2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

# Alternating Simulation on Hierarchical Abstractions

Julien Calbert Benoît Legat Lucas N. Egidio Raphaël Jungers

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9683448

*Abstract*— **Abstraction techniques provide formal guarantees for generic optimal control problems on nonlinear and hybrid systems. Computing an abstraction solving the problem over the whole state-space is computationally demanding in high-dimensional spaces. We circumvent this curse of dimensionality by introducing a hierarchical abstraction approach for solving an optimal control problem for nonlinear systems with three nested partitions. These nested partitions allow the construction of auxiliary systems that characterize simulation relations, which are suitably exploited to provide upper and lower bounds for a branch and bound algorithm to yield an optimal solution for the control problem. An example illustrates the proposed method.**

## I. Introduction

Generalizing classical control techniques based on frequency analysis or convex optimization to hybrid systems is challenging. In the literature, some works achieve this for specific classes of hybrid systems but this is usually limited to systems for which the discrete part is not too complex. However, the increasing interconnection between computers and dynamical systems demands more general frameworks, which should be able to match the current challenges of these cyber-physical systems in a seamless way. On the other hand, abstraction techniques rely on the relation between the system and an automaton to leverage algorithms on graphs or hyper-graphs. While these techniques suffer from the curse of dimensionality, their complexity is less affected by the complexity of the discrete part. The classical abstraction method [1] suffers from having to compute the abstraction on the complete state space. In [2] the authors propose to adapt the size of the abstraction gradually but uniformly over the whole state space. In this work, we propose to co-design the abstraction and the controller guided by the optimal control problem in order to reduce the computed part of the abstraction. To this end, we provide algorithms based on the following three observations.

First, a common approach is to seek for an abstraction that is in alternating bisimulation relation with the original system like in [3]. That is, the abstraction is both an alternating simulation of and alternatingly simulated by the original system. This requires both that the system satisfies some assumptions such as incremental stability and that the abstraction is sufficiently fine-grained, which heavily intensifies the curse of dimensionality. It is shown in [4] that bisimulation can be replaced by *alternating simulation* relations, while still retaining some key properties enabled by bisimulation. In particular, two of these properties are Lyapunov functions and BQ-functions (see Definition 3). While similar concepts had been used in particular frameworks, e.g in [5] where the concept of Lyapunov function was used for a specific case of alternating simulation, the conjugate use of alternating simulation and BQ- and Lyapunov functions allows for leveraging optimal control tools in the framework of symbolic control. As we show in this paper, for any partition of the state-space, one can build an alternating simulation and an alternatingly simulated system, together with a BQ- and Lyapunov function, which in turn provides lower and upper bounds to the optimal cost of an optimal control problem [4, Lemma 13]. While finer abstractions reduce the gap between the two bounds, the techniques developed in this work do not have any requirements on this gap thus allowing the use of coarser abstractions to improve scalability.

Second, while computing a fine enough abstraction over the whole state-space is often impossible due to high computational cost, computing the abstraction in a restricted area, close to the optimal trajectory is much less affected by an increase in the number of dimensions. In practice, this optimal trajectory is unknown, and we propose a methodology based on the so-called A$^*$ algorithm (revisited in Algorithm 1) in order to determine it in an incremental way. The A$^*$ algorithm is commonly used in artificial intelligence and usually dramatically reduces the computational time when a good consistent lower bound on the optimal cost is provided. For discrete-time systems, this lower bound takes the form of a BQ-function. We show in Algorithm 1 how to exploit the BQ-function to compute *lazily* an alternatingly simulated system and its associated Lyapunov function. By *lazy*, we mean that the computation of the transitions for each state of the abstraction is postponed until the moment when it is needed. This is a key difference between search algorithms such as A$^*$ or *minimax with* $\alpha$-$\beta$ *pruning* [6, Section 6.3.2], which explore part of the state space of a 2-player game and classical *value or policy iteration* strategies [6, Section 7.2], which require the computation of all transitions of a Markov Decision Process.

Third, similarly to the crucial importance of adaptive step-size for simulations, using a uniform grid as partition of the state-space is not appropriate. We show in Theorem 3 and Theorem 7 how to decouple the abstraction into distinct parts of the state-space in which the Lyapunov function or BQ-function as well as the abstraction itself can be

JC is a FRIA Research Fellow. 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. RJ is also supported by the Innoviris Foundation and the FNRS (Chist-Era Druid-net).

J. Calbert, B. Legat, L. N. Egidio and R. Jungers are with the ICTEAM, UCLouvain, 4 Av. G. Lemaître, 1348 Louvain-la-Neuve, Belgium. {julien.calbert,benoit.legat,lucas.egidio, raphael.jungers}@uclouvain.be

computed independently and then combined. This allows the computation to be carried out in a distributed manner.

Our main goal is to combine these ideas in a hierarchical approach with three nested levels of partitions in order to solve an optimal control problem for a nonlinear system while mitigating classical problems induced by the curse of dimensionality. We provide a Branch-and-Bound algorithm that is able to exploit the decoupling to avoid computing some part of the abstraction if it is pruned by bounds provided by the Lyapunov and BQ-functions.

An extended version of this work is available in [7].

## II. BQ-FUNCTIONS

In this section we define the first building blocks of our method, starting by a formal definition of the system to be considered.

**Definition 1** (Discrete-time control system)**.** *A* discrete-time control system *is defined as a triple $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$ where $\mathcal{X}$ is the set of states, $\mathcal{U}$ is the set of inputs and $\rightsquigarrow$ is the subset of transitions $(x, u, x')$ such that the system can reach $x' \in \mathcal{X}$ from $x \in \mathcal{X}$ with input $u \in \mathcal{U}$. We denote $(x, u, x') \in \rightsquigarrow$ as $x \underset{u}{\rightsquigarrow} x'$, and the set of $x'$ such that $x \underset{u}{\rightsquigarrow} x'$ as $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x)$. We denote the set of $x'$ such that $x' \underset{u}{\rightsquigarrow} x$ as $\mathrm{Pre}_{\underset{u}{\rightsquigarrow}}(x)$.*

We denote the set of inputs associated with transitions departing from some state $x \in \mathcal{X}$ as $\mathcal{U}(x) = \{\, u \in \mathcal{U} \mid \mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \neq \varnothing \,\}$. Also, we say that a discrete-time control hybrid system is *deterministic* if for every state $x \in \mathcal{X}$ and control input $u \in \mathcal{U}$, $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x)$ is either empty or a singleton. Otherwise, we say that it is *non-deterministic*.

The simulation used in this paper is commonly referred to as an *alternating simulation*, which was introduced in [8] and can be defined as follows.

**Definition 2** (Alternating simulation relation [9, Definition 4.19 and Definition 4.22])**.** *Consider discrete-time control systems $S_1 = (\mathcal{X}_1, \mathcal{U}_1, \rightsquigarrow^1)$ and $S_2 = (\mathcal{X}_2, \mathcal{U}_2, \rightsquigarrow^2)$, as defined in Definition 1. Given a relation $R \subseteq \mathcal{X}_1 \times \mathcal{X}_2$, consider the* extended relation *$R^e$ defined by the set of $(x_1, x_2, u_1, u_2)$ such that for every $x_2' \in \mathrm{Post}_{\underset{u_2}{\overset{2}{\rightsquigarrow}}}(x_2)$, there exists $x_1' \in \mathrm{Post}_{\underset{u_1}{\overset{1}{\rightsquigarrow}}}(x_1)$ such that $(x_1', x_2') \in R$. If for all $(x_1, x_2) \in R$, and for all $u_1 \in \mathcal{U}_1(x_1)$, there exists $u_2 \in \mathcal{U}_2(x_2)$ such that $(x_1, x_2, u_1, u_2) \in R^e$ then $R$ is an* alternating simulation relation*, $R^e$ is its associated* extended alternating simulation relation *and $S_2$ is an* alternating simulation *of $S_1$.*

We denote the empty tuple as $\emptyset$, the $l$-tuple $(u_i)_{i=1}^{l}$ as $\mathbf{u}_l$ and the concatenation of tuples $\mathbf{u}_{l_1}, \mathbf{u}'_{l_2}$ as the $(l_1+l_2)$-tuple $(\mathbf{u}_{l_1}; \mathbf{u}'_{l_2})$. A *cost function* is a given function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \cup \{\infty\}$, a *value function* is a function $V : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ and a *Q-function* is a function $Q : \mathcal{X} \times \mathcal{U}^l \to \mathbb{R} \cup \{\infty\}$. Given a *Q-function* $Q : \mathcal{X} \times \mathcal{U}^l \to \mathbb{R} \cup \{\infty\}$ for some $l \in \mathbb{N}$ with some cost function $c$, we recursively define the value of $Q(x, \mathbf{u}_{l'})$ for $l' > l$ with the following identity for $k = l+1, \ldots, l'$:

$$Q(x, \mathbf{u}_k) = c(x, u_1) + \max_{x' \in \mathrm{Post}_{\underset{u_1}{\rightsquigarrow}}(x)} Q(x', (u_i)_{i=2}^{k}). \quad (1)$$

Given a cost function $c$ and a value function $V$, $\mathcal{T}_c^Q V$ denotes the Q-function with cost function $c$ such that $\mathcal{T}_c^Q V(x, \emptyset) = V(x)$.

The *Bellman operator* $\mathcal{T}_c$ is defined as[1]:

$$\mathcal{T}_c V(x) = \min_{u \in \mathcal{U}} \mathcal{T}_c^Q V(x, u) \quad (2)$$
$$\mathcal{T}_c Q(x, \mathbf{u}_l) = \min_{u' \in \mathcal{U}} Q(x, (\mathbf{u}_l; u')).$$

The value function $V$ solving the Bellman equation $\mathcal{T}_c V(x) = V(x)$, commonly referred to as the Bellman value function, is the Grail of optimal control but its computation is challenging. We show in this paper that a function satisfying an inequality instead, e.g., $\mathcal{T}_c V(x) \leq V(x)$, still provides most of the useful properties of the Bellman functions and the computation of such functions is more reasonable. As the inequality $\mathcal{T}_c V(x) \leq V(x)$ corresponds to the Lyapunov inequality, we refer to such functions as Lyapunov functions and detail their properties and computation in Section III. In the case of the inequality $\mathcal{T}_c Q(x, \emptyset) \geq Q(x, \emptyset)$, we use the name *Bellman-like Q-function* (BQ-function for short).

**Definition 3** (BQ-function [4, Definition 4])**.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$. A function $Q : \mathcal{X} \times \mathcal{U}^k \to \mathbb{R}$ is a* BQ-function *of $S$ with cost function $c$ if $Q(x, \mathbf{u}_k) \leq Q(x, (\mathbf{u}_k; \mathbf{u}'_l))$ for all $x \in \mathcal{X}, \mathbf{u}_k \in \mathcal{U}^k$ and $\mathbf{u}'_l \in \mathcal{U}^l$, where $Q(x, (\mathbf{u}_k; \mathbf{u}'_l))$ is defined from Eq.* (1)*.*

As described in the next theorem, knowing an alternating simulation for a discrete-time control system allows the derivation of BQ-functions under given conditions.

**Theorem 1** ([4, Theorem 6])**.** *Consider discrete-time control systems $S_1 = (\mathcal{X}_1, \mathcal{U}_1, \rightsquigarrow^1)$, $S_2 = (\mathcal{X}_2, \mathcal{U}_2, \rightsquigarrow^2)$, defined in Definition 1, and an alternating simulation relation $R$ such that for each $x_1 \in \mathcal{X}_1$, there is exactly one $x_2 \in \mathcal{X}_2$ such that $(x_1, x_2) \in R$, which we denote by $R(x_1)$. Given a cost function $c_1$ for $S_1$, consider an associated cost function satisfying, for all $x_2 \in \mathcal{X}, u_2 \in \mathcal{U}$,*

$$c_2(x_2, u_2) \leq \min_{(x_1, x_2, u_1, u_2) \in R^e} c_1(x_1, u_1). \quad (3)$$

*If $V_2(x)$ is a Bellman-like value function for $S_2$ with cost function $c_2$, then $V_1(x_1) = V_2(R(x_1))$ is a Bellman-like value function for $S_1$ with cost function $c_1$.*

The following proposition allows us to combine different BQ-functions, e.g., one that is known by the control engineer and one that is determined with an abstraction.

**Proposition 1.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, two BQ-functions $Q_1, Q_2 : \mathcal{X} \times \mathcal{U}^0 \to \mathbb{R}$ of $S$ with cost function $c$ and the Q-function $Q : \mathcal{X} \times \mathcal{U}^0 \to \mathbb{R}$ defined by $Q(x, \emptyset) = \max(Q_1(x, \emptyset), Q_2(x, \emptyset))$. Then $Q$ is a BQ-function of $S$ with cost function $c$.*

[1]Note that we have $\mathcal{T}_c V(x) = \mathcal{T}_c \mathcal{T}_c^Q V(x, \emptyset)$.

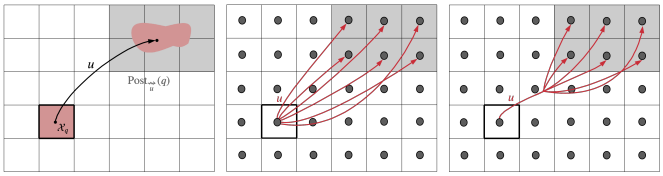

Fig. 1. Left: Illustration of the set of $q'$ such that $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(\mathcal{X}_q) \cap \mathcal{X}_{q'} \neq \varnothing$. Middle: Corresponding transitions created in Theorem 2. Right: Corresponding transition created in Theorem 5.

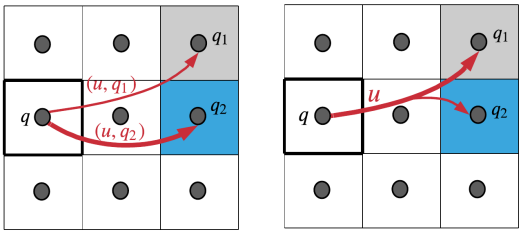


Fig. 2. Illustrations of transitions defined for a given state $q \in \mathcal{Q}$ and an input $u \in \mathcal{U}$ for the abstraction defined in Theorem 2 (left) and in Theorem 5 (right). In the case on the left, one can choose the destination (optimistic) while in the case on the right, one cannot (pessimistic).

Given a partition of the state-space, the following theorem, inspired by discussions in [4, Section 4], allows us to compute a deterministic alternating simulation for a discrete-time system.

**Theorem 2.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, as defined in Definition 1, a set of discrete states $\mathcal{Q}$ and a partition $(\mathcal{X}_q)_{q \in \mathcal{Q}}$ of $\mathcal{X}$. Let $S' = (\mathcal{Q}, \mathcal{U} \times \mathcal{Q}, \rightsquigarrow')$ be the discrete-time control system such that $q \underset{u,q'}{\rightsquigarrow}' q'$ if and only if $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(\mathcal{X}_q) \cap \mathcal{X}_{q'} \neq \varnothing$. Then $S'$ is an alternating simulation of $S$.*

To summarize the implications of Theorem 1 and Theorem 2, if $\mathcal{Q}$ is a finite set then the abstraction defined in Theorem 2 with the cost given by Theorem 1 is a weighted directed graph as illustrated by Figs. 1 and 2. Thus, a BQ-function for the abstraction can be computed using a shortest path algorithm such as the Dijkstra or Bellman-Ford algorithm. The corresponding BQ-function for the original system given by Theorem 1 is a piecewise constant function.

**Theorem 3.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, as defined in Definition 1, and a subset $\mathcal{Z} \subseteq \mathcal{X}$.*

*Let $\hat{S} = (\mathcal{X}, \mathcal{U} \cup (\mathcal{U} \times \mathcal{Z}), \widehat{\rightsquigarrow})$ be the discrete-time control system such that for all $x, x' \in \mathcal{X}, u \in \mathcal{U}$, $x \underset{u}{\widehat{\rightsquigarrow}} x'$ if and only if $x \underset{u}{\rightsquigarrow} x'$ and $x' \notin \mathcal{Z}$ and for all $x, x', x'' \in \mathcal{X}, u \in \mathcal{U}$, $x \underset{u,x'}{\widehat{\rightsquigarrow}} x''$ if and only if $x' = x'' \in \mathcal{Z}$ and $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \cap \mathcal{Z} \neq \varnothing$. Then $\hat{S}$ is an alternating simulation of $S$.*

*Let $\tilde{S} = (\mathcal{X} \cup \{t\}, \mathcal{U} \cup \mathcal{Z}, \widetilde{\rightsquigarrow})$ be the discrete-time control system such that, for all $x, x' \in \mathcal{X}, u \in \mathcal{U}$, we have: $x \underset{u}{\widetilde{\rightsquigarrow}} x'$ if and only if $x' \in \mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \setminus \mathcal{Z}$; $x \underset{u}{\widetilde{\rightsquigarrow}} t$ if and only if $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \cap \mathcal{Z} \neq \varnothing$; and $t \underset{x'}{\widetilde{\rightsquigarrow}} x'$ if and only if $x' \in \mathcal{Z}$. If $\tilde{Q} : \mathcal{X} \times \mathcal{U}^0 \to \mathbb{R}$ is a BQ-function for $\tilde{S}$ with cost function $\tilde{c}(x, u) = c(x, u)$ and $\tilde{c}(t, x') = 0$ then the Q-function $\hat{Q} : \mathcal{X} \times \mathcal{U}^0 \to \mathbb{R}$ defined by $\hat{Q}(x, \emptyset) = \tilde{Q}(x, \emptyset)$ for all $x \in \mathcal{X}$ is a BQ-function for $\hat{S}$ with cost function $\hat{c}(x, u) = c(x, u)$ and $\hat{c}(x, (u, x')) = c(x, u)$.*

## III. Lyapunov functions

Analogously to the definition of BQ-function and its implications given in the previous section, at this point we define the Lyapunov function for a given discrete-time control system and present associated results.

**Definition 4** (Lyapunov function)**.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, a set $\mathcal{X}_f \subseteq \mathcal{X}$ and a cost function $c$. A value function $L$ is a* Lyapunov function *with cost function $c$ for $S$ in $\mathcal{X}_f$ if, for all $x \in \mathcal{X} \setminus \mathcal{X}_f$, $L(x) = \infty$, and for all $x \in \mathcal{X}_f$, $L(x)$ is finite and $L(x) \geq \mathcal{T}_c L(x)$ where $\mathcal{T}_c$ is defined in Eq. (2).*

**Theorem 4** ([4, Theorem 8])**.** *Consider discrete-time control systems $S_1 = (\mathcal{X}_1, \mathcal{U}_1, \rightsquigarrow^1)$, $S_2 = (\mathcal{X}_2, \mathcal{U}_2, \rightsquigarrow^2)$, as defined in Definition 1, and an alternating simulation relation $R$ such that for each $x_2 \in \mathcal{X}_2$, there is exactly one $x_1 \in \mathcal{X}_1$ such that $(x_1, x_2) \in R$, which we denote by $R(x_2)$. Given a cost function $c_2$ for $S_2$, consider an associated cost function satisfying, for all $x_1 \in \mathcal{X}, u_1 \in \mathcal{U}$,*

$$c_1(x_1, u_1) \geq \max_{(x_1, x_2, u_1, u_2) \in R^e} c_2(x_2, u_2). \tag{4}$$

*If $L_1(x)$ is a Lyapunov function for $S_1$ with cost function $c_1$, then $L_2(x_2) = L_1(R(x_2))$ is a Lyapunov function for $S_2$ with cost function $c_2$.*

The following proposition allows us to combine different Lyapunov functions, e.g., one that is known by the control engineer and one that is determined with an abstraction.

**Proposition 2.** *Consider two Lyapunov functions $L_1, L_2 : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ and the function $L : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ defined by $L(x) = \min(L_1(x), L_2(x))$. Then $L$ is a Lyapunov function.*

Given a partition of the state-space, the following theorem allows us to compute a (non-deterministic / hypergraph) alternating simulation for a discrete-time system.

**Theorem 5** ([10, Theorem 4.1])**.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, as defined in Definition 1, a set $\mathcal{Q}$ and a partition $(\mathcal{X}_q)_{q \in \mathcal{Q}}$ of $\mathcal{X}$. Let $S' = (\mathcal{Q}, \mathcal{U}, \rightsquigarrow')$ be the discrete-time control system such that $q \underset{u}{\rightsquigarrow}' q'$ if and only if $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(\mathcal{X}_q) \cap \mathcal{X}_{q'} \neq \varnothing$. Then $S$ is an alternating simulation of $S'$.*

If $\mathcal{Q}$ is a finite set, the abstraction defined in Theorem 5 with the cost given by Theorem 4 is a weighted directed *forward hypergraph* $G$. That is, each transition corresponds to a *forward hyperarc* which is a hyperarc with one tail and multiple heads as illustrated by Figs. 1 and 2; see [11] for a introduction to hypergraphs. The *transpose* of the hypergraph $G^\top$ (which corresponds to the hypergraph where the heads and tails are reversed) is a *backward hypergraph*. That is, each transition corresponds to a *backward hyperarc* which is a hyperarc with one head and multiple tails. A Lyapunov

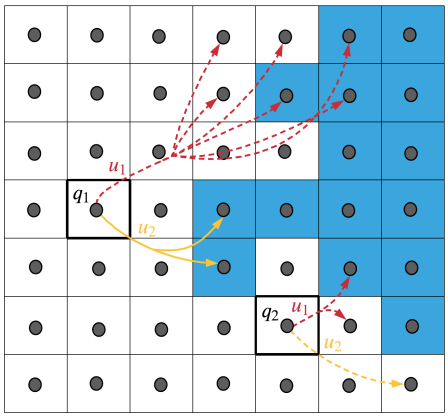


Fig. 3. Illustration of the generalization of the Dijkstra algorithm to forward hypergraphs detailed in Algorithm 1. We represent in blue the set $\mathcal{M} = \{q \in \mathcal{X} \mid \ell^\star[q] < \infty\}$, which contains the cells for which we already know the optimal control law to lead them to the target set. A cell can be added in $\mathcal{M}$ when for at least one input, all destinations are in $\mathcal{M}$. This is the case for $q_1$ and not yet for $q_2$. In Algorithm 1, the A* procedure smartly selects the next cell $q$ and input $u$ to expand, i.e. to compute $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(q)$.

function for the abstraction can be obtained by finding the minimal weight forward (resp. backward) hyperpath to (resp. from) any node of $\mathcal{X}_f$ in $G$ (resp. $G^\top$) where the weight of a hyperpath corresponds to the *distance in G (resp. $G^\top$)*. While shortest hyperpath problems on hypergraphs are NP-hard in general [12], this particular case can be solved in polynomial time by a generalization of the Dijkstra or Bellman-Ford algorithm [11]. The corresponding Lyapunov function for the original system given by Theorem 4 is a piecewise constant function.

The following definition formalizes the *time-reverse* of a discrete-time control system.

**Definition 5.** *Given a discrete-time control systems $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, its* controlled reverse *is the discrete-time control system defined by $\overleftarrow{S} = (\mathcal{X}, \mathcal{U} \times \mathcal{X}, \overleftarrow{\rightsquigarrow})$ where $x \underset{u,x'}{\overleftarrow{\rightsquigarrow}} x''$ if and only if $x' = x''$ and $x' \underset{u}{\rightsquigarrow} x$. Given a cost function $c$ for system $S$, its corresponding* controlled reverse *cost function $\overleftarrow{c}$ is defined as $\overleftarrow{c}\,(x,(u,x')) = c(x',u)$.*

Algorithm 1 lazily builds an abstraction and computes a Lyapunov value function for it. The tighter the BQ-function is, the smaller part of the abstraction we will have to build. Theorem 4 then allows us to deduce a Lyapunov function for the original system. The following theorem is based on a generalization of the $A^*$ algorithm for hypergraphs [13]. Algorithm 1 computes a controller for a subset of the state space given by $\mathcal{N} = \{q \in \mathcal{X} \mid \ell[q] < \infty, \ell^\star[q] = \infty\}$. The controller is optimal for the subset $\mathcal{M} = \{q \in \mathcal{X} \mid \ell^\star[q] < \infty\}$ that contains at the end of the execution the initial set $\mathcal{X}_s$.

**Theorem 6.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, a cost function $c$, a source set $\mathcal{X}_s \subseteq \mathcal{X}$, a target set $\mathcal{X}_t \subseteq \mathcal{X}$, a BQ-function $Q$ for the controlled reverse of $S$. If the state-space $\mathcal{X}$ is finite then Algorithm 1 terminates in finite time and returns a Lyapunov value function.*

**Algorithm 1:** Algorithm for the computation of a Lyapunov function $\ell^\star$ based on a generalization of the $A^*$ algorithm for hypergraphs. The abstraction is computed lazily leveraging the ability of the $A^*$ algorithm to explore a reduced part of the state-space while still guaranteeing optimality. See Fig. 3 and [7, Figure 4] for an illustration of the algorithm.

**Data:** A discrete-time control system $S$, a cost function $c$, a source set $\mathcal{X}_s \subseteq \mathcal{X}$, a target set $\mathcal{X}_t \subseteq \mathcal{X}$, a BQ-function $Q$ for the controlled reverse of $S$ with cost function $\overleftarrow{c}$.

$\ell^\star[q] \leftarrow \infty, \ \forall q \in \mathcal{X}$;
$\ell[q] \leftarrow \infty, \ \forall q \in \mathcal{X} \setminus \mathcal{X}_t$;
$\ell[q] \leftarrow Q(q, \emptyset), \ \forall q \in \mathcal{X}_t$;
$\mathcal{N} \leftarrow \mathcal{X}_t$;
$\mathcal{M} \leftarrow \varnothing$;
**while** $\mathcal{N} \neq \varnothing$ *and* $\exists q \in \mathcal{X}_s$ *such that* $\ell^\star[q] = \infty$ **do**
  $q \leftarrow \arg\min\{\ell[q] \mid q \in \mathcal{N}\}$;
  $\ell^\star[q] \leftarrow \ell[q] - Q(q, \emptyset)$;
  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{q\}$;
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{q\}$;
  **for** $u \in \mathcal{U}$ **do**
    **for** $q' \in \mathrm{Pre}_{\underset{u}{\rightsquigarrow}}(q)$ **do**
      **if** $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(q')$ *not computed yet* **then**
        Compute $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(q')$;
      **end**
      **if** $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(q') \subseteq \mathcal{M}$ **then**
        $\ell[q'] \leftarrow \min(\ell[q'], Q(q', \emptyset) + c(q', u) + \max_{q'' \in \mathrm{Post}_{\underset{u}{\rightsquigarrow}}(q')} \ell^\star[q''])$;
        $\mathcal{N} \leftarrow \mathcal{N} \cup \{q'\}$;
      **end**
    **end**
  **end**
**end**
**return** $\ell^\star$;

**Theorem 7.** *Consider a discrete-time control system $S = (\mathcal{X}, \mathcal{U}, \rightsquigarrow)$, as defined in Definition 1, and a subset $\mathcal{Z} \subseteq \mathcal{X}$.*

*Let $\hat{S} = (\mathcal{X}, \mathcal{U}, \widehat{\rightsquigarrow})$ be the discrete-time control system such that $x \underset{u}{\widehat{\rightsquigarrow}} x'$ if and only if $x \underset{u}{\rightsquigarrow} x'$ or $x' \in \mathcal{Z}$ and $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \cap \mathcal{Z} \neq \varnothing$. Then $S$ is an alternating simulation of $\hat{S}$.*

*Let $\tilde{S} = (\mathcal{X} \cup \{t\}, \mathcal{U} \cup \{\emptyset\}, \widetilde{\rightsquigarrow})$ be the discrete-time control system such that for all $x, x' \in \mathcal{X}, u \in \mathcal{U}$, we have $x \underset{u}{\widetilde{\rightsquigarrow}} x'$, if and only if $x' \in \mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x)$ and $x' \notin \mathcal{Z}$, $x \underset{u}{\widetilde{\rightsquigarrow}} t$ if and only if $\mathrm{Post}_{\underset{u}{\rightsquigarrow}}(x) \cap \mathcal{Z} \neq \varnothing$, and $t \underset{\emptyset}{\widetilde{\rightsquigarrow}} x'$ if and only if $x' \in \mathcal{Z}$. If $\tilde{L} : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is a Lyapunov function for $\tilde{S}$ with cost function $\tilde{c}(x,u) = c(x,u)$ and $\tilde{c}(t,\emptyset) = 0$ then the value function $\hat{L} : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ defined by $\hat{L}(x) = \tilde{L}(x)$ is a Lyapunov function for $\hat{S}$ with cost function $\hat{c}(x,u) = c(x,u)$.*

## IV. Hierarchical abstraction control

We will now combine these lower and upper bounds in a hierarchical approach with three nested levels of partitions. Consider a discrete-time system $S_0 = (\mathcal{X}_0, \mathcal{U}_0, \rightsquigarrow_0)$ and a partition $({}_2\mathcal{X}_q)_{q\in\mathcal{Q}_2}$ of $\mathcal{X}$. For each $q_2 \in \mathcal{Q}_2$, consider the partition $({}_1\mathcal{X}_q^{q_2})_{q\in\mathcal{Q}_1^{q_2}}$ of ${}_2\mathcal{X}_{q_2}$ with $\mathcal{Q}_1 = \bigcup_{q_2\in\mathcal{Q}_2} \mathcal{Q}_1^{q_2}$. Also, for each $q_2 \in \mathcal{Q}_2, q_1 \in \mathcal{Q}_1^{q_2}$, consider the partition $({}_{-1}\mathcal{X}_q)_{q\in\mathcal{Q}_{-1}^{q_1,q_2}}$ of ${}_1\mathcal{X}_{q_1}^{q_2}$ with $\mathcal{Q}_{-1}^{q_2} = \bigcup_{q_1\in\mathcal{Q}_1^{q_2}} \mathcal{Q}_{-1}^{q_1,q_2}$ and $\mathcal{Q}_{-1} = \bigcup_{q_2\in\mathcal{Q}_2} \mathcal{Q}_{-1}^{q_2}$.

Let $\overleftarrow{S}_0$ with cost $\overleftarrow{c}$ be the controlled reverse of $S_0$ with cost $c$. Consider the alternating simulation $S_2 = (\mathcal{Q}_2, \mathcal{U}\times\mathcal{X}, \rightsquigarrow_2)$ (resp. $\overleftarrow{S}_{-1} = (\mathcal{Q}_{-1}, \mathcal{U}\times\mathcal{X}, \overleftarrow{\rightsquigarrow}_{-1})$, $\overleftarrow{S}_1 = (\mathcal{Q}_1, \mathcal{U}\times\mathcal{X}, \overleftarrow{\rightsquigarrow}_1)$) of $S_0$ (resp. $\overleftarrow{S}_0$, $\overleftarrow{S}_{-1}$) provided by Theorem 2 with the partition $({}_2\mathcal{X}_q)_{q\in\mathcal{Q}_2}$ (resp. $({}_{-1}\mathcal{X}_q)_{q\in\mathcal{Q}_{-1}}$, $({}_1\mathcal{X}_q)_{q\in\mathcal{Q}_1}$). Moreover, consider the system $S_{-1} = (\mathcal{Q}_{-1}, \mathcal{U}, \rightsquigarrow_{-1})$ for which $S_0$ is an alternating simulation provided by Theorem 5 with the partition $({}_{-1}\mathcal{X}_q)_{q\in\mathcal{Q}_{-1}}$. The goal of the coarser abstraction $S_2$ is to decouple the problem into local objectives in order to reduce the part of the state space where the Lyapunov and the BQ-function are computed. These local objectives are defined by the following sets.

For any $q, q' \in \mathcal{Q}_2$, we define the sets

$$\mathcal{I}_{q,q'}^{1} = \{\, t \in \mathcal{Q}_1^{q'} \mid \exists s \in \mathcal{Q}_1^{q}, u \in \mathcal{U} \text{ s.t. } t \underset{u,s}{\overleftarrow{\rightsquigarrow}}_1 s \,\},$$

$$\mathcal{I}_{q,q'}^{-1} = \{\, t \in \mathcal{Q}_{-1}^{q'} \mid \exists s \in \mathcal{Q}_{-1}^{q}, u \in \mathcal{U} \text{ s.t. } s \underset{u}{\rightsquigarrow}_{-1} t \,\},$$

We then define $\hat{S}_{-1} = (\mathcal{Q}_{-1}, \mathcal{U}, \widehat{\rightsquigarrow}_{-1})$ and $\tilde{S}_{-1} = (\mathcal{Q}_{-1}, \mathcal{U}, \widetilde{\rightsquigarrow}_{-1})$ as the systems $\hat{S}$ and $\tilde{S}$ obtained by successively applying Theorem 7 for $S_{-1}$ with $\mathcal{Z} = \mathcal{I}_{q,q'}^{-1}$ for each $q, q' \in \mathcal{Q}_2$. Analogously, we define $\hat{S}_1 = (\mathcal{Q}_1, \mathcal{U}, \widehat{\rightsquigarrow}_1)$ and $\tilde{S}_1 = (\mathcal{Q}_1, \mathcal{U}, \widetilde{\rightsquigarrow}_1)$ by applying Theorem 3 for $\overleftarrow{S}_1$ with $\mathcal{Z} = \mathcal{I}_{q,q'}^{1}$ for each $q, q' \in \mathcal{Q}_2$.

By Theorem 2, Theorem 3, Theorem 5 and Theorem 7, we have the following where $A \preceq B$ means that $B$ is an alternating simulation of $A$:

$$\begin{aligned} \hat{S}_{-1} \preceq S_{-1} \preceq S_0 \preceq S_2 \\ \overleftarrow{S}_0 \preceq \overleftarrow{S}_{-1} \preceq \overleftarrow{S}_1 \preceq \hat{S}_1 \end{aligned}$$

Therefore, by Theorem 3 and Theorem 1, we can deduce a BQ-function for $\overleftarrow{S}_0$ with cost $\overleftarrow{c}$ given a BQ-function for $\tilde{S}_1$ with cost $\overleftarrow{c}$. Similarly, by Theorem 7 and Theorem 4, we can deduce a Lyapunov function for $S_0$ with cost $c$ given a Lyapunov function for $\tilde{S}_{-1}$ with cost $c$.

The computation of the BQ-function (resp. Lyapunov function) for $\tilde{S}_i$ for $i = 1$ (resp. $i = -1$) can be decoupled into the computation of such function for the subsystem corresponding to a specific $q \in \mathcal{Q}_2$ defined by the subset of states

$$\{\, t_{qq'}^{i} \mid q' \in \mathcal{Q}_2 \,\} \cup \{\, t_{q'q}^{i} \mid q' \in \mathcal{Q}_2 \,\} \cup \mathcal{Q}_i^{q}.$$

As $\overleftarrow{S}_1$ is an alternating simulation of $\overleftarrow{S}_{-1}$, a BQ-function for $\overleftarrow{S}_{-1}$ can be deduced from a BQ-function of $\overleftarrow{S}_1$ using Theorem 2. Moreover, as $\overleftarrow{S}_{-1}$ is the controlled reverse of $S_{-1}$, this provides the BQ-function required for applying Algorithm 1 on $S_{-1}$. Note that as these BQ-function are already computed by Algorithm 2, they can be memoized for Algorithm 3 to avoid recomputing.

**Algorithm 2:** Lower bound algorithm that can be used as $\alpha$ function for Algorithm 4.

**Data:** A discrete-time control system $S_0$, sequence of states $(q_i)_{i=0}^{k}$, initial state $x_0 \in {}_2\mathcal{X}_{q_0}$, alternating simulation $\overleftarrow{S}_1$ with alternating simulation relation $R_1$, alternating simulation $S_2$ and corresponding BQ-function $Q_2$.

```
for j ← 0 to k do
    if j = 0 then
        source_j ← {R_1(x_0)};
    else
        source_j ← {t^1_{q_{j-1} q_j}};
    end
end
for j ← 0 to k − 1 do
    Q̃_j ← Dijkstra for subsystem of S̃_1 at q_j with
      target source_j;
end
return Q_2(q_k) + Σ_{j=1}^{k} Q̃_{j−1}(source_j);
```

**Remark 1.** *The time and space needed for the computation of the BQ-function and Lyapunov function in Algorithm 2 and Algorithm 3 can be drastically reduced by memoization techniques. The abstraction only depends on the state $q \in \mathcal{Q}_2$ so it only needs to be computed at most once. For the BQ-function, the value of the function only depends on the target, not on the whole sequence $(q_i)_{i=0}^{k}$ which allows different sequences to reuse the same computation. For the Lyapunov function, the abstraction is computed lazily and is stopped once the source is completely covered. Therefore, when computing the Lyapunov function for the same target and a different source, the computation may need to be continued where it was left off until this new source is covered, reusing the part of the function and abstraction that was already computed.*

**Theorem 8** ([4, Theorem 15])**.** *Algorithm 4 returns an optimal solution in finite time for the system $S_{-1}$.*

## V. Conclusion

In this paper, we formulated modular foundations for the construction of hierarchical abstractions. We formalized the connections between alternating simulations and the Bellman operator. This provides a modular methodology to transfer bounds on the optimal cost from abstractions from different levels of nested partitions. We developed a branch and bound algorithm that leverages the bounds gathered from the the constructed abstractions. Finally, numerical experiments illustrate the practical significance of these results.

**Algorithm 3:** Upper bound algorithm that can be used as $\beta$ function for Algorithm 4.

**Data:** A deterministic discrete-time control system $S_0$ with cost $c_0$, sequence of states $(q_i)_{i=0}^k$ with $q_k = q_t$, initial state $x_0 \in {}_2\mathcal{X}_{q_0}$, a target set $\mathcal{X}_t \subseteq {}_2\mathcal{X}_{q_t}$ and an alternatingly simulated system $\overleftarrow{S}_{-1}$ with alternating simulation relation $R_{-1}$.

$x \leftarrow x_0$; $l \leftarrow 0$; $\mathsf{cost} \leftarrow 0$;
**for** $j \leftarrow 0$ **to** $k$ **do**
  **if** $j = 0$ **then**
    **for** $i \leftarrow -1$ **to** $1$ **by** $2$ **do**
      $\mathsf{source}_i \leftarrow \{R_i(x_0)\}$;
    **end**
  **else**
    **for** $i \leftarrow -1$ **to** $1$ **by** $2$ **do**
      $\mathsf{source}_i \leftarrow \{t^i_{q_{j-1}q_j}\}$;
    **end**
  **end**
  **if** $q_k = q_t$ **then**
    $\mathsf{target} \leftarrow \{\, q \in \mathcal{Q}^{q_t}_{-1} \mid {}_{-1}\mathcal{X}_q \subseteq \mathcal{X}_t \,\}$;
  **else**
    $\mathsf{target} \leftarrow \{t^{-1}_{q_j q_{j+1}}\}$;
  **end**
  $Q_1 \leftarrow$ Dijkstra for subsystem of $\tilde{S}_1$ at $q_j$ with target $\mathsf{source}_1$;
  $Q_{-1} \leftarrow$ Theorem 2 with $Q_1$;
  $L \leftarrow$ Algorithm 1 for subsystem of $\tilde{S}_{-1}$ at $q_j$ with target $\mathsf{target}$, source $\mathsf{source}_{-1}$ and reverse BQ-function $Q_{-1}$;
  **while** $R_{-1}(x) \notin \mathsf{target}$ **do**
    **if** $L(R_{-1}(x)) = \infty$ **then**
      **return** $\emptyset, \infty$
    **end**
    $l \leftarrow l + 1$;
    $u_l \leftarrow \arg\min_u c_0(x,u) + \max_{x' \in \mathrm{Post}_{\rightsquigarrow_u}(x)} L(R_{-1}(x'))$;
    $x \leftarrow$ unique element in $\mathrm{Post}_{\rightsquigarrow_{u_l}}(x)$;
    $\mathsf{cost} \leftarrow \mathsf{cost} + c_0(x, u_l)$;
  **end**
**end**
**return** $\mathbf{u}_l, \mathsf{cost}$;

**Algorithm 4:** Branch and bound algorithm. The set $\mathcal{N}$ represents the set of nodes of the search tree for which subtrees still need to be explored. The heuristic function determines which node is considered next. Note the difference between the notation $\emptyset$ used to denote the empty tuple of discrete control inputs and the notation $\varnothing$ used to denote the empty set.

**Data:** A discrete-time control system $S_0$, initial state $x_0 \in {}_2\mathcal{X}_{q_0}$, a target set $\mathcal{X}_t \subseteq {}_2\mathcal{X}_{q_t}$, a heuristic function $h$, a lower bound function $\alpha$, an upper bound function $\beta$ and a maximum number of steps in $S_2$ of $\tilde{l} \in \mathbb{N}$.

$\bar{\beta} \leftarrow \infty$;
$\mathcal{N} \leftarrow \{\emptyset\}$;
**while** $\mathcal{N} \neq \varnothing$ **do**
  $(q_i)_{i=0}^l \leftarrow h(\mathcal{N})$;
  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{(q_i)_{i=0}^l\}$;
  **if** $\alpha((q_i)_{i=0}^l) \leq \bar{\beta}$ **and** $l < \tilde{l}$ **then**
    **for** $q' \in \mathcal{Q}_2$ **do**
      $\hat{\mathbf{u}}_{\hat{l}}, \hat{\beta} \leftarrow \beta(x_0, ((q_i)_{i=0}^l; q'))$;
      **if** $\hat{\beta} < \bar{\beta}$ **then**
        $\bar{\mathbf{u}}_{\bar{l}}, \bar{\beta} \leftarrow \hat{\mathbf{u}}_{\hat{l}}, \hat{\beta}$;
      **end**
      $\mathcal{N} \leftarrow \mathcal{N} \cup \{((q_i)_{i=0}^l; q')\}$;
    **end**
  **end**
**end**
**return** $\bar{\mathbf{u}}_{\bar{l}}, \bar{\beta}$;

## References


[1] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1781–1796, 2016.

[2] K. Hsu, R. Majumdar, K. Mallik, and A.-K. Schmuck, "Multi-layered abstraction-based controller synthesis for continuous-time systems," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, ser. HSCC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 120–129.

[3] A. Girard, "Approximately bisimilar abstractions of incrementally stable finite or infinite dimensional systems," in *53rd IEEE conference on decision and control*. IEEE, 2014, pp. 824–829.

[4] B. Legat, J. Bouchat, and R. M. Jungers, "Abstraction-based branch and bound approach to q-learning for hybrid optimal control," in *Proceedings of the 3rd Annual Learning for Dynamics & Control Conference*, 2021.

[5] E. Aydin Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for linear systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1163–1176, 2014.

[6] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[7] J. Calbert, B. Legat, L. N. Egidio, and R. M. Jungers, "Alternating Simulation on Hierarchical Abstractions," *ArXiv e-prints*, 2021.

[8] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi, "Alternating refinement relations," in *CONCUR'98 Concurrency Theory*. Springer Berlin Heidelberg, 1998, pp. 163–178.

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

[10] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1804–1809, 2012.

[11] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete applied mathematics*, vol. 42, no. 2-3, pp. 177–201, 1993.

[12] G. F. Italiano and U. Nanni, "Online maintenance of minimal directed hypergraphs," Department of Computer Science, Columbia University Series, Tech. Rep. CUCS-435-89, 1989.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.