Multi-Layered Abstraction-Based Controller Synthesis for Continuous-Time Systems

Kyle Hsu*
University of Toronto, Canada

Rupak Majumdar*
MPI-SWS, Kaiserslautern, Germany

Kaushik Mallik*
MPI-SWS, Kaiserslautern, Germany

Anne-Kathrin Schmuck*
MPI-SWS, Kaiserslautern, Germany

ABSTRACT

We present *multi-layered abstraction-based controller synthesis*, which extends standard abstraction-based controller synthesis (ABCS) algorithms for continuous-time control systems by simultaneously maintaining several "layers" of abstract systems with decreasing precision. The resulting abstract multi-layered controller uses the coarsest abstraction whenever this is feasible, and dynamically adjusts the precision—by moving to a more precise abstraction and back to a coarser abstraction—based on the structure of the given control problem. Abstract multi-layered controllers can be refined to controllers with non-uniform resolution using feedback refinement relations established between each abstract layer and the concrete system, resulting in a sound ABCS method. We provide multi-layered controller synthesis algorithms for reachability, safety, and generalized Büchi specifications; our approach can be generalized to any ω-regular objective. Our algorithms are complete relative to single-layered synthesis on the finest layer. We empirically demonstrate that multi-layered synthesis can outperform standard (single-layer) ABCS algorithms on a number of examples, despite the additional cost of constructing multiple abstract systems.

ACM Reference Format:
Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. 2018. Multi-Layered Abstraction-Based Controller Synthesis for Continuous-Time Systems. In *HSCC '18: 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), April 11–13, 2018, Porto, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178126.3178143

1 INTRODUCTION

Abstraction-based controller synthesis (ABCS) [18] is a general technique for automatic synthesis of controllers for non-linear dynamical systems such that the closed-loop system satisfies a temporal logic specification. In this approach, a time-sampled version of the continuous dynamics of the open-loop system (the *concrete system*) is abstracted by a symbolic finite state model (the *abstract system*). Then, automata-theoretic algorithms from finite-state reactive synthesis [7, 14] are used to synthesize a discrete controller on the abstract system for a given temporal logic specification. If there is a *feedback refinement relation* (FRR) [16] between the concrete and abstract systems, the abstract controller can be easily refined to a controller for the concrete system such that the resulting closed loop satisfies the specification.

In practice, the abstract system is computed by first fixing a parameter τ for the sample time and a parameter η for the state and input spaces, and then representing the abstract state space as a set of hypercubes, each of diameter η. The hypercubes partition the continuous concrete state space. The abstract transition relation adds a transition between two hypercubes iff there exists some state in the first hypercube which can reach some state of the second by following the original dynamics for time τ. Clearly, this is an over-approximation of the original dynamics.

The success of ABCS depends on the choice of η and τ. Intuitively, increasing η (and τ)[1] results in fewer hypercubes but leads to a more imprecise abstract transition relation. Thus, one may not be able to find a controller for the abstract system. On the other hand, decreasing η (and τ) results in a more precise abstraction and a higher chance of successful controller synthesis. However, the larger state space can make the synthesis problem computationally intractable.

In this paper, we extend ABCS to a *multi-layered* setting, i.e., we maintain several "layers" of abstract systems by picking hypercube partitions of different resolution. The resulting abstract controller synthesis procedure tries to find a controller for the coarsest abstraction whenever feasible, but adaptively considers finer abstractions when necessary. We provide multi-layered controller synthesis algorithms for reachability, safety, and generalized Büchi conditions; our approach can be generalized to any ω-regular objective.

One might believe that soundness of our algorithms should immediately follow from techniques developed for abstraction refinement [5, 12]. Indeed, FRR is an instance of abstract interpretation [6]. However, abstraction refinement approaches usually assume a strong property: that there is an abstraction relation between *every* pair of abstract systems. This is not true in our setting. Thus, our algorithms and soundness proofs use a different approach. We introduce abstract multi-layered controllers which are each constructed from a set of "local" single-layered abstract controllers computed by a fixed-point algorithm. Based on ranking functions induced by such fixed-point algorithms, we show that abstract multi-layered controllers are sound. We obtain a sound ABCS method by refining

*{kylehsu,rupak,kmallik,akschmuck}@mpi-sws.org

HSCC '18, April 11–13, 2018, Porto, Portugal

ACM ISBN 978-1-4503-5642-8/18/04…$15.00
https://doi.org/10.1145/3178126.3178143

[1] Usually, τ is increased along with η to reduce non-determinism due to self loops.

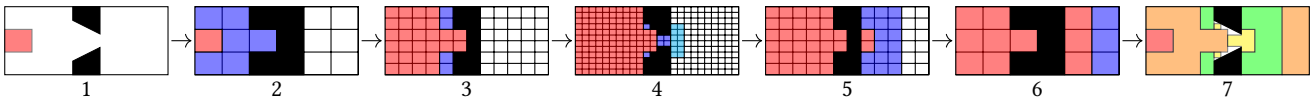

Figure 1: Application of the algorithm in Fig. 2 to the reachability problem in Pic. 1 using $m = 2$ and three layers $l = 1$ (Pic. 4), $l = 2$ (Pic. 3, 5) and $l = 3$ (Pic. 2, 6). In Pics. 1-6, the target (T) and the obstacles are indicated in red and black, respectively, and the winning states (W) are indicated in blue and cyan depending on whether they were computed in the first attempt or the second attempt, respectively. Pic. 7 indicates the domains of the resulting controllers with different granularity: $l = 1$ (yellow), $l = 2$ (green), and $l = 3$ (orange).

controllers locally using the FRR between each abstract layer and the concrete system. Our algorithms are also complete relative to single-layered synthesis on the finest layer.

We have implemented our algorithms on top of SCOTS [17]. Our implementation maintains the different layers symbolically using binary decision diagrams (BDDs) and structures the BDD representations for different layers in a way that allows for efficient switching between abstractions. We use our implementation to empirically evaluate the inherent trade-off in applying our approach to a synthesis problem: while the controller synthesis algorithm runs faster, we pay the up-front cost of constructing multiple abstract transition relations of varying coarseness. Our method is effective if the number of state variables of the concrete system is small and the cost of constructing the abstractions can be amortized over multiple controller synthesis problems. Using our implementation, we show that for Generalized Büchi control objectives arising in robot motion planning problems, one can gain significant savings in running times. More savings can be expected when the same dynamical system is used for multiple synthesis problems, e.g., in the design of motion primitives.

Informal Overview. We now provide an informal overview of our method. Consider the problem of controlling the motion of a vehicle in a given space (Pic. 1 in Fig. 1) s.t. a specified target (red region) is reached while avoiding static obstacles (black regions). Fix η and τ, and consider three abstractions S_1, S_2, and S_3 of the system, with parameters (η, τ), $(2\eta, 2\tau)$, and $(4\eta, 4\tau)$, respectively. Thus, S_1 is the finest abstraction, and S_3 is the coarsest. Each abstract state space is a set of squares (2D hypercubes) partitioning the underlying 2D space. The abstract transition function for each S_i allows transitions only between cells that share a corner.

Given Fig. 1, we see that we can only cross the passage between the obstacles by using S_1, yet S_2 and S_3 suffice in the remaining parts of the state space. This suggests an algorithm, like the one informally stated in Fig. 2, which tries to use the coarsest abstraction S_3, but switches to a finer abstraction (and back) when necessary.

When applying this algorithm to the problem in Pic. 1 of Fig. 1, we start by running a usual fixed-point algorithm for reachability (REACH) on S_3 using the red states as the target. All states returned by REACH (marked in blue in Pic. 2) allow us to apply a (sequence of) control input(s) to steer the system to the red region, and are therefore usually called *winning states*.

At this point, we proceed to the next finer layer S_2 and run REACH for S_2 by using the previously computed winning states as the new target (marked in red in Pic. 3). We see that this fixed-point terminates after one step. Therefore, we go down to S_1 (Pic. 4).

At this point, we could run the controller synthesis algorithm in S_1 until convergence. While correct, this does not capture the

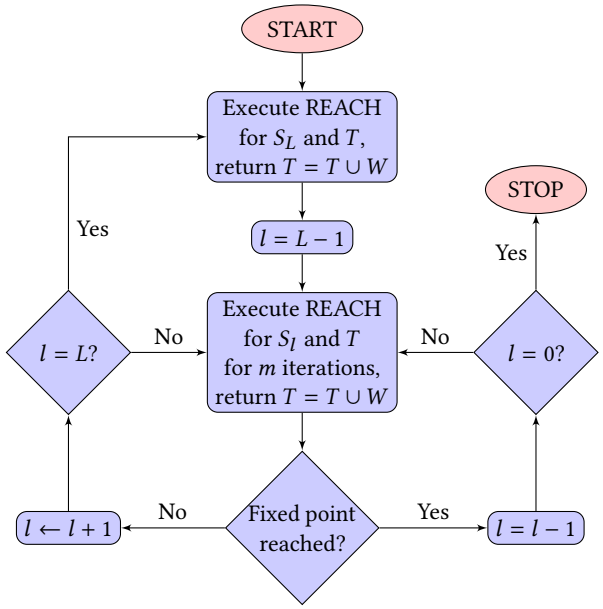

Figure 2: Flow chart of the proposed multi-layered reachability fixed-point. T and W denote the target and the winning states, respectively. There are L layers and m is a parameter determining when to switch to a more abstract layer.

intuition that S_2 and S_3 suffice for controller synthesis after crossing the passage. Hence, our synthesis algorithm uses a tuning parameter m (equal to 2 in Fig. 1) to specify that, for layers that are not the coarsest, the reachability fixed-point iterates for up to m steps and returns the current winning set. After iterating REACH on S_1 for $m = 2$ steps (gaining the blue states in Pic. 4), we switch to layer 2 and see that no new states are added to the target set for S_2 compared to Pic. 3. We thus immediately return to S_1 and run REACH for two more iterations (gaining the cyan states in Pic. 4). As REACH did not converge for S_1, we move to S_2 and run $m = 2$ iterations of REACH over S_2, but now for the new target (Pic. 5). Since this also does not converge, we finally move back to S_3 and run REACH until convergence (Pic. 6). At this point, we can verify that the set of states is a fixed-point for every layer; hence, the algorithm terminates.

As a by-product of the algorithm, we obtain a partition of the state space where each sub-region corresponds to the domain of one locally computed controller. This is illustrated in Pic. 7 of Fig. 1. The overall abstract controller is given by the union of all local ones, and its soundness follows from the fact that each local controller treats the union of all lower ranked controllers' domains

(i.e., controllers computed earlier in the algorithm) as its target. By refining this controller to the underlying concrete system, we obtain a controller with non-uniform resolution which works as follows: for each continuous state x whose related abstract state $\hat{x}$ is in the domain of one local controller computed for layer l, the control input corresponding to $\hat{x}$ is applied for duration τ_l to the underlying control system.

Related Work in ABCS. Multi-layered ABCS was first proposed by Girard et al. [3, 4] (recently revisited in [10]) in the special case of safety and reachability control of unperturbed switched systems, i.e., the considered systems and specifications are both a strict subclass of ours. Furthermore, [10] uses a modified version of approximate bisimulation relations, resulting in a deterministic abstract model, which allows for a forward search based technique to synthesize safety controllers. While forward search is usually faster for reachability or safety, it is not known how to, using a forward algorithm, efficiently synthesize controllers for ω-regular objectives or efficiently (symbolically) handle external disturbances and non-determinism in the abstraction. For this reason, we use a backward search-based symbolic technique and employ FRR as the abstraction relation. Similarly, [13] presents a multi-resolution path planning technique which adaptively reduces the time step in order to find a path to the target. Due to its use of forward computation over an unperturbed system model, it is more closely related to [10] than to our work.

In the context of multi-layered abstraction methods, [19] proposes a technique where a coarse abstraction is refined locally to bound accumulated errors w.r.t. the underlying system dynamics. Non-uniform state space partitioning has also been presented for stochastic systems [8]. Both methods are different from our technique in that refinement is triggered by the underlying system dynamics, whereas our multi-layered abstraction is induced by the multi-layered controller computed for a particular control problem. For uniformly coarse abstractions, [20] presents a technique to optimize the discretization parameter η in order to minimize the number of outgoing transitions from constructed abstract states. Furthermore, [9] discretizes inputs (instead of states) non-uniformly. Both techniques could be incorporated in our method for choosing the abstraction parameter of the lowest layer and obtaining abstract control inputs, respectively.

2 PRELIMINARIES

Notation. We use the symbols $\mathbb{N}, \mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}_{>0}, \mathbb{Z}$, and $\mathbb{Z}_{>0}$ to denote the sets of natural numbers, reals, non-negative reals, positive reals, integers, and positive integers, respectively. Given $a, b \in \mathbb{R}$ s.t. $a \leq b$, we denote by $[a, b]$ a closed interval and define $[a; b] = [a, b] \cap \mathbb{Z}$ as its discrete counterpart. Given $a, b \in \mathbb{R}^n$, we denote by a_i and b_i their i-th element and write $[\![a, b]\!]$ for the closed hyper-interval $\mathbb{R}^n \cap ([a_1, b_1] \times \ldots \times [a_n, b_n])$. We define the relations $<, \leq, \geq, >$ on a, b component-wise.

For a set W, we write W^*, W^+, and W^ω for the sets of finite sequences, non-empty finite sequences, and infinite sequences over W, respectively. We define $W^\infty = W^* \cup W^\omega$. For $w \in W^*$, we write $|w|$ for the length of w; the length of $w \in W^\omega$ is ∞. We define $\mathrm{dom}(w) = \{0, \ldots, |w| - 1\}$ if $w \in W^*$, and $\mathrm{dom}(w) = \mathbb{N}$ if $w \in W^\omega$. We denote by $\mathrm{dom}^+(w) = \mathrm{dom}(w) \setminus \{0\}$ the positive domain of w. For $k \in \mathrm{dom}(w)$ we write $w(k)$ for the k-th symbol of w and $w|_{[0,k]}$ for the restriction of w to the domain $[0, k]$. If $W = A \times B$, the projection of $w \in W^\infty$ on A is denoted by $w|_A$.

Given two sets A and B, $f : A \rightrightarrows B$ and $f : A \to B$ denote a set-valued and ordinary map, respectively. f is called *strict* if $f(a) \neq \emptyset$ for all $a \in A$. We identify set-valued maps with their respective binary relation over $A \times B$, i.e., $(a, b) \in f$ iff $b \in f(a)$. The inverse mapping $f^{-1} : B \rightrightarrows A$ is defined via its respective binary relation: $f^{-1}(b) = \{a \in A \mid b \in f(a)\}$.

2.1 Abstraction-Based Controller Synthesis

We now recall the general procedure of abstraction-based controller synthesis (ABCS) using the framework of feedback refinement relations (FRR) as introduced in [16].

Systems. A *system* $S = (X, U, Y, F, H)$ consists of a state space X, an input space U, an output space Y, and set-valued maps $F : X \times U \rightrightarrows X$ and $H : X \times U \rightrightarrows Y$ representing the transition function and the output function, respectively. A system S is *finite* if X, U, and Y are finite. It is *simple* if $X = Y$ and $H(x, u) = x$ for all $x \in X$ and $u \in U$, and *static* if X is a singleton. If S is simple (resp. static) we use the triple $S = (X, U, F)$ (resp. $S = (U, Y, H)$ with $H : U \rightrightarrows Y$) for notational convenience. The *behavior* $\mathcal{B}(S)$ of a system $S = (X, U, Y, F, H)$ is given by the set

$$\{\xi \in X^\infty \mid \forall k \in \mathrm{dom}^+(\xi) \,.\, \xi(k) \in \textstyle\bigcup_{u \in U} F(\xi(k-1), u)\}. \quad (1)$$

Controllers and Closed Loop Systems. A *controller* is a tuple $C = (Z, B, Y^c, F^c, G)$, where Z, B, and Y^c are the state, input and output space of C, respectively, and $F^c : Z \times B \rightrightarrows Z$ and $G : Z \times B \rightrightarrows Y^c$ are the transition and output function of C, respectively. Given a system $S = (X, U, F)$, we say that $C = (Z, B, Y^c, F^c, G)$ is *composable* with S if[2] $B \subseteq Y$ and $Y^c \subseteq U$. For notational simplicity, we only consider controllers where $Y^c = U$. Given a system $S = (X, U, F)$ and a controller $C = (Z, B, U, F^c, G)$, the *closed loop system* formed by interconnecting S and C in *feedback* is defined by the system[3]

$$S^{cl} = (X \times Z, U \times B, F^{cl}) \quad \text{with} \quad (2)$$

$$F^{cl}((x, z), (u, b)) = \left\{ (x', z') \;\middle|\; \begin{array}{l} u \in G(z, b), b = x, \\ x' \in F(x, u), z' \in F^c(z, b)) \end{array} \right\}.$$

Control Problem. We consider ω-regular specifications whose atomic predicates are interpreted as sets of states. We omit the standard definitions. Given a specification ψ, a system S, and an interpretation of the predicates as sets of states of S, we write $\langle\!\langle \psi \rangle\!\rangle_S \subseteq \mathcal{B}(S)$ for the set of behaviors of S satisfying ψ. The pair $\langle S, \psi \rangle$ is called a *control problem* on S for ψ. A controller C solves the control problem $\langle S, \psi \rangle$ if it is composable with S and $\mathcal{B}(S^{cl})|_X \subseteq \langle\!\langle \psi \rangle\!\rangle_S$. The set of all controllers solving $\langle S, \psi \rangle$ is denoted by $\mathcal{C}(S, \psi)$.

Feedback Refinement Relations. Let $S_i = (X_i, U_i, F_i)$, $i \in \{1, 2\}$ be two systems, and suppose $U_2 \subseteq U_1$. A *feedback refinement relation* (FRR) from S_1 to S_2 is a strict relation $Q \subseteq X_1 \times X_2$ s.t. for all $(x_1, x_2) \in Q$, we have (i) $U_{S_2}(x_2) \subseteq U_{S_1}(x_1)$, and (ii) $u \in U_{S_2}(x_2) \Rightarrow$

[2] In [16] a system is feedback composable with a controller if $X \subseteq B$. We invert this relationship; we consider controllers whose domain is only a subset of the system's state space and assume that controllers do not have other inputs.

[3] In contrast to the definition used in [16], we keep the input used in F^{cl} explicit. This allows us to apply feedback refinement relations to closed loop systems.

$Q(F_1(x_1,u)) \subseteq F_2(x_2,u)$ where $U_{S_i}(x) := \{u \in U_i \mid F_i(x,u) \neq \emptyset\}$. We write $S_1 \preccurlyeq_Q S_2$ if Q is an FRR from S_1 to S_2.

Consider two simple systems S_1 and S_2, with $S_1 \preccurlyeq_Q S_2$. Let $C = (Z, B, \hat{U}, F^c, G)$ be a controller composable with S_2. Then, as shown in [16], C can be refined into a controller composable with S_1, defined by $C \circ Q = (Z, \tilde{B}, Y^c, \tilde{F^c}, \tilde{G})$ with $\tilde{B} = Q^{-1}(B)$, $\tilde{F^c}(z,x) = \{z' \mid \exists \hat{x} \in Q(x) \,.\, z' \in F^c(z,\hat{x})\}$, and $\tilde{G}(z,x) = \{\hat{u} \mid \exists \hat{x} \in Q(x) \,.\, \hat{u} \in G(z,\hat{x})\}$ for all $z \in Z$ and $x \in B$. This implies soundness for ABCS.

Proposition 1 ([16], Def. VI.2, Thm. VI.3). *Let $S_1 \preccurlyeq_Q S_2$ and $C \in C(S_2, \psi)$ for a specification ψ. If for all $\xi_1 \in \mathcal{B}(S_1)$ and $\xi_2 \in \mathcal{B}(S_2)$ with $\mathrm{dom}(\xi_1) = \mathrm{dom}(\xi_2)$ and $(\xi_1(k), \xi_2(k)) \in Q$ for all $k \in \mathrm{dom}(\xi_1)$ it holds that $\xi_2 \in \langle\!\langle \psi \rangle\!\rangle_{S_2} \Rightarrow \xi_1 \in \langle\!\langle \psi \rangle\!\rangle_{S_1}$, then $C \circ Q \in C(S_1, \psi)$.*

2.2 ABCS for Continuous Control Systems

We now recall how ABCS can be applied to continuous-time systems by delineating the abstraction procedure [16].

Continuous-Time Control Systems. A *control system* $\Sigma = (X, U, W, f)$ consists of a state space $X = \mathbb{R}^n$, a non-empty input space $U \subseteq \mathbb{R}^m$, a disturbance space $W = [\![-w, w]\!]$ s.t. $w \in \mathbb{R}^n_{\geq 0}$ and a nonlinear differential inclusion

$$\dot{\xi} \in f(\xi(t), u(t)) + W \tag{3}$$

where $f(\cdot, u)$ is locally Lipschitz for all $u \in U$. Σ defines a perturbed continuous-time nonlinear system, and w is a component wise bound on perturbations to its dynamics.

Given a positive parameter $\tau > 0$ and a constant input trajectory $\mu_u : [0,\tau] \to U$ which maps every $t \in [0,\tau]$ to u, a solution of the inclusion in (3) on $[0,\tau]$ is an absolutely continuous function $\xi : [0,\tau] \to X$ that fulfills (3) for almost every $t \in [0,\tau]$. We collect all such solutions in the set $\mathrm{Sol}_f(\tau, u)$. Given an initial condition $x_0 \in X$, the solution to the unperturbed control system $\dot{\xi} = f(\xi(t), u(t))$ associated with (3) is unique, and its value at time $t \in [0,\tau]$ is denoted by $\zeta(t, x_0, \mu)$. Given a subset of states $X' \subseteq X$, and a subset of inputs $U' \subseteq U$ s.t. $[0,\tau] \times X' \times U' \subseteq \mathrm{dom}(\zeta)$, the map $\beta_\tau : \mathbb{R}^n_{\geq 0} \times U' \to \mathbb{R}^n_{\geq 0}$ is a *growth bound* on X' and U' associated with τ and (3) if

$$\forall r, r' \in \mathbb{R}^n_{\geq 0}, u \in U' \,.\, r \geq r' \Rightarrow \beta_\tau(r,u) \geq \beta_\tau(r',u) \text{ and}$$
$$\forall\, \xi \in \mathrm{Sol}_f(\tau, u), x_0 \in X' \,.$$
$$\xi(0) \in X' \Rightarrow |\xi(\tau) - \zeta(\tau, x_0, \mu)| \leq \beta_\tau(|\xi(0) - x_0|, u).$$

Time-Sampled System. Given a time sampling parameter $\tau > 0$, we define by $\vec{S}(\Sigma, \tau) = (X, U, \vec{F})$ the (simple) *time-sampled system* associated with Σ, where

$$x' \in \vec{F}(x,u) \Leftrightarrow \exists \xi \in \mathrm{Sol}_f(\tau, u) \,.\, \xi(0) = x \wedge \xi(\tau) = x'. \tag{4}$$

The state space of $\vec{S}(\Sigma, \tau)$ is still infinite; we next define a finite system associated with Σ.

Abstract System. A *cover* $\hat{X}$ of the state space X is a set of non-empty, closed hyper-intervals $[\![a, b]\!]$ with $a, b \in (\mathbb{R} \cup \{\pm\infty\})^n$ called *cells*, such that every $x \in X$ belongs to some cell in $\hat{X}$. We assume that there exists a compact subset $X' \subseteq X$ of the state space, which is quantized by compact cells, whereas the (unbounded) region covered by $\hat{X} \setminus \hat{X}'$ is not of interest to the control problem and is covered by a finite number of large unbounded cells.

Given a *grid parameter* $\eta \in \mathbb{R}^n_{>0}$ and $X' \subseteq X$ with $X' = [\![\alpha, \beta]\!]$ s.t. $\beta - \alpha = k\eta$ for some $k \in \mathbb{Z}^n$, the set

$$\eta\mathbb{Z}^n = \{c \in X' \mid \exists k \in \mathbb{Z}^n.(\forall i \in [1;n].c_i = \alpha_i + k_i\eta_i - 0.5 * \eta_i)\} \tag{5}$$

defines the center points of cells in $\hat{X}'$ with diameter η, i.e.

$$\hat{x} \in \hat{X}' \Rightarrow \exists c \in \eta\mathbb{Z}^n \,.\, \hat{x} = c + [\![-\eta/2, \eta/2]\!]. \tag{6}$$

This results in congruent cells which are uniformly aligned on a grid.[4] We denote by $c_{\hat{x}}$ the unique center point of $\hat{x}$.

The (simple) system $\hat{S}(\Sigma, \tau, \eta, \beta_\tau) = (\hat{X}, \hat{U}, \hat{F})$ is a *symbolic abstract system* associated with Σ, τ, η, and β_τ if the following holds: (i) $\hat{X}$ is a finite cover of X, there exists a non-empty subset $\hat{X}' \subseteq \hat{X}$ s.t. $\hat{X}'$ satisfies (6), and β_τ is a growth bound[5] on $\hat{X}'$ and $\hat{U}$, (ii) a finite $\hat{U} \subseteq U$, (iii) for all $\hat{x} \in \hat{X} \setminus \hat{X}'$ and $u \in \hat{U}$, $\hat{F}(\hat{x}, u) = \emptyset$, and (iv) for all $\hat{x} \in \hat{X}'$, $\hat{x}' \in \hat{X}$, and $u \in \hat{U}$, $\hat{x}' \in \hat{F}(\hat{x}, u)$ iff

$$\left(\zeta(\tau, c_{\hat{x}}, u) + [\![-\beta_\tau(\frac{\eta}{2}, u), \beta_\tau(\frac{\eta}{2}, u)]\!]\right) \cap \hat{x}' \neq \emptyset. \tag{7}$$

If the control system Σ and parameters τ, η, and β_τ are clear from the context, we omit them in $\vec{S}$ and $\hat{S}$.

Induced FRR. It was shown in [16], Thm. III.5 that the relation $\hat{Q} \subseteq X \times \hat{X}$ defined by $(x, \hat{x}) \in \hat{Q}$ iff $x \in \hat{x}$ is an FRR between $\vec{S}$ and $\hat{S}$, i.e., $\vec{S} \preccurlyeq_{\hat{Q}} \hat{S}$. Hence, we can apply ABCS as described in Sec. 2.1 by computing a controller for $\hat{S}$ which can then be refined to a controller for $\vec{S}$ under the pre-conditions of Prop. 1.

3 A MULTI-LAYERED SETTING

We now extend ABCS to a multi-layered setting with L layers. The abstract states in layer $l \in [1;L]$ are hyper-cells with parameter η_l, and the transitions are discrete jumps with sampling time τ_l. A control system Σ may be abstracted *non-uniformly* by using different layers for different parts of the state space.

3.1 Multi-Layered Systems

Given a grid parameter $\underline{\eta}$, a time sampling parameter $\underline{\tau}$, and $L \in \mathbb{Z}_{>0}$, define[6] $\eta_l = 2^{l-1}\underline{\eta}$ and $\tau_l = 2^{l-1}\underline{\tau}$. Fix a control system Σ and a subset $X' \subseteq X$ with $X' = [\![\alpha, \beta]\!]$, s.t. $\beta - \alpha = k\eta_L$ for some $k \in \mathbb{Z}^n$. For each $l \in [1;L]$, we define the grid $\eta_l\mathbb{Z}^n$ and the cover $\hat{X}'_l$ as in (5) and (6), respectively, by substituting η with η_l. This construction induces a sequence of time-sampled systems $\{\vec{S}_l(\Sigma, \tau_l)\}_{l\in[1;L]}$ and a sequence of symbolic abstract systems $\{\hat{S}_l(\Sigma, \tau_l, \eta_l, \beta_{\tau_l})\}_{l\in[1;L]}$. For simplicity, we assume that all layers use the same continuous and abstract input spaces U and $\hat{U} \subseteq U$, respectively. If Σ, τ_l, η_l, and β_{τ_l} are clear from the context, we use $\vec{S}_l$ and $\hat{S}_l$ as short forms of $\vec{S}_l(\Sigma, \tau_l)$ and $\hat{S}_l(\Sigma, \tau_l, \eta_l, \beta_{\tau_l})$, respectively.

It trivially follows from our construction that, for all $l \in [1;L]$, we have $\vec{S}_l \preccurlyeq_{\hat{Q}_l} \hat{S}_l$, where $\hat{Q}_l \subseteq X \times \hat{X}_l$ is the FRR induced by $\hat{X}_l$.

[4] In the standard distribution of SCOTS, the grid is aligned such that (an extension of) $\eta\mathbb{Z}^n$ has a center point that coincides with the origin. Shifting this grid by $\eta/2$ yields our grid alignment.

[5] We consider a universal growth bound for notational simplicity. There exist systems where abstract controller synthesis is successful only if different growth bounds for different regions of the state space are considered (see [20]).

[6] Our method is applicable to grid parameters defined by $\eta_1 = \underline{\eta}$ and $\eta_{l+1} = \gamma\eta_l$ for $l \in [1; L-1]$ and $\gamma \in \{2^k \mid k \in \mathbb{N}\}$, and sampling times $\tau_l = \alpha(l)\underline{\tau}$ where $\alpha : [1;L] \to \mathbb{R}$ is a monotonically increasing function. For notational simplicity, we restrict our attention to $\gamma = 2$ and $\alpha(l) = 2^{l-1}$.

The set of relations $\{\hat{Q}_l\}_{l\in[1;L]}$ induces transformers $\hat{R}_{ll'} \subseteq \hat{X}_l \times \hat{X}_{l'}$ for $l, l' \in [1;L]$ between abstract states of different layers such that

$$\hat{x} \in \hat{R}_{ll'}(\hat{x}') \Leftrightarrow \hat{x} \in \hat{Q}_l(\hat{Q}_{l'}^{-1}(\hat{x}')). \tag{8}$$

Given the sequence $\{\hat{S}_l\}_{l\in[1;L]}$ with $\hat{S}_l = (\hat{X}_l, \hat{U}, \hat{F}_l)$, we can use $\hat{R}_{ll'}$ to define the (simple) *abstract multi-layered system* $\hat{\mathbf{S}} = (\hat{\mathbf{X}}, \hat{U}, \hat{\mathbf{F}})$, where $\hat{\mathbf{X}} = \bigcup_{l\in[1;L]} \hat{X}_l$ and

$$\hat{\mathbf{F}}(\hat{x}, \hat{u}) = \bigcup_{l\in[1;L]} \hat{R}_{ll'}(\hat{F}_{l'}(\hat{x}, \hat{u})) \tag{9}$$

for all $\hat{x} \in \hat{X}_{l'}$, $l' \in [1;L]$, and $\hat{u} \in \hat{U}$. Intuitively, $\hat{\mathbf{S}}$ is a non-deterministic system; for every state $\hat{x} \in \hat{X}_{l'}$ and input $\hat{u} \in \hat{U}$, transitions to states of all layers are possible. That is, $\hat{x}' \in \hat{\mathbf{F}}(\hat{x}, \hat{u})$ if there exists an $\hat{x}'' \in \hat{X}_{l'}$ s.t. $\hat{x}'' \in \hat{F}_{l'}(\hat{x}, \hat{u})$ and $\hat{x}' \in \hat{R}_{ll'}(\hat{x}'')$ for some $l \in [1;L]$.

Similarly, given the sequence $\{\vec{S}_l\}_{l\in[1;L]}$ with $\vec{S}_l = (X_l, U, \vec{F}_l)$, we define a (simple) *time-sampled multi-layered system* $\vec{\mathbf{S}} = (X, U, \vec{\mathbf{F}})$ s.t.

$$\vec{\mathbf{F}}(x, u) = \bigcup_{l\in[1;L]} \vec{F}_l(x, u) \tag{10}$$

for all $x \in X$ and $u \in U$. Again, $\vec{\mathbf{S}}$ is a non-deterministic system; in every state $x \in X$ transitions of any duration τ_l, $l \in [1;L]$ can be chosen, which correspond to some $\vec{F}_l(x, u)$.

The behaviors $\mathcal{B}(\hat{\mathbf{S}})$ and $\mathcal{B}(\vec{\mathbf{S}})$ are defined via (1).

Remark 1. *Even though we have $\vec{S}_l \preccurlyeq_{\hat{Q}_l} \hat{S}_l$ for all $l \in [1;L]$, each relation is evaluated for a different sampling time τ_l. Therefore, the relations $\hat{R}_{ll'}$ cannot define a FRR between $\hat{S}_l$ and $\hat{S}_{l'}$ using the definition from Sec. 2.*

Given that $\tau_l = 2^{l-1}\underline{\tau}$, a natural extension of FRR seems to be that, for any $(\hat{x}_{l+1}, \hat{x}_l) \in \hat{R}_{(l+1)l}$, it holds that

$$\hat{u} \in \hat{U} \Rightarrow \hat{R}_{(l+1)l}(\hat{F}_l(\hat{F}_l(\hat{x}_l, \hat{u}), \hat{u})) \subseteq \hat{F}_{l+1}(\hat{x}_{l+1}, \hat{u}). \tag{11}$$

That is, we would expect that states $\hat{x}_l \in \hat{X}_l$ and $\hat{x}_{l+1} \in \hat{X}_{l+1}$ related via $\hat{R}_{(l+1)l}$ remain related when the l-th layer transition function is applied to $\hat{x}_l$ twice for τ_l (resulting in a duration $2\tau_l = \tau_{l+1}$) and when the $l+1$-th layer transition function is only applied once to $\hat{x}_{l+1}$ (also resulting in a duration τ_{l+1}). While this seems intuitive, we can prove that (11) *does not hold in general. This is because applying the l-th layer transition function twice introduces an additional over-approximation step which is caused by* (7).

3.2 Multi-Layered Controllers

Given $\hat{\mathbf{S}}$ as in (9) and some $P \in \mathbb{N}$, we define a *multi-layered controller* compatible with $\hat{\mathbf{S}}$ as the tuple

$$\mathbf{C} = \left(Z, \{B^p\}_{p\in[1;P]}, \hat{U}, F^c, \{G^p\}_{p\in[1;P]}\right) \tag{12}$$
$$\text{s.t. } \forall p \in [1;P] \,.\, \exists! l_p \in [1;L] \,.\, B^p \subseteq \hat{X}_{l_p},$$

and $F^c : Z \times \hat{\mathbf{X}} \rightrightarrows Z$ and $G^p : Z \times B^p \rightrightarrows \hat{U}$ are the transition and output functions of $\mathbf{C}$, respectively. We call $\mathbf{C}$ static, if Z is a singleton. In this case, we can omit the state dependence of $\mathbf{C}$ and simplify its notation to the set $\mathbf{C} = \{C^p\}_{p\in[1;P]}$ with $C^p = (B^p, \hat{U}, G^p)$ and $G^p : B^p \rightrightarrows \hat{U}$. We denote by $\mathrm{dom}(G^p, z) = \{\hat{x} \in \hat{X}_{l_p} \mid G^p(z, \hat{x}) \neq \emptyset\}$ the domain of G^p w.r.t. $z \in Z$. We do not require any connection between P and L. In particular, we allow for multiple controllers to be composable with the same layer, i.e., $l_p = l_q$ for $p, q \in [1;P]$, $p \neq q$, and there might exist an $l \in [1;L]$ s.t. there exists no $p \in [1;P]$ s.t. $l_p = l$.

Given a multi-layered controller $\mathbf{C}$ compatible with $\hat{\mathbf{S}}$, we define the *quantizer induced by* $\mathbf{C}$ as the strict map $\mathbf{Q} : (X \times Z) \rightrightarrows (\hat{\mathbf{X}} \times Z)$ s.t. for all $(x, z) \in X \times Z$ it holds that $(\hat{x}, z) \in \mathbf{Q}(x, z)$ iff either
(i) there exists a $p \in [1;P]$ s.t.

$$\hat{x} \in \hat{Q}_{l_p}(x) \wedge \hat{x} \in \mathrm{dom}(G^p, z)$$

and there exists no other $p' \in [1;P]$ with $l_{p'} > l_p$ s.t.

$$\hat{R}_{l_{p'} l_p}(\hat{x}) \in \mathrm{dom}(G^{p'}, z),$$

or (ii) $\hat{x} \in \hat{Q}_1(x)$ and there exists no $p \in [1;P]$ s.t.

$$\hat{R}_{l_p 1}(\hat{x}) \in \mathrm{dom}(G^p, z).$$

We identify $\mathbf{Q}$ with the map $\mathbf{Q}_z : X \rightrightarrows \hat{\mathbf{X}}$ s.t. $\hat{x} \in \mathbf{Q}_z(x)$ iff $(\hat{x}, z) \in \mathbf{Q}(x, z)$. We define $\mathrm{img}(\mathbf{Q}_z) = \{\hat{x} \in \hat{\mathbf{X}} \mid \exists x \in X \,.\, \hat{x} \in \mathbf{Q}_z(x)\}$. If $\mathbf{C}$ is strict, $\mathbf{Q}_z$ is simply denoted by $\mathbf{Q}$.

Intuitively, $\mathbf{Q}_z$ maps states $x \in X$ to the coarsest abstract state $\hat{x}$ that is both related to x and in the domain of the controller $\mathbf{C}$ (condition (i)). However, if such an abstract state does not exist, $\mathbf{Q}_z$ maps x to its related layer $l = 1$ states (condition (ii)). It should be noted that $\hat{Q}_l$ is non-deterministic for states which lie at the boundary of two cells $\hat{x}, \hat{x}' \in \hat{X}_l$. If such an x happens to be located at the boundary of controller domains computed for different layers, $\mathbf{Q}_z$ maps x to two abstract cells within different layers.

3.3 Multi-Layered Closed Loops

Given a multi-layered controller $\mathbf{C}$ which is compatible with the multi-layered abstract system $\hat{\mathbf{S}}$, the usual construction of a closed loop in (2) results in a system which selects the layer l' of the next state non-deterministically (due to (9)). This will result in a blocking-behavior of the closed loop whenever the selected layer does not correspond to a layer currently admitting a control input.

We therefore propose a different closed-loop definition for multi-layered systems which restricts available transitions to those connecting states in the image of $\mathbf{Q}$. Formally, the closed loop formed by $\hat{\mathbf{S}}$ and $\mathbf{C}$ is defined as the *abstract multi-layered closed-loop system*

$$\hat{\mathbf{S}}^{cl} = (\hat{\mathbf{X}} \times Z, \hat{U} \times B, \hat{\mathbf{F}}^{cl}) \text{ s.t.} \tag{13}$$
$$\hat{\mathbf{F}}^{cl}((\hat{x}, z), (\hat{u}, b)) = \left\{ (\hat{x}', z') \,\middle|\, \begin{array}{l} \hat{x} \in \mathrm{img}(\mathbf{Q}_z) \cap \hat{X}_{l_p} \\ \hat{u} \in G^p(z, \hat{x}),\ b = \hat{x} \\ (\hat{x}', z') \in \Lambda((\hat{x}, z), \hat{u}) \end{array} \right\}$$

where $(\hat{x}', z') \in \Lambda((\hat{x}, z), \hat{u})$ iff

$$\exists (\hat{x}'', z') \in \hat{F}_{l_p}(\hat{x}, \hat{u}) \times F^c(z, \hat{x}) \,.\, \hat{x}' \in \mathbf{Q}_{z'}(\hat{Q}_{l_p}^{-1}(\hat{x}'')).$$

When refining $\mathbf{C}$ via $\mathbf{Q}$ to a controller for $\vec{\mathbf{S}}$, the resulting controller should select (i) the current input $u \in \hat{U} \subseteq U$, and (ii) the duration τ_l for which this input should be applied to the underlying continuous system. As $\mathbf{Q}$ might map a particular state $x \in X$ to multiple abstract states within different layers, we cannot define the refined controller as the serial composition $\mathbf{C} \circ \mathbf{Q}$ in analogy to Sec. 2.1. Instead, we directly define the *time-sampled multi-layered closed loop system* in analogy to $\hat{\mathbf{S}}^{cl}$ by

$$\vec{\mathbf{S}}^{cl} = (X \times Z, \hat{U} \times B, \vec{\mathbf{F}}^{cl}) \text{ s.t.} \tag{14}$$
$$\vec{\mathbf{F}}^{cl}((x,z),(\hat{u},\hat{x})) = \left\{ (x',z') \middle| \begin{matrix} \hat{x} \in \mathbf{Q}_z(x) \cap \hat{X}_{l_p},\ \hat{u} \in G^p(z,\hat{x}) \\ x' \in \vec{F}_{l_p}(x,\hat{u}),\ z' \in F^c(z,\hat{x}) \end{matrix} \right\}.$$

$\mathcal{B}(\hat{\mathbf{S}}^{cl})$ and $\mathcal{B}(\vec{\mathbf{S}}^{cl})$ can now be defined via (1).

3.4 Soundness

Intuitively, $S_1 \preccurlyeq_Q S_2$ ensures that, given a state $x_1 \in X_1$, no matter which related state $x_2 \in Q(x_1)$ and enabled control input $u \in U_{S_2}(x_2)$ is used, all resulting states in $F_1(x_1,u)$ and $F_2(x_2,u)$ are related. This intuition can be transferred to the closed loop systems $\hat{\mathbf{S}}^{cl}$ and $\vec{\mathbf{S}}^{cl}$ as follows. Intuitively, $\hat{\mathbf{S}}^{cl}$ and $\vec{\mathbf{S}}^{cl}$ can generate all closed-loop trajectories without resolving the non-determinism induced by the set-valued maps $\mathbf{Q}$ and G^p. If $\mathbf{Q}$ is a FRR from $\vec{\mathbf{S}}^{cl}$ to $\hat{\mathbf{S}}^{cl}$, any implementation resolving this non-determinism in $\mathbf{Q}$ and G^p returns a sound closed loop. This leads us to the following theorem.

Theorem 1. *Let* $\mathbf{C}$ *be a multi-layered controller composable with the abstract multi-layered system* $\hat{\mathbf{S}}$, $\mathbf{Q}$ *the quantizer induced by* $\mathbf{C}$, *and* $\vec{\mathbf{S}}^{cl}$ *and* $\hat{\mathbf{S}}^{cl}$ *the time-sampled and abstract multi-layered closed loop systems defined in* (14) *and* (13), *respectively. Then* $\vec{\mathbf{S}}^{cl} \preccurlyeq_{\mathbf{Q}} \hat{\mathbf{S}}^{cl}$.

Proof. We prove both conditions for FRR separately.
(i) Show $\hat{\mathbf{F}}^{cl}((\hat{x},z),(\hat{u},\hat{x})) \neq \emptyset \Rightarrow \vec{\mathbf{F}}^{cl}((x,z),(\hat{u},\hat{x})) \neq \emptyset$: First observe that for any $l \in [1;L]$, it follows from the construction of $\vec{S}_l$ and $\hat{S}_l$ from Σ that, for any $\hat{u} \in \hat{U} \subseteq U$, $\hat{x} \in \hat{X}_l$, and $x \in X$, it holds that $\hat{F}_l(\hat{x},\hat{u}) \neq \emptyset$ and $\vec{F}_l(x,\hat{u}) \neq \emptyset$. With this, it follows from (13) and the fact that $\mathbf{Q}_z$ is strict that $\hat{\mathbf{F}}^{cl}((\hat{x},z),(\hat{u},\hat{x})) \neq \emptyset$ iff $\hat{x} \in \mathrm{img}(\mathbf{Q}_z) \cap \hat{X}_{l_p}$ and $\hat{u} \in G^p(z,\hat{x})$, implying $\hat{\mathbf{F}}^{cl}((\hat{x},z),(\hat{u},\hat{x})) \neq \emptyset$ (from (14)).
(ii) Pick $((x,z),(\hat{x},z)) \in \mathbf{Q}$ and $(\hat{u},\hat{x}) \in U_{\hat{\mathbf{S}}^{cl}}(\hat{x},z)$ and show $\mathbf{Q}(\vec{\mathbf{F}}^{cl}((x,z),(\hat{u},\hat{x}))) \subseteq \hat{\mathbf{F}}^{cl}((\hat{x},z),(\hat{u},\hat{x}))$: first, consider the case that $\hat{x} \in \hat{Q}_1(x)$ and case (ii) in the definition of $\mathbf{Q}$ holds. Then $\hat{\mathbf{F}}^{cl}((\hat{x},z)(\hat{u},\hat{x})) = \emptyset$ and hence $U_{\hat{\mathbf{S}}^{cl}}(\hat{x},z) = \emptyset$, i.e., the statement trivially holds. Therefore, assume that case (i) of the definition of $\mathbf{Q}$ holds, implying that there exists some $p \in [1;P]$ s.t. $\hat{x} \in \mathbf{Q}_z(x) \cap \hat{X}_{l_p}$ and $\hat{x} \in \mathrm{dom}(G^p,z)$. This implies $\hat{u} \in G^p(\hat{x},z)$ (from part (i)) and therefore $(\hat{x}',z') \in \mathbf{Q}(\vec{\mathbf{F}}^{cl}((x,z),(\hat{u},\hat{x}))$ iff

$$z' \in F^c(z,\hat{x}) \wedge \hat{x}' \in \mathbf{Q}_{z'}(\vec{F}_{l_p}(x,\hat{u})). \tag{15}$$

Now recall that $\vec{S}_{l_p} \preccurlyeq_{\hat{Q}_{l_p}} \hat{S}_{l_p}$, hence $\hat{Q}_{l_p}(\vec{F}_{l_p}(x,\hat{u})) \subseteq \hat{F}_{l_p}(\hat{x},\hat{u})$. With this we see that the second term in (15) implies $\hat{x}' \in \mathbf{Q}_{z'}(\hat{Q}_{l_p}^{-1}(\hat{F}_{l_p}(\hat{x},\hat{u})))$ and hence (15) implies $(\hat{x}',z') \in \Lambda((\hat{x},z),\hat{u})$. With this, it immediately follows from (13) that $(\hat{x}',z') \in \hat{\mathbf{F}}^{cl}((\hat{x},z),(\hat{u},\hat{x}))$. □

Using the properties of feedback refinement relations, Thm. 1 implies that the usual soundness property of ABCS stated in Prop. 1 can be transferred to the multi-layered setting. This is summarized by the following corollary.

Corollary 1. *Given the preliminaries of Thm. 1, let* $\mathbf{C} \in \mathcal{C}(\hat{\mathbf{S}},\psi)$ *for a specification* ψ *with associated behavior* $[\![\psi]\!]_{\hat{\mathbf{S}}} \subseteq \mathcal{B}(\hat{\mathbf{S}})$ *and* $[\![\psi]\!]_{\vec{\mathbf{S}}} \subseteq \mathcal{B}(\vec{\mathbf{S}})$. *Suppose that for all* $\xi \in \mathcal{B}(\vec{\mathbf{S}})$ *and* $\hat{\xi} \in \mathcal{B}(\hat{\mathbf{S}})$ *s.t. (i)* $\mathrm{dom}(\xi) = \mathrm{dom}(\hat{\xi})$ *and (ii) for all* $k \in \mathrm{dom}(\xi_1)$ *there exists a* $z \in Z$ *s.t.* $(\xi(k),\hat{\xi}(k)) \in \mathbf{Q}_z$, *it also holds that (iii)* $\hat{\xi} \in [\![\psi]\!]_{\hat{\mathbf{S}}} \Rightarrow \xi \in [\![\psi]\!]_{\vec{\mathbf{S}}}$. *Then* $\mathcal{B}(\vec{\mathbf{S}}^{cl})|_X \subseteq [\![\psi]\!]_{\vec{\mathbf{S}}}$, *i.e., the time-sampled multi-layered closed loop* $\vec{\mathbf{S}}^{cl}$ *defined in* (14) *fulfills specification* ψ.

Cor. 1 can be interpreted as follows. Consider the control system Σ at state x_0. This state is mapped by $\mathbf{Q}_{z_0}$ to $\hat{x}_0 \in B^p$. Choosing any $u_0 \in G^p(x_0,z_0)$ and applying this input for time τ_{l_p} to Σ results in a continuous trajectory ξ with $x_0 = \xi(0)$ and $x_1 = \xi(\tau_{l_p}) \in \vec{F}_{l_p}(x_0,u)$. Reapplying this procedure leads to an infinite trajectory ξ, with sampled version $\vec{\xi} = x_0x_1 \ldots \in \mathcal{B}(\vec{\mathbf{S}})$ and abstract version $\hat{\xi} = \hat{x}_0\hat{x}_1 \ldots \in \mathcal{B}(\hat{\mathbf{S}})$. As $\mathbf{C} \in \mathcal{C}(\hat{\mathbf{S}},\psi)$ condition (iii) in Cor. 1 ensures that $\vec{\xi} \in [\![\psi]\!]_{\vec{\mathbf{S}}}$. For reachability, this implies that $\vec{\xi}$ (and ξ) eventually reaches the target. For safety, this implies that $\vec{\xi}$ (i.e. sampling instances of ξ) lies inside the safe set.

4 MULTI-LAYERED SYNTHESIS

We now provide algorithms for synthesizing multi-layered controllers for given (abstract) control problems which are compatible with $\hat{\mathbf{S}}$. We use the following notational conventions from μ-calculus [2] and linear temporal logic [15]. Let f denote a monotone operator on a finite set Q, i.e., $f(P') \subseteq f(P'') \subseteq Q$ for all $P' \subseteq P'' \subseteq Q$. The least and greatest fixed-points exist uniquely and are denoted $\mu P.f(P)$ and $\nu P.f(P)$, respectively.

We use $\Diamond$ and $\Box$ to denote the temporal operators "eventually" and "always".

4.1 Reachability

Given an abstract multi-layered system $\hat{\mathbf{S}} = (\hat{\mathbf{X}},\hat{U},\hat{\mathbf{F}})$, a *reachability control problem* is the control problem $(\hat{\mathbf{S}},\Diamond T)$ where T is interpreted as a subset of $\hat{X}_1'$ and

$$[\![\Diamond T]\!]_{\hat{\mathbf{S}}} = \{\hat{\xi} \in \mathcal{B}(\hat{\mathbf{S}}) \mid \exists k \in \mathrm{dom}(\hat{\xi})\ .\ \hat{\xi}(k) \in T\}. \tag{16}$$

We furthermore define $\mathrm{Pre}_{\hat{S}_l}(A_l) = \{\hat{x} \mid \exists \hat{u}\ .\ \hat{F}_l(\hat{x},\hat{u}) \subseteq A_l\}$ for any $A_l \subseteq \hat{X}_l$ and $l \in [1;L]$.

Single-Layered Systems. When $L = 1$, the reachability control problem reduces to $(\hat{S}_1,\Diamond T)$, and is solved by computing the minimal fixed-point [14]:

$$\mathcal{W}[\Diamond T] = \mu W\ .\ \mathrm{Pre}_{\hat{S}_1}(W) \cup T. \tag{17}$$

This fixed-point is evaluated iteratively by computing

$$W^0 = T \text{ and } W^{i+1} = \mathrm{Pre}_{\hat{S}_1}(W^i) \cup T \tag{18}$$

until we reach some $N \in \mathbb{N}$ s.t. $W^N = W^{N+1}$. This defines a *static* controller $C = (B,\hat{U},G)$ with $B = W^N \setminus T$ and

$$G(x) = \{\hat{u} \in \hat{U} \mid \hat{F}(x,\hat{u}) \subseteq W^{i^*}\} \tag{19}$$

for all $x \in B$, where $i^* = \min\{i \mid x \in W^i \setminus T\} - 1$. The controller C is sound: the closed loop system $\hat{S}_1^{cl}$ composed from C and $\hat{S}_1$ satisfies $\mathcal{B}(\hat{S}_1^{cl}) \subseteq [\![\Diamond T]\!]_{\hat{S}_1}$ and the system can be steered to the target T from any state x in the controller domain B.

Using the set $\{W^i\}_{i\in[1;N]}$ computed by (18) one can also define a multi-layered controller $\mathbf{C} = \{C^i\}_{i\in[1;N]}$ by interpreting each iteration in (18) as the computation of a simple controller $C^i = (B^i,\hat{U},G^i)$, with domain $B^i = W^i \setminus W^{i-1}$ and output function

$G^i(x) = \{\hat{u} \in \hat{U} \mid \hat{F}_1(x, \hat{u}) \subseteq W^{i-1}\}$. This interpretation of (18), not too useful when $L = 1$, is convenient for the multi-layered setting.

Multi-Layered Systems. The simplest way to extend $\mathbf{C}$ to the general case ($L > 1$) is to pick an arbitrary granularity l_i in every iteration of (18). First, we introduce the operator

$$\Gamma_{ll'}(A_{l'}) = \begin{cases} \hat{R}_{ll'}(A_{l'}), & l \leq l' \\ \{\hat{x}_l \in \hat{X}_l \mid \hat{R}_{l'l}(\hat{x}_l) \subseteq A_{l'}\}, & l > l'. \end{cases} \tag{20}$$

which underapproximates a set $A_{l'} \subseteq \hat{X}_{l'}$ by a set $A_l = \Gamma_{ll'}(A_{l'}) \subseteq \hat{X}_l$. Then, for any sequence $\{l_i\}_{i \geq 1}$ with $l_i \in [1; L]$, we compute

$$W^0 = T,\ l_0 = 1 \text{ and}$$
$$W^{i+1} = \mathrm{Pre}_{\hat{S}_{l_{i+1}}}(\Gamma_{l_{i+1}l_i}(W^i)) \cup \Gamma_{l_{i+1}l_1}(T)$$

iteratively until $W^{i+1} = \Gamma_{l_{i+1}l_i}(W^i)$. When the fixed-point converges, we get a multi-layered controller $\mathbf{C}$ as described before. It is easy to see that $\mathbf{C}$ is sound no matter how $\{l_i\}$ is chosen: each state in the winning set can be controlled using $\mathbf{C}$ to reach T. Unfortunately, this simple algorithm is not complete due to two reasons: (i) the iteration can reach a fixed-point at level l even if more states can be added at a lower level, and (ii) lower level winning states can be lost via $\Gamma_{ll'}$, which only gives an exact map for $l < l'$ (recall Pic. 2 and Pic. 3 in Fig. 1).

To fix these problems, we propose REACHIT$_m$ (Alg. 1), which follows the flowchart in Fig. 2. The overall multi-layered reachability algorithm, ML_REACH$_m$ (with tuning parameter $m \in \mathbb{N}$), calls REACHIT$_m(T, l, \emptyset)$ with target T, level l, and an empty controller, and returns its result. The subroutine REACH$_m$ called in Alg. 1 (line 2 and 12) runs m iterations of (18). Formally, given the input $\Lambda \subseteq \hat{X}_l$ and $l \in [1; L]$, it returns a set $W \subseteq \hat{X}_l$ s.t. $W = W^j$, where W^j is computed via (18) and either $j \leq m$ with $W^j = W^{j-1}$ (then line 15 of Alg. 1 is true) or $j = m$ otherwise. Additionally, REACH$_m$ returns the controller $C = (W \setminus \Lambda, \hat{U}, G)$ with G computed using (19). We extend REACH$_m$ to REACH$_\infty$ in the obvious way, s.t. the returned W is the fixed-point of (18). The set Υ is used in Alg. 1 to save computed winning states to the lowest layer $l = 1$. To see why this is necessary, recall Pic. 4 in Fig. 1. There, we first computed REACH for $l = 1$ and $m = 2$ (obtaining the blue states) and moved to $l = 2$; observe that no winning states were added in $l = 2$. Had we not saved the blue states in Υ, they would have been lost and been re-generated after returning to $l = 1$, causing an infinite loop.

Soundness and Completeness. To see why ML_REACH$_m$ is sound, consider the following induction on the depth of recursive calls d of REACHIT$_m$. For depth 1 (base case), the single controller that we get in the set $\mathbf{C}$ is sound: this follows from the fact that REACH$_\infty$ (in Alg. 1 Line. 2) is sound. Moreover, it is easy to observe that the controller obtained in depth $d + 1$ can enforce a visit (in at most m steps) to the winning region of the controller obtained in depth d (from Line 8, 19, and 23), implying soundness by induction.

Now recall that any controller C computed via REACH has the property that any state in B can reach the target. Therefore, the multi-layered controller $\mathbf{C}$ computed by ML_REACH$_m$ is complete w.r.t. the single-layered controller C computed via REACH if

$$\hat{Q}_1^{-1}(B) \subseteq \bigcup_{d \in [1;D]} \mathbf{Q}^{-1}(B^d), \tag{21}$$

Algorithm 1 REACHIT$_m$

Require: $\Upsilon \subseteq \hat{X}_1$, l, $\mathbf{C}$
1: **if** $l = L$ **then**
2: $\langle W, C\rangle \leftarrow$ REACH$_\infty(\Gamma_{L,1}(\Upsilon), l)$
3: $\mathbf{C} \leftarrow \mathbf{C} \cup \{C\}$
4: $\Upsilon \leftarrow \Upsilon \cup \Gamma_{1l}(W)$ // Save W to Υ
5: **if** $L = 1$ **then** // Single-layered reachability
6: **return** $\langle \Upsilon, \mathbf{C}\rangle$
7: **else**
8: $\langle \Upsilon, \mathbf{C}\rangle \leftarrow$ REACHIT$_m$ $(\Upsilon, l - 1, \mathbf{C})$
9: **return** $\langle \Upsilon, \mathbf{C}\rangle$
10: **end if**
11: **else**
12: $\langle W, C\rangle \leftarrow$ REACH$_m(\Gamma_{l,1}(\Upsilon), l)$
13: $\mathbf{C} \leftarrow \mathbf{C} \cup \{C\}$
14: $\Upsilon \leftarrow \Upsilon \cup \Gamma_{1l}(W)$ // Save W to Υ
15: **if** Fixed-point is reached in line 12 **then**
16: **if** $l = 1$ **then** // Finest layer reached
17: **return** $\langle \Upsilon, \mathbf{C}\rangle$
18: **else** // Go finer
19: $\langle \Upsilon, \mathbf{C}\rangle \leftarrow$ REACHIT$_m(\Upsilon, l - 1, \mathbf{C})$
20: **return** $\langle \Upsilon, \mathbf{C}\rangle$
21: **end if**
22: **else** // Go coarser
23: $\langle \Upsilon, \mathbf{C}\rangle \leftarrow$ REACHIT$_m(\Upsilon, l + 1, \mathbf{C})$
24: **return** $\langle \Upsilon, \mathbf{C}\rangle$
25: **end if**
26: **end if**

where D is the maximum number of recursive calls of REACHIT$_m$, and C^d is the controller synthesized in recursion depth d of REACHIT$_m$.

First, it should be noted that for any state $x \in X$ for which $\mathbf{Q}(x) \cap B^d \neq \emptyset$ holds, we have: if there exists $d' \in [1; D]$ and $\hat{x} \in \mathbf{Q}(x) \cap B^d$ s.t. $\hat{R}_{l_{d'}l_d}(\hat{x}) \cap B^{d'} \neq \emptyset$ then $d' \leq d$. Hence, the quantizer $\mathbf{Q}$ formally defined via a ranking over layers l_d in Sec. 3.2 is equivalent to a quantizer defined via the ranking induced by the induction depth d.

To see why (21) holds, recall that, for any $x \in \hat{Q}_1^{-1}(B)$, every trajectory $\xi \in \mathcal{B}(\vec{S}_1^{cl})|_X$ with $\xi(0) = x$ will reach (the projection of) the target T, and that Alg. 1 will eventually reach $l = 1$. Now assume that Alg. 1 was run until depth d where $l = 1$ and let $k' \in \mathrm{dom}(\xi)$ be s.t. for all $k > k'$ with $k \in \mathrm{dom}(\xi)$, $\xi(k) \in \bigcup_{d' < d} \mathbf{Q}^{-1}(B^{d'})$ while this is not true for $\xi(k')$. Given that $l_d = 1$, we execute REACH for $l = 1$ implying that $\xi(k')$ up to $\xi(k' - m)$ will be added to the domain of controller B^d and saved in Υ. As $x \in \hat{Q}_1^{-1}(B)$, Alg. 1 can only terminate if $\hat{Q}_1(\xi(0))$ is eventually reached. Therefore, we can apply the above argument iteratively to prove the statement.

4.2 Generalized Büchi Control Problems

In this section, we discuss the generalized Büchi control problem, which is the simplest extension of the reachability control problem whose solution requires an internal controller state.

Given an abstract multi-layered system $\hat{\mathbf{S}} = (\hat{\mathbf{X}}, \hat{U}, \hat{\mathbf{F}})$, a *Generalized Büchi control problem* is the control problem $(\hat{\mathbf{S}}, \bigwedge_{a \in A} \square\lozenge T_a)$ where each T_a is interpreted as a subset of $\hat{X}_1'$ and $[\![\bigwedge_{a \in A} \square\lozenge T_a]\!]_{\hat{\mathbf{S}}}$

is defined by the set

$$\{\hat{\xi} \in \mathcal{B}(\hat{\mathbf{S}}) \mid \forall a \in A \,.\, \forall k \in \mathbb{N} \,.\, \exists k' \in \mathbb{N} \,.\, k' > k \wedge \hat{\xi}(k') \in T_a\}.$$

We assume $A = [1; N]$ for some N. If $|A| = 1$, the above control problem is usually referred to as a *Büchi control problem* and simplified to $(\hat{\mathbf{S}}, \square\lozenge T)$ with $T \subseteq \hat{X}_1'$.

Single-Layered Systems. Again, we first examine the simple case where $L = 1$ and associate $(\hat{\mathbf{S}}, \bigwedge_{a\in A} \square\lozenge T_a)$ with $(\hat{S}_1, \bigwedge_{a\in A} \square\lozenge T_a)$. Generalized Büchi control problems are solved by the help of the chained two-nested fixed point:

$$\nu \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_A \end{bmatrix} \cdot \begin{matrix} \mu W \,.\, \mathrm{Pre}_{\hat{S}_1}(W) \cup (\mathrm{Pre}_{\hat{S}_1}(V_2) \cap T_1) \\ \mu W \,.\, \mathrm{Pre}_{\hat{S}_1}(W) \cup (\mathrm{Pre}_{\hat{S}_1}(V_3) \cap T_2) \\ \vdots \\ \mu W \,.\, \mathrm{Pre}_{\hat{S}_1}(W) \cup (\mathrm{Pre}_{\hat{S}_1}(V_1) \cap T_A) \end{matrix} \tag{22}$$

For any fixed V_a, every line of (22) reduces to the reachability fixed point in (17) with target $\mathrm{Pre}_{\hat{S}_1}(V_{a^+}) \cap T_a$, where a^+ denotes $(a \mod |A|) + 1$. Hence, the fixed point in (22) is evaluated iteratively by computing

$$V_a^0 = \hat{X} \text{ and } V_a^j = \mathrm{REACH}(\mathrm{Pre}_{\hat{S}}(V_{a^+}^{j-1}) \cap T_a), \tag{23}$$

where $\mathrm{REACH}(T)$ denotes the reachability fixed point in (18) for target T, until we reach some $N \in \mathbb{N}$ s.t. $V_a^N = V_a^{N+1}$ for all $a \in A$. Based on the last call of REACH which computes V_a^N, we can construct a static controller $C_a = (B_a, \hat{U}, G_a)$ using (19) and extend it by re-defining $B_a = V_a^N$ and extending G_a s.t. $G_a(x) = \{\hat{u} \in \hat{U} \mid \hat{F}(x, \hat{u}) \subseteq V_{a^+}^{j-1}\}$ for all $x \in V_a^N \cap T_a$. To enable switching between the computed controllers, we introduce an internal controller state space $Z = \{1, .., |A|\}$ and define

$$F^c(a, x) = a \text{ if } x \notin T_a \text{ and } F^c(a, x) = a^+ \text{ if } x \in T_a.$$

If we keep the single controllers C_a after introducing the internal state Z, we get the multi-layered controller[7] $\mathbf{C} = \{Z, \{B_a\}_{a\in A}, \hat{U}, F^c, \{G_a\}_{a\in A}\}$. The usual synthesis algorithm for controllers solving $(\hat{S}_1, \bigwedge \square\lozenge T_a)$, however, returns the single (stateful) controller $C = (Z, B, \hat{U}, F^c, G)$ with $B = \bigcup_{a\in A} B_a$ and $G(x, a) = G_a(x)$. For the simple Büchi case with $|A| = 1$, the controller C becomes static again as Z reduces to a singleton.

Multi-Layered Systems. Following the idea of constructing a multi-layered controller by introducing an internal controller state which distinguishes between the different reachability controllers, we get a multi-layered controller solving the generalized Büchi control problem by calling $\mathrm{ML\_REACH}_m$ instead of REACH in (23). For simplicity, we compute the next target of $\mathrm{ML\_REACH}_m$ on the lowest layer $l = 1$, resulting in an iteration over

$$\langle V_a^j, \mathbf{C}_a \rangle = \mathrm{ML\_REACH}_m(\mathrm{Pre}_{\hat{S}_1}(V_{a^+}^{j-1}) \cap T_a). \tag{24}$$

where $V_a^0 = \hat{X}_1$. After reaching a fixed-point after N steps, $\mathbf{C}_a$ is a multi-layered controller for reaching $V_{a^+}^N \cap T_a$ for every $a \in A$. Instead of manipulating this controller, we simply add another simple controller $(B_a^p, \hat{U}, G_a^p)$ to $\mathbf{C}_a$ with $p = P_a + 1$ which steers the system from the target region T_a to the part of the state space that allows $\mathbf{C}_{a^+}$ to steer the system to the next target T_{a^+}, i.e. to $V_{a^+}^N$. This region is ensured to exist due to the properties of the fixed-point. The resulting multi-layered controller $\mathbf{C} = (Z, B, \hat{U}, F^c, G)$ with $Z = A$, $B = \bigcup_{a\in A}\{B_a^{p_a}\}_{p_a \in [1;P_a+1]}$ and $G = \bigcup_{a\in A}\{G_a^{p_a}\}_{p_a\in[1;P_a+1]}$ collects all these multi-layered controllers $\mathbf{C}_a$ by slightly abusing notation and having $|A|$ different index sets P_a. We refer to the described routine of computing $\mathbf{C}$ by $\mathrm{ML\_GBÜCHI}_m$.

When running $\mathbf{C}$ in feedback with $\hat{\mathbf{S}}$, a certain state $x \in X$ might be mapped to a different controller depending on the current target T_a. This functionality is realized by $\mathbf{Q}_a$.

Soundness and Completeness. Soundness of the resulting multi-layered controller $\mathbf{C}$ follows directly from the soundness of $\mathrm{ML\_REACH}_m$ by extending the ranking argument to the last added controller $C_a^{P_a+1}$, which ensures that the domain of the multi-layered controller $\mathbf{C}_{a^+}$ is reached. Similarly, completeness relative to $l = 1$ trivially follows from the completeness of $\mathrm{ML\_REACH}_m$, as the additional controller $C_a^{P_a+1}$ was computed on layer $l = 1$ and therefore does not require any special abstraction.

4.3 Safety

Note that the ABCS method already incorporates the possibility to restrict any control problem to a designated safe region X' of the state space $X \supset X'$ by constructing the abstract system such that it blocks for every abstract state in $\hat{X} \setminus \hat{X}'$. For the problem of reaching a target region $R \subseteq \hat{X}'$ while staying safely inside $\hat{X}'$, this simplifies to removing all abstract states $\hat{X} \setminus \hat{X}'$ before running the algorithm for reachability. This feature was implicitly used in the example of Sec. 1. However, safety control problems may arise as sub-problems in other synthesis tasks, and so we now give an algorithm for multi-layered synthesis for safety properties.

Given an abstract multi-layered system $\hat{\mathbf{S}} = (\hat{\mathbf{X}}, \hat{U}, \hat{\mathbf{F}})$, a *safety control problem* is the control problem $(\hat{\mathbf{S}}, \square T)$ where the set $T \subseteq \hat{X}_1'$ in layer 1 is associated with the multi-layered collection of safe sets $\mathbf{T} = \{\Gamma_{l1}(T)\}_{l\in[1;L]}$ and

$$[\![\square T]\!]_{\hat{\mathbf{S}}} = \{\hat{\xi} \in \mathcal{B}(\hat{\mathbf{S}}) \mid \forall k \in \mathbb{N} \,.\, \hat{\xi}(k) \in \mathbf{T}\}. \tag{25}$$

For a single-layered system, safety controllers can be synthesized by evaluating the maximal fixed-point [14]

$$\mathcal{W}[\square T] = \nu V \,.\, \mathrm{Pre}_{\hat{S}}(V) \cap T \tag{26}$$

by computing

$$V^0 = \hat{X} \text{ and } V^{i+1} = T \cap \mathrm{Pre}_{\hat{S}}(V^i) \tag{27}$$

iteratively. Since (27) is initialized by the full state set $\hat{X}$, which shrinks in every iteration, we cannot apply the same intuition as in the reachability case to obtain a construction of a multi-layered safety controller. Instead, our algorithm computes the winning set layer-by-layer. First, in layer L, we compute the fixed-point of (27). In lower layers $l < L$, we win either by staying forever in T or by reaching, through T, an already computed winning state in a higher layer. Formally, we compute the layer-wise maximal fixed-point

$$W^L = \nu V \,.\, \mathrm{Pre}_{\hat{S}_L}(V) \cap T \tag{28}$$

$$W^l = \nu V \,.\, \mathrm{Pre}_{\hat{S}_l}(V \cup \Omega_l) \cap T_l \tag{29}$$

where $\Omega_l = \cup_{l'>l} \Gamma_{ll'}(W^{l'})$ and $T_l = \Gamma_{l1}(T) \setminus \Omega_l$. The resulting winning sets define the static multi-layered controller $\mathbf{C} = \{(W^l, \hat{U}, G^l)\}_{l\in[1;L]}$ where $G^l(\hat{x}) = \{\hat{u} \in \hat{U} \mid \hat{F}_l(\hat{x}, \hat{u}) \subseteq W^{l-1} \cup \Omega_l\}$. The soundness of $\mathbf{C}$ relies on the fact that whenever Ω_l is reached,

[7] For notational simplicity we associate $G^p : Z \times \hat{X} \rightrightarrows \hat{U}$ with $G_z^p : \hat{X} \rightrightarrows \hat{U}$ for all $z \in Z$.

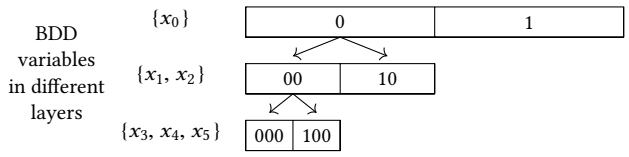

Figure 3: Arrangement of BDD variables across different layers for one state dimension and $|\hat{X}_L| = 2$.

a controller $C^{l'}$ with $l' > l$ can be used to keep the trajectory safe or move this obligation to an even higher layer. We refer to the described routine of computing $\mathbf{C}$ by ML_SAFE.

4.4 ω-regular Properties

Sec. 4.2 showed that a multi-layered controller solving the generalized Büchi control problem calls ML_REACH$_m$ repeatedly. Given the soundness of ML_REACH$_m$, soundness also follows for the generalized Büchi case. In a similar manner, one can construct synthesis algorithms for multi-layered controllers solving any ω-regular objective. The key insight is that by adding a finite number of controller states, one can reduce such objectives to parity games [7], and the fixed-point characterization of parity games alternates between safety and reachability. Specifically, we can synthesize multi-layered controllers for GR(1) objectives [1].

5 IMPLEMENTATION AND CASE STUDY

We have implemented the presented multi-layered controller synthesis algorithms in C++ as an extension to SCOTS [17]. SCOTS is a tool for ABCS using FRR. It natively supports the fixed-point computations for reachability, safety, and reach-while-avoid specifications. Like SCOTS, our implementation uses *binary decision diagrams* (BDDs); a set of abstract states $\hat{X}'$ is represented as a BDD over a set of boolean variables. An assignment of the boolean variables uniquely represents one state $\hat{x} \in \hat{X}'$.

Key to our multi-layered algorithms is the efficient implementation of the operator $\Gamma_{ll'}$ in (20). We observe: since the ratio of η-s of two adjacent layers is 2 (see Sec. 3.1), the BDD variables across different layers can be allocated to follow a particular pattern (shown in Fig. 3 for the simple case $|\hat{X}_L| = 2$). With this observation, we now describe an efficient BDD implementation for $\Gamma_{ll'}(\cdot)$ assuming an 1D state space; our implementation generalizes to any state space dimension. Let $W_l \subseteq \hat{X}_l$ be a set of states in layer l represented by the BDD $\mathcal{W}_l$ over a set of boolean variables $\{b_j, b_{j+1}, \ldots, b_k\}$. We compute the BDD representations $\mathcal{W}_{l+1} = \Gamma_{(l+1)l}(W_l)$ and $\mathcal{W}_{l-1} = \Gamma_{(l-1)l}(W_l)$ by simple transformations

$$\mathcal{W}_{l+1} = [\![\forall b_j.\mathcal{W}_l]\!][\{b_{j+1}, \ldots, b_k\}/\{b_{2j-k}, \ldots, b_{j-1}\}] \text{ and}$$
$$\mathcal{W}_{l-1} = [\![\exists \mathcal{W}_l]\!][\{b_j, \ldots, b_k\}/\{b_{k+2}, \ldots, b_{2k-j+2}\}],$$

where $[\![\mathcal{B}]\!][\{x_1, \ldots, x_p\}/\{y_1, \ldots, y_p\}]$ represents the renaming of the variables of the BDD $\mathcal{B}$ from $\{x_1, \ldots, x_p\}$ to $\{y_1, \ldots, y_p\}$. Recall that BDD renaming is linear in the size of the BDD.

5.1 Performance Evaluation

In this section, we compare the performance of our multi-layered ABCS algorithms to their single-layered counterparts. All computation times presented in this section were obtained by running the respective algorithms on an Intel Core i5 2.7 GHz processor.

Table 1: REACH vs. ML_REACH$_6$

	Runtime (s)		
Number of layers	$L = 1$	$L = 2$	$L = 3$
Abstraction	2200	2400	2420
Synthesis	860	139	113
Total	3060	2540	2530
	(100%)	(83.0%)	(82.7%)

Table 2: SAFE vs. ML_SAFE

	Runtime (s)		
Number of layers	$L = 1$	$L = 2$	$L = 3$
Abstraction	21.2	27.0	29.2
Synthesis	203	14.3	7.25
Total	224	41.3	36.5
	(100%)	(18.4%)	(16.3%)

Table 3: GBÜCHI vs. ML_GBÜCHI$_5$

	Runtime (s)		
Number of layers	$L = 1$	$L = 2$	$L = 3$
Abstraction	2210	2410	2460
Synthesis	3370	568	444
Total	5580	2970	2900
	(100%)	(53.2%)	(52.0%)

Vehicle Motion Planning. Consider an autonomous vehicle whose dynamics are given by a *unicycle model*. This results in a nonlinear control system with three state variables representing the position, speed and the steering angle of the vehicle.

Three abstraction layers were constructed. The restricted state space $X' = [0, 11.4]\times[0, 11.4]\times[-\pi-0.4, \pi+0.4]$ was discretized by grid parameters $\eta_1 = [0.15, 0.15, 0.075]^T$, $\eta_2 = [0.3, 0.3, 0.15]^T$, and $\eta_3 = [0.6, 0.6, 0.3]^T$. The abstract input space was $\hat{U} = \{u_1 \mid u_1 = -1.05+0.3\mu_1, \mu_1 \in [0; 7]\}\times\{u_2 \mid u_2 = -1.5+0.2\mu_2, \mu_2 \in [0; 15]\}$. The dynamics were sampled using sampling times $\tau_1 = 0.225$, $\tau_2 = 0.45$, and $\tau_3 = 0.9$ and a perturbation bound $w = [0.05, 0.05, 0]^T$. The 2D workspace depicted in Fig. 4 shows the first two components of X' with obstacles (gray) and a target (red).

We first compare the single- and multi-layered reachability algorithms (REACH and ML_REACH$_m$ with $m = 6$); see Table 1. We observe that it takes slightly more time to compute two and three abstractions (first line, second and third column) than it does to compute only $\hat{S}_1$ (first line, first column). However, multi-layered controller synthesis results in significant savings over single-layered synthesis (second line, second and third column compared to the first). Therefore, ML_REACH$_6$ with $L > 1$ outperforms REACH (i.e., ML_REACH with $L = 1$) on this particular reachability control problem. Fig. 4 shows (the 2D projection of) the domains of all local controllers C^p contained in $\mathbf{C}$ resulting from ML_REACH$_6$ run for $L = 3$. We see that layer 1 is required to traverse narrow passages or to approach the small target, whereas coarser abstractions are used in more open parts of the state space.

Secondly, we compare the results of GBÜCHI and ML_BÜCHI$_m$ with $m = 5$ run over the same workspace but with three spatially distributed targets; see Table 3. Computation times for the abstractions are almost equivalent to the ones in Table 1. As expected, the

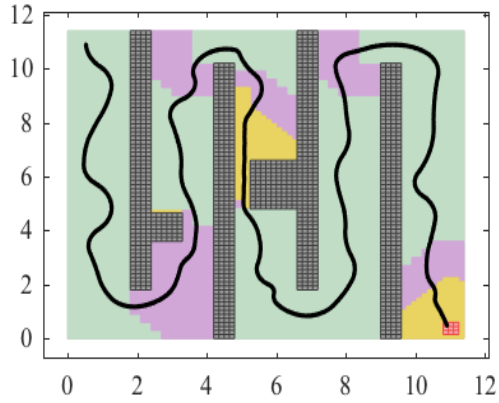

Figure 4: Vehicle workspace projected to $\mathbb{R}^2$. Local controller domains synthesized in layers l_1, l_2, and l_3 are resp. yellow, purple, and green. Obstacle and target cells in l_1 are resp. gray and red. The black wavy line is the vehicle's trajectory.

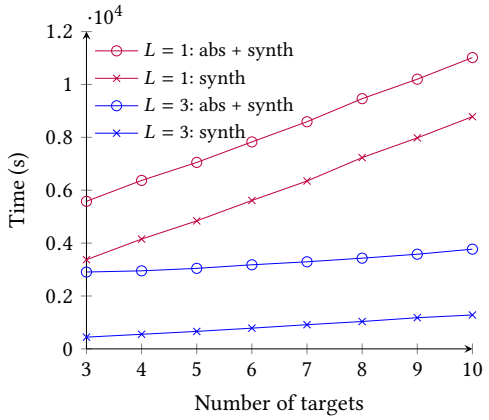

Figure 5: Performance of GBÜCHI ($L = 1$) and ML_GBÜCHI$_5$ ($L = 3$).

savings in controller synthesis time are higher than in the simple reachability case, as an equally complex reachability control problem has to be solved at least three (but in general, more than three) times for the generalized Büchi objective considered.

We have also performed a comparative study between the execution times of GBÜCHI and ML_GBÜCHI$_m$ with $m = 5$ while changing the number of targets (Fig. 5). We see that ML_GBÜCHI scales much better than GBÜCHI since the abstraction construction times stay constant while the synthesis savings build with each added target.

Boost DC-DC Power Converter. Consider a boost DC-DC converter from [11], Sec. V.A. This converter is modeled as a switched system with two modes, a two dimensional state (consisting of the inductor current and the capacitor voltage), and an affine transition function for each mode.

Three abstraction layers were constructed. The restricted state space $X' = [1.15, 1.55] \times [5.45, 5.85]$ was discretized by grid parameters $\eta_1 = [0.0005, 0.0005]^T$, $\eta_2 = [0.001, 0.001]^T$, and $\eta_3 = [0.002, 0.002]^T$. The abstract input space was $\hat{U} = \{0, 1\}$ representing the activation of the two modes. The dynamics were sampled using sampling time $\tau = 0.5$ and a perturbation bound $w = [0.05, 0.05, 0]^T$ for all layers.

Table 2 shows the performance comparison between the single-layered safety algorithm for S_1 (first column) and the multi-layered algorithm for two and three abstractions; the improvement of the overall performance is substantial.

Parameter choices. Tuning parameters η_1, τ_1, L, and m are chosen by the user. While the algorithms are sound for any choice, optimality of these parameters is not fully determined by the system dynamics but also depends on the specification. For example, in the unicycle example, η_1 must be chosen such that it allows for passing narrow passages (to achieve large coverage) and L can be chosen s.t. there exists an open space which fits boxes of size η_L.

6 CONCLUSION

We have presented a method to use multiple abstractions during abstract controller synthesis. The resulting abstract multi-layered controller induces a quantizer which enables the application of this controller to the underlying concrete system. Empirically, our synthesis algorithm runs faster than standard (single-layered) algorithms on various control problems.

ACKNOWLEDGMENTS

This research was sponsored in part by the ERC Synergy award IMPACT. Kyle Hsu was supported by the DAAD RISE Fellowship.

REFERENCES

[1] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911 – 938, 2012.
[2] J. Bradfield and C. Stirling. Modal mu-calculi. In *The Handbook of Modal Logic*, pages 721––756. Elsevier, 2006.
[3] J. Cámara, A. Girard, and G. Gössler. Safety controller synthesis for switched systems using multi-scale symbolic models. In *CDC '11*, pages 520–525, 2011.
[4] J. Cámara, A. Girard, and G. Gössler. Synthesis of switching controllers using approximately bisimilar multiscale abstractions. In *HSCC*, pages 191–200, 2011.
[5] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV 2000*, pages 154–169. Springer, 2000.
[6] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252. ACM, 1977.
[7] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS'91*, pages 368–377, 1991.
[8] S. Esmaeil Zadeh Soudjani and A. Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956, 2013.
[9] A. Girard. Towards a multiresolution approach to linear control. *TAC*, 51(8):1261–1270, 2006.
[10] A. Girard, G. Gössler, and S. Mouelhi. Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *TAC*, 61(6):1537–1549, 2016.
[11] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *TAC*, 55(1):116–126, 2010.
[12] T. A. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *ICALP*, pages 886–902. Springer, 2003.
[13] S. R. Lindemann and S. M. LaValle. Multiresolution approach for motion planning under differential constraints. In *ICRA*, pages 139–144. IEEE, 2006.
[14] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS'95*, volume 900 of *LNCS*, pages 229–242. Springer, 1995.
[15] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992.
[16] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *TAC*, 62(4):1781–1796, 2017.
[17] M. Rungger and M. Zamani. SCOTS: A tool for the synthesis of symbolic controllers. In *HSCC'16*, pages 99–104. ACM, 2016.
[18] P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer, 2009.
[19] Y. Tazaki and J.-i. Imura. Discrete-state abstractions of nonlinear systems using multi-resolution quantizer. In *HSCC*, pages 351–365. Springer, 2009.
[20] A. Weber, M. Rungger, and G. Reissig. Optimized state space grids for abstractions. *TAC*, 2016.