\mathcal{S}$ *together with a specification* Σ *constitute a* control problem $(\mathcal{S}, \Sigma)$. *Additionally, a controller* C *is said to* solve *the control problem* $(\mathcal{S}, \Sigma)$ *if* $C \times \mathcal{S}$ *satisfies the specification* Σ.

Given $\mathcal{X}_I, \mathcal{X}_T, \mathcal{X}_O \subseteq \mathcal{X}$, we define the specific *reach-avoid* specification that we will use to motivate new relations

$$\Sigma^{\text{Reach}} = \{(\boldsymbol{x}, \boldsymbol{u}) \in (\mathcal{X} \times \mathcal{U})^\infty \mid x(0) \in \mathcal{X}_I \Rightarrow \exists k \in \mathbb{Z}_+ : \big(x(k) \in \mathcal{X}_T \wedge \forall k' \in [0; k) : x(k') \notin \mathcal{X}_O\big)\}, \tag{1}$$

which enforces that all states in the initial set $\mathcal{X}_I$ will reach the target $\mathcal{X}_T$ in finite time while avoiding obstacles in $\mathcal{X}_O$. We use the abbreviated notation $\Sigma^{\text{Reach}} = [\mathcal{X}_I, \mathcal{X}_T, \mathcal{X}_O]$ to denote the specification (1).

3 ALTERNATING SIMULATION RELATION

In this paper, we will refer to $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ as the concrete system and $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ as its abstraction.

In practice, the abstract domain $\mathcal{X}_2$ of $\mathcal{S}_2$ is constructed by discretizing the concrete state space $\mathcal{X}_1$ of $\mathcal{S}_1$ into subsets (called *cells*). The discretization is induced by the relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, i.e., the cell associated with the abstract state $x_2 \in \mathcal{X}_2$ is $R^{-1}(x_2) \subseteq \mathcal{X}_1$. Note that in this context, we refer to the set-valued map $R(x_1) = \{x_2 \mid (x_1, x_2) \in R\}$ as the *quantizer*. When R is a single-valued map, we refer to it as defining a *partition* of $\mathcal{X}_1$, in contrast to the case of set-valued maps where we say that it defines a *cover* of $\mathcal{X}_1$, see Figure 2 for clarity. Notably, the condition that R is a strict relation is equivalent to ensuring that the discretization completely covers $\mathcal{X}_1$.

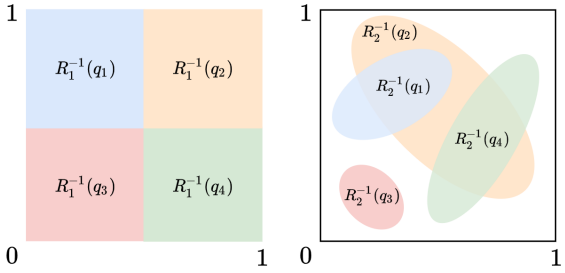

Figure 2: Types of discretization of the concrete state space. Let $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ with $\mathcal{X}_1 = [0, 1]^2$, $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ with $\mathcal{X}_2 = \{q_1, q_2, q_3, q_4\}$, $R_1 \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ and $R_2 \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ are explicit from the figure. Left: R_1 is a strict single-valued map, i.e., it induces a full partition of $\mathcal{X}_1$. Right: R_2 is a non-strict set-valued map, i.e., it induces a partial cover of $\mathcal{X}_1$.

As mentioned in the introduction, the concrete specification Σ_1 must also be translated into an abstract specification Σ_2. For example, given $\Sigma_1^{\text{Reach}} = [\mathcal{X}_I, \mathcal{X}_T, \mathcal{X}_O]$ with $\mathcal{X}_I, \mathcal{X}_T, \mathcal{X}_O \subseteq \mathcal{X}_1$, the abstract specification $\Sigma_2^{\text{Reach}} = [Q_I, Q_T, Q_O]$ with $Q_I, Q_T, Q_O \subseteq \mathcal{X}_2$ must satisfy the following conditions $R(\mathcal{X}_I) \subseteq Q_I, Q_T \subseteq R(\mathcal{X}_T)$ and $R(\mathcal{X}_O) \subseteq Q_O$.

Given an abstract controller C_2 that solves the control problem $(\mathcal{S}_2, \Sigma_2)$, to guarantee the existence of a controller C_1 that solves the concrete problem $(\mathcal{S}_1, \Sigma_1)$, the tuple $(\mathcal{S}_1, \mathcal{S}_2, R)$ must satisfy the following *controlled simulability property*, which guarantees that the behavior of the concrete controlled system $C_1 \times \mathcal{S}_1$ is simulated by the abstract controlled system $C_2 \times \mathcal{S}_2$ via the relation R.

Property 1 (Controlled simulability property-[16, (5)]). *Given two systems* $\mathcal{S}_1$ *and* $\mathcal{S}_2$, *and a relation* $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, *we say that the tuple* $(\mathcal{S}_1, \mathcal{S}_2, R)$ *satisfies the* controlled simulability property *if for every controller* C_2 *there exists a (possibly non-static) controller* C_1 *such that for every trajectory* $\boldsymbol{x}_1$ *of length* T *of the controlled system* $C_1 \times \mathcal{S}_1$, *there exists a trajectory* $\boldsymbol{x}_2$ *of length* T *of the controlled system* $C_2 \times \mathcal{S}_2$ *satisfying*

$$\forall k \in [0; T-1] : \ (x_1(k), x_2(k)) \in R. \tag{2}$$

The condition (2) *can be equivalently reformulated as follows*

$$\mathcal{B}_x(C_1 \times \mathcal{S}_1) \subseteq R^{-1}(\mathcal{B}_x(C_2 \times \mathcal{S}_2)). \tag{3}$$

Controlled simulability can be used to guarantee that properties satisfied by the abstract controlled system $C_2 \times \mathcal{S}_2$ are also satisfied by the concrete controlled system $C_1 \times \mathcal{S}_1$. To design C_1 from C_2 in Property 1, the relation must impose conditions on the local dynamics of the systems in the associated states, accounting for the impact of different input choices on state transitions. The *alternating simulation relation* [18, Definition 4.19] is a comprehensive definition of such a relation. It guarantees that any control applied to $\mathcal{S}_2$ can be concretized into a controller for $\mathcal{S}_1$ that maintains the relation between the controlled trajectories.

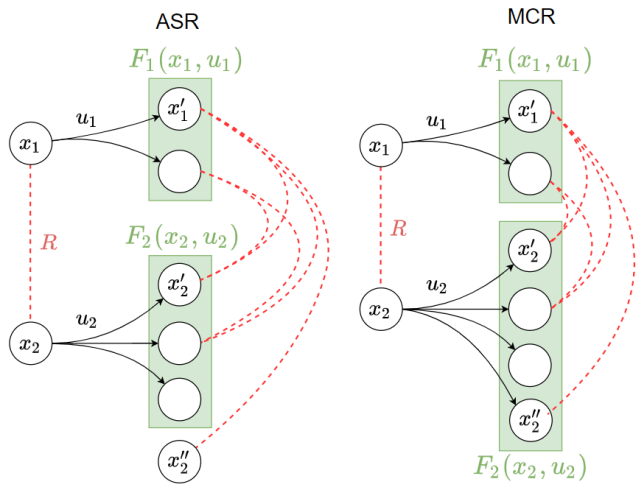

Figure 3: Local conditions between related states to satisfy the relation. Note that here, the relation is fixed (i.e., here a cover of the state space since $|R(x_1')| > 1$) as well as the transition map of the concrete system and that we are simply changing the transition map of the abstract system. Left: The relation R is a ASR **but not a** MCR**. Right: The relation R is a** MCR**.**

Definition 5 (ASR). *A relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is an* alternating simulation relation *from $\mathcal{S}_1$ to $\mathcal{S}_2$ if for every $(x_1, x_2) \in R$ and for every $u_2 \in \mathcal{U}_2(x_2)$ there exists $u_1 \in \mathcal{U}_1(x_1)$ such that for every $x_1' \in F_1(x_1, u_1)$ there exists $x_2' \in F_2(x_2, u_2)$ such that $(x_1', x_2') \in R$.*

Note that sometimes one prefers to speak of the *extended relation* $R_e \subseteq \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{U}_1 \times \mathcal{U}_2$, which is defined by the set of (x_1, x_2, u_1, u_2) satisfying the condition given in the definition of R. When it refers to a ASR, we denote it R_e^{ASR}.

The local condition stated in Definition 5 is illustrated for some $(x_1, x_2, u_1, u_2) \in R_e^{\text{ASR}}$ in Figure 3 (left). The fact that R is an alternating simulation relation from $\mathcal{S}_1$ to $\mathcal{S}_2$ will be denoted $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, and we write $\mathcal{S}_1 \preceq^{\text{ASR}} \mathcal{S}_2$ if $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$ holds for some R.

Note that this definition slightly differs from [18, Definition 4.19], where there is an additional requirement regarding the outputs of related states. This requirement is useful for the concept of approximate alternating simulation relation [18, Definition 9.6] which we do not discuss here.

One needs an *interface* to map abstract inputs to concrete ones.

Definition 6 (Interface). *Given two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, a relation R of type* T*, i.e. $\mathcal{S}_1 \preceq_R^{\text{T}} \mathcal{S}_2$, and its associated extended relation R_e^{T}, a map $I_R^{\text{T}} : \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{U}_2 \to 2^{\mathcal{U}_1}$ is an* interface *from $\mathcal{S}_2$ to $\mathcal{S}_1$ if:*

$$\forall (x_1, x_2) \in R,\ \forall u_2 \in \mathcal{U}_2(x_2),$$

$$I_R^{\text{T}}(x_1, x_2, u_2) \neq \emptyset \tag{4}$$

$$I_R^{\text{T}}(x_1, x_2, u_2) \subseteq \{u_1 \mid (x_1, x_2, u_1, u_2) \in R_e^{\text{T}}\}. \tag{5}$$

The *maximal interface* associated to R satisfies $I_R^{\text{T}}(x_1, x_2, u_2) = \{u_1 \mid (x_1, x_2, u_1, u_2) \in R_e^{\text{T}}\}$.

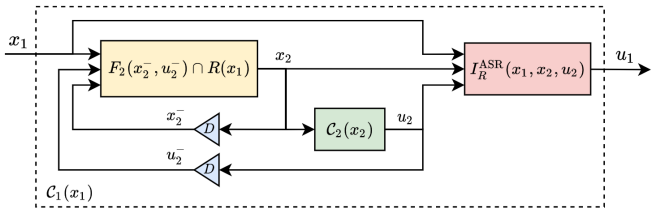

Figure 4: Dynamic controller architecture C_1 such that $(\mathcal{S}_1, \mathcal{S}_2, C_1, C_2, R)$ satisfies the controlled simulability property if $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$. The red block is the *interface* I_R^{ASR}. The yellow block contains the *abstract system* and the *quantizer*. The green block is the *abstract controller*. The blue blocks are *delay blocks* which represent memory elements that store and provide a past value of a signal, e.g., $D(x_2(k)) = x_2(k-1)$.

Theorem 1. *Let two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, and a relation R such that $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, an interface I_R^{ASR}, and any trajectories $(\boldsymbol{x}_1, \boldsymbol{u}_1)$ and $(\boldsymbol{x}_2, \boldsymbol{u}_2)$ of length T where $T \in \mathbb{N} \cup \{\infty\}$ such that $(x_1(0), x_2(0)) \in R$ and*

$$\begin{aligned}
u_2(k) &\in \mathcal{U}_2(x_2(k)) && k \in [0; T-1[;\\
u_1(k) &\in I_R^{\text{ASR}}(x_1(k), x_2(k), u_2(k)) && k \in [0; T-1[;\\
x_1(k+1) &\in F_1(x_1(k), u_1(k)) && k \in [0; T-1[;\\
x_2(k+1) &\in F_2(x_2(k), u_2(k)) \cap R(x_1(k+1)) && k \in [0; T-1[.
\end{aligned}$$

Then, the following holds

$$\begin{aligned}
&(\boldsymbol{x}_1, \boldsymbol{u}_1) \in \mathcal{B}(\mathcal{S}_1),\\
&(\boldsymbol{x}_2, \boldsymbol{u}_2) \in \mathcal{B}(\mathcal{S}_2),\\
&\forall k \in [0; T-1] : (x_1(k), x_2(k)) \in R.
\end{aligned}$$

Proof. We proceed by induction on k. Given that $(x_1(k), x_2(k)) \in R$, which is true for $k = 0$. Since $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, and according to (4) and (5), for every $u_2(k) \in \mathcal{U}_2(k)$, there exists $u_1(k) \in I_R^{\text{ASR}}(x_1(k), x_2(k), u_2(k)) \subseteq \mathcal{U}_1(x_1(k))$. Furthermore, since $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, we can conclude that $F_2(x_2(k), u_2(k)) \cap R(x_1(k+1)) \neq \emptyset$, ensuring that $(x_1(k+1), x_2(k+1)) \in R$. □

Theorem 2. *Given two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, if $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, then $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies the controlled simulability property (Property 1).*

Proof. Given a controller C_2, we can define a controller C_1 implementing the algorithm given in Theorem 1 by taking $u_2(k) \in C_2(x_2(k)) \subseteq \mathcal{U}_2(x_2(k))$, which ensure that $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies the controlled simulability property. □

From Theorem 1, we can construct a concrete controller C_1, whose block diagram is given in Figure 4, guaranteeing the controlled simulability property. This concrete controller simulates the abstraction at every time step which can be computationally expensive if F_2 is hard to compute, e.g., if F_1 itself is hard to compute (or approximate). This is what we refer to as the *concretization complexity issue*. In addition, the proposed controller architecture C_1 requires memory to account for last past abstract input and state, as illustrated on the block diagram in Figure 4.

Note that $\mathcal{S}_1 \preceq^{\text{ASR}} \mathcal{S}_2$ guarantees that $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies controlled simulability property. Nevertheless, this does not guarantee

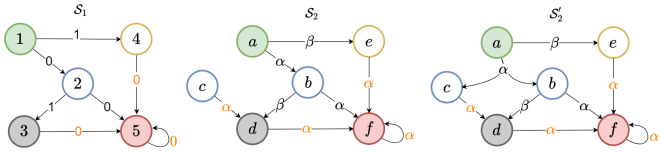

Figure 5: Three transition systems $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ and $\mathcal{S}'_2 = (\mathcal{X}_2, \mathcal{U}_2, F'_2)$ and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. Specifically, $\mathcal{X}_1 = (1, 2, 3, 4, 5)$, $\mathcal{U}_1 = \{0, 1\}$, $\mathcal{X}_2 = (a, b, c, d, e, f)$, $\mathcal{U}_2 = \{\alpha, \beta, \gamma\}$, the transition maps F_1, F_2 and F'_2 and the available inputs maps $\mathcal{U}_1(.)$, $\mathcal{U}_2(.)$ and $\mathcal{U}'_2(.)$ are clear from the illustration, and $R = \{(1, a), (2, b), (2, c), (3, d), (4, e), (5, f)\}$. The colors of the node indicate the related states. Inputs labeled orange simply indicate that their values are not relevant to the current discussion. We consider the concrete specification $\Sigma_1^{\text{Reach}} = [\{1\}, \{5\}, \{3\}]$ and the associated abstract specification $\Sigma_2^{\text{Reach}} = \Sigma_2^{\text{Reach}'} = [\{a\}, \{f\}, \{d\}]$, where the initial, target and obstacle states are respectively in green, red and black.

that any arbitrary property of C_2 will be translated in C_1. In particular the property of a controller being static is certainly not preserved. This will be illustrated in the next section. Another downside is that the controller given here requires simulating the abstract system at every step to know which inputs to allow in the concrete case.

Theorem 1 does not appear formally in [18], but it is suggested in the discussion that follows [18, Def 6.1-Feedback composition]. The content of Theorem 2 was mostly implied in the discussion around [18, Prop 8.7]. In this light, Section 3 gives the necessary background, in standardized form, for the introduction of the memoryless concretization relation.

4 MEMORYLESS CONCRETIZATION RELATION

We start by defining the concrete controller resulting from a specific concretization scheme.

Definition 7 (Memoryless concretized controller). *Given two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, a strict relation R of type* T*, an interface I_R^{T} and a controller C_2, we define the* memoryless concretized controller, *denoted $C_1 = C_2 \circ_{I_R^{\mathrm{T}}} R$ as the mapping*

$$C_1(x_1) = (C_2 \circ_{I_R^{\mathrm{T}}} R)(x_1) = \bigcup_{x_2 \in R(x_1)} I_R^{\mathrm{T}}(x_1, x_2, C_2(x_2)). \quad (6)$$

The term *memoryless* assigned to C_1 is justified by the observation that C_1 exclusively makes decisions based on the current concrete state.

4.1 Motivation

The example below illustrates that the alternating simulation relation guarantees the ability to derive a controller C_1 for $\mathcal{S}_1$ from any controller C_2 of $\mathcal{S}_2$, i.e., $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies the controlled simulability property. However, it also points out that the specific associated memoryless concretized controller $C_1 = C_2 \circ_{I_R^{\mathrm{ASR}}} R$ does not guarantee (3).

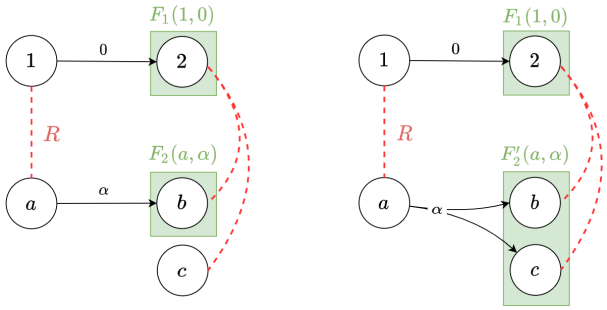

Figure 6: Local transition of two related states $(1, a) \in R$ and relation R defined in Figure 5. Left: Transition for $\mathcal{S}_2$. Right: Transitions for $\mathcal{S}'_2$.

We consider the two deterministic systems $\mathcal{S}_1$ and $\mathcal{S}_2$ given in Figure 5. One can verify from Definition 5 that the relation R is an alternating simulation relation, i.e., $\mathcal{S}_1 \preceq_R^{\mathrm{ASR}} \mathcal{S}_2$. Note that this relation corresponds to a cover-based abstraction, since $R(2) = \{b, c\}$ is not a singleton. We consider the concrete specifications Σ_1 consisting in controlling $\mathcal{S}_1$ from the initial state 1 to the target state 5 while avoiding the obstacle state 3. The associated abstract specifications Σ_2 consist in controlling $\mathcal{S}_2$ from the initial state a to the target state f while avoiding the obstacle state d. The controller C_2 satisfying

$$C_2(a) = \{\alpha\},\ C_2(b) = \{\alpha\} \quad (7)$$

solves the abstract problem. The maximal interface associated to R is given by

$$I_R^{\mathrm{ASR}}(1, a, \alpha) = \{0\},\ I_R^{\mathrm{ASR}}(2, b, \alpha) = \{0\},\ I_R^{\mathrm{ASR}}(2, c, \alpha) = \{1\}.$$

We consider the associated memoryless concretized controller, $C_1 = C_2 \circ_{I_R^{\mathrm{ASR}}} R$. We compute the relevant values:

$$C_1(1) = \{0\},\ C_1(2) = \{0, 1\}.$$

We can construct the trajectory $(\boldsymbol{x}_1, \boldsymbol{u}_1) \in \mathcal{B}(C_1 \times \mathcal{S}_1)$ with $\boldsymbol{x}_1 = (1, 2, 3)$ and $\boldsymbol{u}_1 = (0, 1)$ which leads to a contradiction with the dynamics of $C_2 \times \mathcal{S}_2$ since the controlled simulability property requires the existence of an abstract trajectory $(\boldsymbol{x}_2, \boldsymbol{u}_2)$ such that $x_2(0) = a$ and $x_2(2) = d$. This shows that C_1 does not satisfy (3). As a result, this specific controller does not offer the formal guarantee of avoiding the obstacle and reaching the target.

However, the controller C'_1 defined as

$$C'_1(1) = \{0\},\ C'_1(2) = \{0\}$$

satisfies (3) (note that its existence was guaranteed by Theorem 2, but that there was no guarantee that it was static). Indeed, the trajectory $(\boldsymbol{x}'_1, \boldsymbol{u}'_1) \in \mathcal{B}(C'_1 \times \mathcal{S}_1)$ with $\boldsymbol{x}'_1 = (1, 2, 5)$ and $\boldsymbol{u}'_1 = (0, 0)$ aligns with a corresponding abstract trajectory $(\boldsymbol{x}'_2, \boldsymbol{u}'_2) \in \mathcal{B}(C_2 \times \mathcal{S}_2)$ with $\boldsymbol{x}'_2 = (a, b, f)$ and $\boldsymbol{u}'_2 = (\alpha, \alpha)$, while the previously defined trajectory $(\boldsymbol{x}_1, \boldsymbol{u}_1) \notin \mathcal{B}(C'_1 \times \mathcal{S}_1)$.

On the other hand, if we adopt an abstract controller defined as follows

$$\tilde{C}_2(a) = \{\beta\},\ \tilde{C}_2(e) = \{\alpha\} \quad (8)$$

which also solves the abstract problem, then the controller $\tilde{C}_1$, where $\tilde{C}_1 = \tilde{C}_2 \circ_{I_R^{\text{ASR}}} R$ is the associated memoryless concretized controller, satisfies (3).

Nevertheless, we aim to establish conditions that guarantee that the memoryless concretized controller ensures the controlled simulability property whatever the abstract controller.

In the subsequent subsection, we will introduce a relation that is not only *sufficient* to ensure this specific concretization scheme but is also *necessary* to ensure its feasibility for every abstract controller.

4.2 Definition and properties

The crucial point in the failure of the previous example is that the ASR condition only imposes that for each transition from x_1 to x_1' in $\mathcal{S}_1$ there exists a state $x_2' \in R(x_1')$ that is a successor of x_2 in $\mathcal{S}_2$, but it is not required that every $x_2'' \in R(x_1')$ succeeds x_2 (note that it was the case in the previous example as illustrated in Figure 6). The relation presented below is introduced to circumvent the specific problem describe previously.

Definition 8 (MCR). *A relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a* memoryless concretization relation *from $\mathcal{S}_1$ to $\mathcal{S}_2$ if for every $(x_1, x_2) \in R$ for every $u_2 \in \mathcal{U}_2(x_2)$ there exists $u_1 \in \mathcal{U}_1(x_1)$ such that for every $x_1' \in F_1(x_1, u_1)$ for every x_2' such that $(x_1', x_2') \in R$: $x_2' \in F_2(x_2, u_2)$.*

The local condition stated in Definition 8 is illustrated for some $(x_1, x_2, u_1, u_2) \in R_e^{\text{MCR}}$ in Figure 3 (right). The fact that R is a memoryless concretization relation from $\mathcal{S}_1$ to $\mathcal{S}_2$ will be denoted $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$, and we write $\mathcal{S}_1 \preceq^{\text{MCR}} \mathcal{S}_2$ if $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$ holds for some R.

The memoryless concretization relation is a relaxed version of the feedback refinement relation introduced in [16, V.2 Definition]. Specifically, we relax the requirement that the concrete and abstract inputs must be identical.

Theorem 3. *Given two systems, $\mathcal{S}_1$ and $\mathcal{S}_2$, and a strict relation R, the following statements hold*

(i) *If $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$, then $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$.*
(ii) *Additionally, if R is single-valued (i.e., defines a partition), the converse is true.*

Proof.

(i) Rewriting the definition slightly, we have
- ASR: for every $(x_1, x_2) \in R$ and for every $u_2 \in \mathcal{U}_2(x_2)$ there exists $u_1 \in \mathcal{U}_1(x_1)$ such that for every $x_1' \in F_1(x_1, u_1)$: $R(x_1') \cap F_2(x_2, u_2) \neq \emptyset$.
- MCR: for every $(x_1, x_2) \in R$ and for every $u_2 \in \mathcal{U}_2(x_2)$ there exists $u_1 \in \mathcal{U}_1(x_1)$ such that for every $x_1' \in F_1(x_1, u_1)$: $R(x_1') \subseteq F_2(x_2, u_2)$.

Since R is strict, $R(x_1')$ is nonempty and therefore $R(x_1') \subseteq F_2(x_2, u_2)$ implies $R(x_1') \cap F_2(x_2, u_2) \neq \emptyset$.

(ii) Given that R is a strict relation and single-valued, it can be established that for any $x_1 \in \mathcal{X}_1$: $|R(x_1)| = 1$. Therefore $R(x_1') \cap F_2(x_2, u_2) \neq \emptyset$ if and only if $R(x_1') \subseteq F_2(x_2, u_2)$.

□

Note that the condition that the relation forms a partition is sufficient to guarantee that an ASR is a MCR, but it is not a necessary condition.

We prove that $\preceq^{\text{MCR}}$ is reflexive and transitive, which justifies the use of the pre-order symbol $\preceq$.

Proposition 1. *Let $\mathcal{S}_1, \mathcal{S}_2$ and $\mathcal{S}_3$ be transition systems, and R, Q be strict relations. Then*

(1) $\mathcal{S}_1 \preceq_{Id_{\mathcal{X}_1}}^{\text{MCR}} \mathcal{S}_1$;
(2) *If $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$ and $\mathcal{S}_2 \preceq_Q^{\text{MCR}} \mathcal{S}_3$, then $\mathcal{S}_1 \preceq_{R\circ Q}^{\text{MCR}} \mathcal{S}_3$.*

Proof. The identity relation $Id_{\mathcal{X}_1}$ satisfies the definition of MCR with $\mathcal{S}_1 = \mathcal{S}_2$, $x_1 = x_2$ and $u_1 = u_2$, which proves (1). To prove (2), assume that $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2 \preceq_Q^{\text{MCR}} \mathcal{S}_3$. Then $R \circ Q$ is strict since both R and Q are strict. Let $(x_1, x_3) \in R \circ Q$. Then there exists $x_2 \in \mathcal{X}_2$ such that $(x_1, x_2) \in R$ and $(x_2, x_3) \in Q$. Since both Q and R are MCR, then we have

$$\forall u_3 \in \mathcal{U}_3(x_3) \exists u_2 \in \mathcal{U}_2(x_2) : \; Q(F_2(x_2, u_2)) \subseteq F_3(x_3, u_3);$$
$$\forall u_2 \in \mathcal{U}_2(x_2) \exists u_1 \in \mathcal{U}_1(x_1) : \; R(F_1(x_1, u_1)) \subseteq F_2(x_2, u_2);$$

from which follows

$$\forall u_3 \in \mathcal{U}_3(x_3) \exists u_1 \in \mathcal{U}_1(x_1) : \; Q(R(F_1(x_1, u_1))) \subseteq F_3(x_3, u_3)$$

and so $\mathcal{S}_1 \preceq_{R\circ Q}^{\text{MCR}} \mathcal{S}_3$. □

Property 2 (Memoryless concretization property). *Given two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, and a strict relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ of type* T*, we say that the tuple $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies the* memoryless concretization property *if if every controller C_2*

$$R(\mathcal{B}_x(C_1 \times \mathcal{S}_1)) \subseteq \mathcal{B}_x(C_2 \times \mathcal{S}_2) \tag{9}$$

with $C_1 = C_2 \circ_{I_R^{\text{T}}} R$.

This property guarantees that the controller $C_1 = C_2 \circ_{I_R^{\text{T}}} R$ can only generate abstract trajectories that belongs to the behavior of the system $C_2 \times \mathcal{S}_2$. To compare and contrast with Property 1, we require two different things. The first is that every quantization of a concrete trajectory is a valid abstract trajectory, as opposed to the existence of one related abstract trajectory. This is preferable if the quantizer is assumed adversarial, or alternatively, if you are free to choose the quantization that suits you best, e.g., the fastest to compute. The second thing that Property 2 requires is that (9) holds for a very specific controller, which gives us a straightforward implementation.

Theorem 4. *Given two transition systems $\mathcal{S}_1$ and $\mathcal{S}_2$, and a strict relation R, if $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$ then the tuple $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies the* memoryless concretization property *(Property 2).*

Proof. Consider a trajectory $(\boldsymbol{x}_1, \boldsymbol{u}_1) \in \mathcal{B}(C_1 \times \mathcal{S}_1)$ of length T with $C_1 = C_2 \circ_{I_R^{\text{MCR}}} R$, i.e.,

(1) $\forall k \in [0; T[: \; x_1(k+1) \in F_1(x_1(k), u_1(k))$;
(2) $\forall k \in [0; T-1[: \; u_1(k) \in C_1(x_1)$.

Then by definition of C_1, for any related sequence $\boldsymbol{x}_2$ of length T such that $\forall k \in [0; T[: \; x_2(k) \in R(x_1(k))$, there exists a sequence $\boldsymbol{u}_2$ of length $T-1$ such that

(1') $\forall k \in [0; T-1[: \; u_2(k) \in C_2(x_2(k))$;
(2') $\forall k \in [0; T-1[: \; u_1(k) \in I_R^{\text{MCR}}(x_1(k), x_2(k), u_2(k))$.

because C_2 and I_R^{MCR} are non-empty.

Our goal is to establish that $(\boldsymbol{x}_2, \boldsymbol{u}_2) \in \mathcal{B}(C_2 \times \mathcal{S}_2)$, which translates to

(1") $\forall k \in [0; T-1[:\ u_2(k) \in \mathcal{U}_2(x_2(k))$;
(2") $\forall k \in [0; T-1[:\ x_2(k+1) \in F_2(x_2(k), u_2(k))$.

To prove (1"), we can directly use condition (1'). To prove (2"), taking into account condition (2') and $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$, we can observe that

$$\forall x_1' \in F_1(x_1(k), u_1(k)) :\ (x_1', x_2') \in R \Rightarrow x_2' \in F_2(x_2(k), u_2(k)). \tag{10}$$

Given the preceding conditions (1'), (2'), and the established implication (10), it follows that $x_2(k+1) \in F_2(x_2(k), u_2(k))$. □

Furthermore, the existence of a memoryless concretization relation between the concrete system and the abstraction is not only sufficient to guarantee memoryless concretization property, it is also necessary when we want it to be applicable whatever the particular specification we seek to impose on the concrete system, as established by the following theorem.

Theorem 5. *Given two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, and a strict relation R such that $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$, if $(\mathcal{S}_1, \mathcal{S}_2, R)$ satisfies the memoryless concretization property (Property 2), then $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$.*

Proof. We proceed by contraposition. Let's assume that R does not satisfy the condition of Definition 8. This implies the existence of $(x_1, x_2) \in R$ and $u_2 \in \mathcal{U}_2(x_2)$ such that for every $u_1 \in \mathcal{U}_1(x_1)$, there exist $x_1' \in F_1(x_1, u_1)$ and $x_2' \in R(x_1')$ such that $x_2' \notin F_2(x_2, u_2)$. We consider an abstract controller C_2 such that $C_2(x_2) = \{u_2\}$. Let $u_1 \in I_R^{\text{MCR}}(x_1, x_2, u_2) \subseteq \mathcal{U}_1(x_1)$. Let's define the sequences $\tilde{\boldsymbol{x}}_1 = (x_1, x_1')$, $\tilde{\boldsymbol{u}}_1 = (u_1)$, $\tilde{\boldsymbol{x}}_2 = (x_2, x_2')$, and $\tilde{\boldsymbol{u}}_2 = (u_2)$. We observe that $(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{u}}_1) \in \mathcal{B}(C_1 \times \mathcal{S}_1)$ with $C_1 = C_2 \circ_{I_R^{\text{MCR}}} R$ while $(\tilde{\boldsymbol{x}}_2, \tilde{\boldsymbol{u}}_2) \notin \mathcal{B}(C_2 \times \mathcal{S}_2)$ since $\tilde{x}_2(1) \notin F_2(x_2, u_2)$. This means that the tuple $(\mathcal{S}_1, \mathcal{S}_2, R)$ does not satisfy the memoryless concretization property. This concludes the proof. □

From any given ASR abstraction, it is possible to construct a MCR abstraction, albeit with an increase in non-determinism. This augmentation is achieved by introducing additional transitions, as formally established by the subsequent proposition.

Proposition 2. *Consider two systems $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$ and $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$ and a relation R that satisfies $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$. Then the system $\mathcal{S}_2' = (\mathcal{X}_2, \mathcal{U}_2, F_2')$ such that $\forall x_2 \in \mathcal{X}_2$ and $\forall u_2 \in \mathcal{U}_2(x_2)$:*

(i) $\mathcal{U}_2'(x_2) = \mathcal{U}_2(x_2)$;

(ii) $F_2'(x_2, u_2) = F_2(x_2, u_2) \cup \left(\bigcup_{(x_1, x_2, u_1, u_2) \in R_e^{\text{ASR}}} R(F_1(x_1, u_1)) \right)$;

satisfies $\mathcal{S}_2 \preceq_{Id_{\mathcal{X}_2}}^{\text{MCR}} \mathcal{S}_2'$ and $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2'$.
We call it the MCR*-extension of $\mathcal{S}_2$.*

Proof. The condition of Definition 8, i.e., $\forall (x_2, x_2) \in Id_{\mathcal{X}_2}$ and $\forall u_2' \in \mathcal{U}_2'(x_2)$ there exists $u_2 \in \mathcal{U}_2(x_2)$ such that $F_2(x_2, u_2) \subseteq Id_{\mathcal{X}_2}(F_2'(x_2, u_2'))$, holds by taking $u_2 = u_2'$ (by (i)), and the inclusion is ensured by (ii). This conclude the proof that $\mathcal{S}_2 \preceq_{Id_{\mathcal{X}_2}}^{\text{MCR}} \mathcal{S}_2'$.

Since $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2$ and $\mathcal{S}_2 \preceq_{Id_{\mathcal{X}_2}}^{\text{MCR}} \mathcal{S}_2'$, by the transitivity of ASR, we can easily conclude that $\mathcal{S}_1 \preceq_{R \circ Id_{\mathcal{X}_2}}^{\text{ASR}} \mathcal{S}_2'$, which is equivalent to $\mathcal{S}_1 \preceq_R^{\text{ASR}} \mathcal{S}_2'$. In addition, by (ii), the MCR condition, i.e., $\forall (x_1, x_2) \in R$ and $\forall u_2 \in \mathcal{U}_2(x_2)$, $\exists u_1 \in \mathcal{U}_1(x_1) :\ R(F_1(x_1, u_1)) \subseteq F_2'(x_2, u_2)$, holds. This completes the proof that $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2'$. □

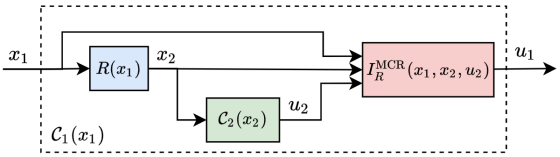

Figure 7: Memoryless concretized controller architecture $C_1 = C_2 \circ_{I_R^{\text{MCR}}} R$ such that $(\mathcal{S}_1, \mathcal{S}_2, C_1, C_2, R)$ satisfies the controlled simulability property if $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$.

Note that condition (i) limits the addition of a transition to existing labeled transitions.

We stress that the system $\mathcal{S}_2' = (\mathcal{X}_2, \mathcal{U}_2, F_2')$ of Proposition 2 is not the only way[2] to define a MCR-extension of $\mathcal{S}_2$. We chose this definition to show existence of a MCR-extension.

4.3 Concretization procedure

As already mention with the motivating example of this section, the memoryless concretization relation enables simpler concrete control architecture illustrated in Figure 7. The advantage of this control architecture is that the concrete controller only depends on the current concrete state and does not need to keep track of where the abstract system is supposed to be. In practice the abstraction is not needed once the abstract controller has been designed. Note, if the abstract controller C_2 is static, then the concretized controller will be static as well.

Nevertheless, the interface I_R^{MCR} implicitly embeds the extended relation R_e^{MCR} into the concrete controller. A very convenient case is when the interface is a function for which we have an explicit characterization, as we can simply compute it as needed, removing the implementation cost of storing the extended relation. For example, consider the context of a piecewise affine controller for the concrete system, i.e., where given $x_2 \in \mathcal{X}_2$ and $u_2 \in C_2(x_2)$, we have $\forall x_1 \in R^{-1}(x_2) :\ I_R^{\text{MCR}}(x_1, x_2, u_2) = \kappa(x_1) = \mathbf{K}x_1 + l$. The abstract input $u_2 \in C_2(x_2)$ can be interpreted as the local controller κ for as long as we are in the cell $R^{-1}(x_2) \subseteq \mathcal{X}_1$.

Let's now return to the motivating example of this section given in Figure 5. First note that the relation R is not a MCR for systems $\mathcal{S}_1$ and $\mathcal{S}_2$ ($\mathcal{S}_1 \not\preceq_R^{\text{MCR}} \mathcal{S}_2$) due to the following observations: $(1, a) \in R$, $\alpha \in \mathcal{U}_2(a)$, $0 \in \mathcal{U}_1(1)$, $2 \in F_1(1, 0)$, $c \in R(2)$ and $c \notin F_2(a, \alpha)$ as illustrated in Figure 6. We will now turn our attention to system $\mathcal{S}_2'$ introduced in Figure 5. We can prove that $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2'$ and that, consequently, Theorem 4 guarantees that any memoryless concretized controller will satisfy the controlled simulability property. However, the gain in this property comes at the cost of an increase in non-determinism in the abstraction (note that $\mathcal{S}_2'$ is the MCR-extension of $\mathcal{S}_2$): whereas $\mathcal{S}_2$ was deterministic, $\mathcal{S}_2'$ is now non-deterministic. As a result, the abstract problem is harder to solve since it is no longer a simple path-finding problem on a directed graph, but a reachability problem on a weighted directed *forward hypergraph*. That is, each transition corresponds to a *forward hyperarc* which is a hyperarc with one tail and multiple heads, see [7] for an introduction to hypergraphs. In addition, whereas

[2] Intuitively, we are extending along all u_1 related by ASR to u_2, when extending along only one would suffice, but in general, the choice of u_1 is not so well defined.

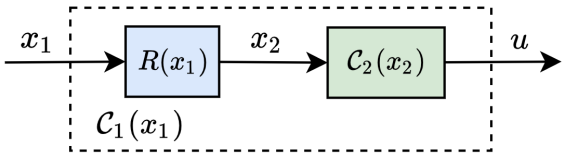

Figure 8: Memoryless concretized controller architecture $C_1 = C_2 \circ R$ for $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$.

the abstract problem $(\mathcal{S}_2, \Sigma_2)$ had two solutions, the controllers C_2 given by (7) and C_2' given by (8), the abstract problem $(\mathcal{S}_2', \Sigma_2')$ has only one solution, the controller C_2'. So, not only may the abstract problem $(\mathcal{S}_2', \Sigma_2')$ be more difficult to solve, it may also admit less or no solution at all.

4.4 Comparison between MCR and FRR

As previously mentioned, the MCR extends to the feedback refinement relation [16, V.2 Definition], whose definition we recall.

Definition 9 (FRR). *A relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ is a* feedback refinement relation *from $\mathcal{S}_1$ to $\mathcal{S}_2$ if for every $(x_1, x_2) \in R$:*

(1) $\mathcal{U}_2(x_2) \subseteq \mathcal{U}_1(x_1)$;
(2) *for every $u \in \mathcal{U}_2(x_2)$ and for every $x_1' \in F_1(x_1, u)$ for every x_2' such that $x_2' \in R(x_1')$: $x_2' \in F_2(x_2, u)$.*

The fact that R is a feedback refinement relation from $\mathcal{S}_1$ to $\mathcal{S}_2$ will be denoted $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$, and we write $\mathcal{S}_1 \preceq^{\text{FRR}} \mathcal{S}_2$ if $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$ holds for some R.

Proposition 3. *Given two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, if $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$, then $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2$.*

Proof. It's the same definition as MCR, with the additional constraint that the abstract and concrete input in the extended relation are the same, that is to say $I_R^{\text{MCR}}(x_1, x_2, u_2) = \{u_2\} \subseteq \mathcal{U}_1(x_1)$. □

In this case, the controller architecture in Figure 7 simplifies with $I_R^{\text{MCR}}(x_1, x_2, u_2) = \{u_2\}$ in the architecture given in Figure 8. The concrete controller can be rewritten as $C_1 = C_2 \circ_{I_R^{\text{MCR}}} R = C_2 \circ R$, i.e., the functional composition of C_2 and R viewed as set-valued maps. This justifies the notation $\circ_{I_R^{\text{MCR}}}$.

This concrete controller only requires the abstract (or symbolic) state, it does not need the concrete state x_1; we say that the concretization is carried out using only symbolic information. Consequently, FRR restricts its class of concrete controllers exclusively to *piecewise constant controllers*. Therefore, the feedback refinement relation is well suited to the context where the exact state is not known and only quantified (or symbolic) state information is available. On the other hand, when information on the concrete state is available, this is a major restriction, as the following example illustrates.

To motivate the use of a MCR versus a FRR, we provide the following example where, for a given relation R (i.e., a given discretization), we can solve the concrete problem following the three-step abstraction-based procedure described in Section 1 with a MCR whereas this is impossible with a FRR, i.e., using only symbolic information for the concretization.

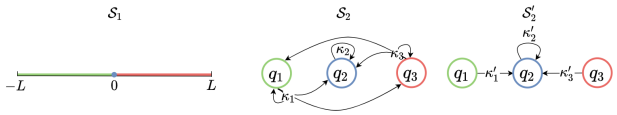

Figure 9: Three transition systems $\mathcal{S}_1 = (\mathcal{X}_1, \mathcal{U}_1, F_1)$, $\mathcal{S}_2 = (\mathcal{X}_2, \mathcal{U}_2, F_2)$, $\mathcal{S}_2' = (\mathcal{X}_2, \mathcal{U}_2', F_2')$ and a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$. Specifically, $\mathcal{X}_1 = [-L, L] \subseteq \mathbb{R}$, $\mathcal{U}_1 = \mathbb{R}$, $\mathcal{X}_2 = \{q_1, q_2, q_3\}$, $\mathcal{U}_2 = \{\kappa_1, \kappa_2, \kappa_3\}$, $\mathcal{U}_2' = \{\kappa_1', \kappa_2', \kappa_3'\}$. The transition maps F_2 and F_2' and the available inputs $\mathcal{U}_2(.)$ and $\mathcal{U}_2'(.)$ are clear from the illustration. Given the function $f(x, u) = x + u$, the available inputs $\mathcal{U}_1(x_1) = \{u \in \mathcal{U}_1 \mid f(x_1, u) \in \mathcal{X}_1\}$ and the transition map $F_1(x_1, u) = f(x_1, u)$. Given the sets $\mathcal{X}_{q_1} = [-L, 0)$, $\mathcal{X}_{q_2} = \{0\}$ and $\mathcal{X}_{q_3} = (0, L]$, we define the relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ such that $(x_1, q) \in R \Leftrightarrow x_1 \in \mathcal{X}_q$. The colors indicate the related states. We consider the concrete specification $\Sigma_1^{\text{Reach}} = [\mathcal{X}_1, \{0\}, \varnothing]$ and the associated abstract specifications $\Sigma_2^{\text{Reach}} = \Sigma_2^{\text{Reach}'} = [\mathcal{X}_2, \{q_2\}, \varnothing]$.

We consider the system $\mathcal{S}_1$ whose the dynamic consists in moving under translation on a segment of the real line, and with objective to reach 0. The full definition of $\mathcal{S}_1$ is in Figure 9. Any abstraction S_2 such that $\mathcal{S}_1 \preceq_R^{\text{FRR}} \mathcal{S}_2$ is constrained to have non-determinism for the cells q_1 and q_3, as exemplified by the system $\mathcal{S}_2$ given in Figure 9. The problem here is that by using piecewise constant controllers for the concrete system, there will always be a portion around 0 that overshoot its target. Indeed, we have $\kappa_1 \in [0, L]$, $\kappa_2 = 0$ and $\kappa_3 \in [-L, 0]$, and therefore, for any admissible choice of $\kappa_1, \kappa_2, \kappa_3$, the resulting abstraction $\mathcal{S}_2$ is non-deterministic. As a result, the abstract problem $(\mathcal{S}_2, \Sigma_2)$ has no solution. However, we can build an abstraction $\mathcal{S}_2'$ such that $\mathcal{S}_1 \preceq_R^{\text{MCR}} \mathcal{S}_2'$ where $\kappa_1'(x_1) = \kappa_3'(x_1) = -x_1$ and $\kappa_2'(x_1) = 0$ are affine controllers. Note that we have the same partition of state-space, but because we can have abstract inputs that are local state-dependent controllers instead of piecewise constant real inputs, we can solve the abstract problem $(\mathcal{S}_2', \Sigma_2')$. For the abstract system $\mathcal{S}_2'$, the inputs κ_1', κ_2' and κ_3' can be interpreted as *move to the right cell*, *do not move* and *move to the left cell* respectively.

By using all the inputs at our disposal and designing local state-dependent controllers, we can remove the non-determinism imposed by the discretization of the concrete system. This is crucial because abstraction-based control guarantees that the concrete controller successfully solves a control problem for the original system, provided that the abstract controller solves the associated abstract control problem, and it is worth noting that the level of non-determinism within the abstraction directly affects the feasibility of the abstract control problem. This example illustrates that the use of concrete state information can be beneficial to construct a *practical* abstraction satisfying the memoryless concretization property.

Finally, let us compare our work with another relation introduced in [3, Definition 6] as the *strong alternating ϵ-approximate simulation relation* (denoted S-ASR) which is characterized by symbolic concretization property, i.e., the existence of a concrete controller using only abstract state information. In fact, S-ASR can be derived from FRR by relaxing condition (2) in Definition 9 in that of ASR,

while MCR is obtained by relaxing condition (1) in Definition 9. Therefore, in the context of abstraction with overlapping cells, S-ASR has the same drawback as ASR, in that it does not allow the use of the controller architecture described in Figure 8, but requires the controller architecture given in Figure 4.

5 CONCLUSION

We have introduced memoryless concretization relation, which provides a framework that guarantees a simple control architecture, requiring only information about the current state. In addition, this concretization procedure is independent of the type of dynamical systems and specifications under consideration. In particular, if the abstract controller is static, the concrete controller will also be static. We have additionally provided a precise characterization of this relation with a necessary and sufficient condition on the concretization architecture.

We have demonstrated that any alternating simulation relation can be extended to a memoryless concretization relation at the price of introducing additional non-determinism. Furthermore, we prove that ASR and MCR coincide in the particular case of a deterministic quantizer, and thus that ASR benefits from the memoryless concretization property in this specific case.

In addition, we showed that this framework allows the use of overlapping cells (referred to as cover-based abstraction) and piecewise state-dependent controllers, enabling the design of low-level controllers within cells, in combination with high-level abstraction-based controllers. This opens up new possibilities when co-creating the abstraction and the controller, as is done in so-called *lazy abstractions*.

ACKNOWLEDGMENTS

JC is a FRIA Research Fellow. This project has received funding from the H2020-EU.1.1. research and innovation programme(s) – ERC-2016-COG under grant agreement No 725144. RJ is a FNRS honorary Research Associate. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 864017 - L2C.

REFERENCES

[1] Rajeev Alur, Thomas A Henzinger, Orna Kupferman, and Moshe Y Vardi. 1998. Alternating refinement relations. In *CONCUR'98 Concurrency Theory: 9th International Conference Nice, France, September 8–11, 1998 Proceedings 9*. Springer, 163–178.

[2] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. 2017. *Formal methods for discrete-time dynamical systems*. Vol. 89. Springer.

[3] Alessandro Borri, Giordano Pola, and Maria Domenica Di Benedetto. 2018. Design of symbolic controllers for networked control systems. *IEEE Trans. Automat. Control* 64, 3 (2018), 1034–1046.

[4] Julien Calbert, Benoît Legat, Lucas N Egidio, and Raphaël Jungers. 2021. Alternating simulation on hierarchical abstractions. In *IEEE Conference on Decision and Control (CDC)* (Cancun, Mexico). 593–598.

[5] Eric Dallal, Alessandro Colombo, Domitilla Del Vecchio, and Stéphane Lafortune. 2013. Supervisory control for collision avoidance in vehicular networks with imperfect measurements. In *52nd IEEE Conference on Decision and Control*. IEEE, 6298–6303.

[6] Lucas N. Egidio, Thiago Alves Lima, and Raphaël M. Jungers. 2022. State-feedback Abstractions for Optimal Control of Piecewise-affine Systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. 7455–7460. https://doi.org/10.1109/CDC51059.2022.9992495

[7] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete applied mathematics* 42, 2-3 (1993), 177–201.

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

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

[10] Lars Grune and Oliver Junge. 2007. Approximately optimal nonlinear stabilization with preservation of the Lyapunov function property. In *2007 46th IEEE Conference on Decision and Control*. IEEE, 702–707.

[11] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. 2018. Lazy abstraction-based control for safety specifications. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 4902–4907.

[12] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. 2019. Lazy abstraction-based controller synthesis. In *International Symposium on Automated Technology for Verification and Analysis*. Springer, 23–47.

[13] Orna Kupferman and Moshe Y Vardi. 2001. Model checking of safety properties. *Formal methods in system design* 19 (2001), 291–314.

[14] Benoît Legat, Raphaël M Jungers, and Jean Bouchat. 2021. Abstraction-based branch and bound approach to Q-learning for hybrid optimal control. In *Learning for Dynamics and Control*. PMLR, 263–274.

[15] Rupak Majumdar, Necmiye Ozay, and Anne-Kathrin Schmuck. 2020. On abstraction-based controller design with output feedback. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*. 1–11.

[16] Gunther Reissig, Alexander Weber, and Matthias Rungger. 2016. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Trans. Automat. Control* 62, 4 (2016), 1781–1796.

[17] Matthias Rungger and Majid Zamani. 2016. SCOTS: A tool for the synthesis of symbolic controllers. In *Proceedings of the 19th international conference on hybrid systems: Computation and control*. 99–104.

[18] Paulo Tabuada. 2009. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media.

[19] Boyan Yordanov, Jana Tumova, Ivana Cerna, Jiří Barnat, and Calin Belta. 2011. Temporal logic control of discrete-time piecewise affine systems. *IEEE Trans. Automat. Control* 57, 6 (2011), 1491–1504.