2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

A trajectory-based stochastic approach to symbolic control

Alessandro Tenaglia, Corrado Possieri and Daniele Carnevale

Abstract— **This paper presents two innovative approaches to design symbolic controllers for dynamical systems. The first novelty involves a new trajectory-based strategy for defining the states of a symbolic model, which provides a more accurate representation of the system's dynamics than the traditional grid-based technique. The second novelty concerns using a Bounded-parameter Markov Decision Process rather than a Finite Transition System to model the behavior of a symbolic model. This procedure allows for handling the system's stochastic behavior and considers uncertainties. The effectiveness of the novel approaches presented is demonstrated through numerical results.**

Index Terms— **Automata, Markov Processes, Machine Learning, Symbolic Control, Finite Transition Systems.**

I. Introduction

Symbolic control refers to a class of control methods that rely on symbolic models, such as finite transition systems or automata, to design controllers for dynamic systems. The fundamental idea behind symbolic control is to find a bisimulation relation between the models that share the same properties. The seminal papers [1] and [2] first introduced this concept in the 1980s. It has since become a popular method for controlling complex systems in computer science and robotics due to its ability to handle systems with large state spaces and provide formal guarantees on the controller's behavior. After several successful results on finite bisimulations for control systems [3]–[5], the idea of approximate bisimulation has been introduced [6], which captures the equivalence of methods approximately. In [7], it has been shown that every incrementally globally asymptotically stable nonlinear control system is approximately bisimilar to a symbolic model with a precision that can be chosen a priori. Moreover, in [8], the stability assumptions have been relaxed in favor of the incremental forward completeness property. In recent years, there has been a significant increase in attention and interest in Reinforcement Learning (RL). RL techniques are a type of machine learning in which an agent is trained to interact with an environment, learning from its experiences to maximize a cumulative reward. The application of RL methods in symbolic control has been demonstrated in [9].

Current techniques for symbolic control of dynamic systems mainly rely on grid-based partitioning of the state space, where each grid cell is associated with a discrete state. Moreover, the behavior of the symbolic model is described by transition systems without probability interpretation.

Novel contribution. The main novelties proposed in this paper consist of the following:

- defining a symbolic system using a trajectory-based approach to partition the state space;
- modeling the symbolic system behavior through a Bounded-parameter Markov Decision Process.

The rest of the paper is structured as follows. Section II provides some preliminary definitions, followed by a description of the problem of interest in Section III. Section IV presents two techniques for partitioning the state space. In Section V, state transitions are modeled using different frameworks, and for each, an algorithm is provided to compute the optimal policy. In Section VI, numerical results are presented to support the validity of the proposed approaches. Lastly, Section VII draws conclusions and future research directions.

Notation. Given a vector $x \in \mathbb{R}^n$ we denote by x_i the i-th element of x, by $|x_i|$ its absolute value, and by $\|x\|$ the infinity norm of x: $\|x\| = \max\{|x_1|, \dots, |x_n|\}$. Given a measurable function $f : \mathbb{R}_0^+ \to \mathbb{R}$, it is denoted by $\|f\|$ the (essential) supremum of f. Given a set $A \in \mathbb{R}^n$, we denote by ∂A the boundary of A and by $|A|$ its area. For any $A \in \mathbb{R}^n$ and $\mu \in \mathbb{R}$, let $[A]_\mu = \{a \in A | a_i = k_i\mu, k_i \in \mathbb{Z}, i = 1, \dots, n\}$.

II. Preliminary definitions

One of the most simple approaches to represent a symbolic model is through a *Finite Transition System* (FTS), [10].

Definition 1 (FTS): An FTS is a tuple $T = (\mathcal{S}, \mathcal{A}, \longrightarrow, R)$ where: $\mathcal{S}$ is a finite set of states, with cardinality $|\mathcal{S}|$; $\mathcal{A}$ is a finite set of actions, with cardinality $|\mathcal{A}|$; $\longrightarrow \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a finite set of admissible transitions; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function that assigns a scalar reward $R(s, a)$ to each state-action pair.

If for any state $s \in \mathcal{S}$ and any action $a \in \mathcal{A}$ there exists a unique next state s' such that $(s, a, s') \in \longrightarrow$, then the FTS is *deterministic*; otherwise, it is *non-deterministic*. In the latter case, given a state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, there are multiple possible next states $s' \in \mathcal{S}$ to which the environment could transition. If it is possible to associate a probability to their occurrence, then the FTS is *probabilistic*. In this case, the FTS can be equivalently modeled as a *Markov Decision Process* (MDP) [11], [12].

This framework requires exact knowledge of the probability for all transitions, which is difficult to obtain in practice due to uncertainties and inaccuracies. Therefore, we introduce a generalization of an exact MDP called *Bounded-parameter*

This work has been partially supported by the Italian Ministry for Research in the framework of the 2020 Program for Research Projects of National Interest (PRIN). Grant No. 2020RTWES4.

A. Tenaglia, C. Possieri, and D. Carnevale are with the Department of Civil Engineering and Computer Science Engineering, University of Rome Tor Vergata, 00133 Rome, Italy. C. Possieri is also with the Institute for System Analysis and Computer Science "A. Ruberti", National Research Council of Italy, 00185 Rome, Italy. E-mails: {alessandro.tenaglia, corrado.possieri, daniele.carnevale}@uniroma2.it.

Markov Decision Process (BMDP) that enables accounting for uncertainties in the underlying model parameters [13].

Definition 2 (BMDP): A BMDP is a tuple $M_{\updownarrow} = (\mathcal{S}, \mathcal{A}, P_{\updownarrow}, R)$ where: $\mathcal{S}$ is a finite set of states, with cardinality $|\mathcal{S}|$; $\mathcal{A}$ is a finite set of actions, with cardinality $|\mathcal{A}|$; $P_{\updownarrow}$ is an interval transition distribution that maps the probability of transition from a state $s \in \mathcal{S}$ to a state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ to a closed interval $[P_{\downarrow}(s,a,s'), P_{\uparrow}(s,a,s')]$. To guarantee properly structured transitions, for any state $s \in \mathcal{S}$ and any action $a \in \mathcal{A}$ the sum of the lower and upper bounds of $P_{\updownarrow}(s,a,s')$ on all possible next states $s' \in \mathcal{S}$ must be less than or equal to 1 and greater than or equal to 1, respectively; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ maps each pair $(s,a) \in \mathcal{S} \times \mathcal{A}$ to a reward $R(s,a) = \mathbb{E}[R_t = r | S_t = s, A_t = a]$.

In a decision-making framework, at each time step, starting from state $S_t \in \mathcal{S}$, the agent chooses an action $A_t \in \mathcal{A}$ based on a policy $\pi : \mathcal{S} \to \mathcal{A}$, which maps states to actions, then the environment transitions to a new state $S_{t+1} \in \mathcal{S}$ and the agent receives a reward R_{t+1}. The corresponding return G_t is defined as $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots$, where $\gamma \in [0,1)$ is called *discount rate*. The goal is to find the optimal policy $\pi_\star$ that maximizes the expected cumulative discounted reward for all possible starting states. The tool used to quantify the goodness of a policy π is the value function $V_\pi : \mathcal{S} \to \mathbb{R}$, which is a mapping from each possible state $s \in \mathcal{S}$ to the expected cumulative reward that can be obtained starting from that state s and following a policy π, namely, $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$. The *optimal value function* $V_\star$ is defined as $V_\star(s) = \max_{\pi \in \Pi} V_\pi(s)$, where Π is the set of all deterministic policies. Hence, the *optimal policy* $\pi_\star$ defines the actions that lead to the highest expected cumulative reward. The value function $V_\star$ is greater than or equal to any value function V_π in the partial order $\geq$ defined as follows: $V_1 \geq V_2$ if and only if $V_1(s) \geq V_2(s)$ for all states $s \in \mathcal{S}$.

In the case of BMDP, uncertainties concerning transitions do not allow the calculation of an exact value function. Thus, it is defined an *interval value function* $V_{\updownarrow}$ that maps each state to a closed interval, with $V_{\updownarrow}(s) = [V_{\downarrow}(s), V_{\uparrow}(s)]$. Given a policy π, the relative *interval value function* is defined by $V_{\updownarrow_\pi}(s) = [\min_{M \in M_{\updownarrow}} V_{M,\pi}(s), \max_{M \in M_{\updownarrow}} V_{M,\pi}(s)]$ and the *optimal interval value function* as $V_{\updownarrow_\star} = \max_{\pi \in \Pi}(V_{\updownarrow_\pi})$, where the partial order $\geq$ is extended in the case of closed intervals in an *optimistic* way, that is, $V_{\updownarrow_1} \geq_{opt} V_{\updownarrow_2}$ if and only if $[V_{\downarrow_1}(s), V_{\uparrow_1}(s)] \geq_{opt} [V_{\downarrow_2}(s), V_{\uparrow_2}(s)]$ for all states $s \in \mathcal{S}$, namely $V_{\uparrow_1}(s) > V_{\uparrow_2}(s)$, or $V_{\uparrow_1}(s) = V_{\uparrow_2}(s)$ and $V_{\downarrow_1}(s) > V_{\downarrow_2}(s)$.

What is described for BMDP also applies to *non-deterministic* FTS, since for a given policy π multiple transitions can occur, making it impossible to compute an exact value function V. Therefore, two distinct value functions are introduced, a *minimal value function* $\underline{V}_\pi$ and a *maximal value function* $\overline{V}_\pi$, which describe, respectively, the expected cumulative reward in the worst-case and the best-case scenario following the policy π. Similarly, the *minimally* and the *maximally optimal value functions*, $\underline{V}_\star$ and $\overline{V}_\star$, are defined, to which correspond, respectively, the *minimally optimal policy* $\underline{\pi}_\star$ and the *maximally optimal policy* $\overline{\pi}_\star$.

III. Problem statement

In this paper, we consider the following class of systems

$$\Sigma : \begin{cases} \dot{x} = f(x,u), \\ x \in X \subseteq \mathbb{R}^n, u \in U \subseteq \mathbb{R}^m, \end{cases} \tag{1}$$

where X is the state space and U is the input set, and $f : \mathbb{R}^n \times U \to \mathbb{R}^n$ is a continuous map that satisfies the following classical Lipschitz assumption, which ensures uniqueness of solutions to system (1).

Assumption 1: For every compact set $W \subset \mathbb{R}^n$, there exists a constant $L \in \mathbb{R}^+$ such that

$$\|f(x,u) - f(y,u)\| \leq L\|x - y\|$$

for all $x, y \in W$ and all $u \in U$.

Moreover, the results presented in this paper require the notion of incrementally forward completeness (δ-FC) and incrementally input-to-state stability (δ-ISS) given in [8].

We denoted by $\mathcal{U}$ all the measurable functions from intervals of the form $]a,b[\subset \mathbb{R}$ to U with $a < 0$ and $b > 0$. A curve $\mathbf{x} :]a,b[\to \mathbb{R}^n$ is said to be a *trajectory* of (1) if there exists a $\mathbf{u} \in \mathcal{U}$ satisfying $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$, for almost all $t \in]a,b[$. Let $\mathbf{x}(t,x,\mathbf{u})$ be the state reached at time t with input $\mathbf{u}$ from the initial condition x. There exists a unique solution of (1) by Assumption 1 [14].

Since the attention of this paper is on digital control, given a $\tau \in \mathbb{R}^+$, it is defined by $\mathcal{U}_\tau$ the set of all constant curves of duration τ, i.e., $\mathcal{U}_\tau = \{\mathbf{u} : [0,\tau] \to U \mid \mathbf{u}(t) = \mathbf{u}(0), \forall t \in [0,\tau]\}$, and we consider trajectories $\mathbf{x} : [0,\tau] \to \mathbb{R}^n$ defined on the closed domain $[0,\tau]$, with the understanding of the existence of a trajectory $\mathbf{x}' :]a,b[\to \mathbb{R}^n$ such that $\mathbf{x} = \mathbf{x}'|_{[0,\tau]}$.

Supposing that Assumption 1 hold, the problem addressed in this paper concerns the design of a controller that from any initial state $x \in X$ and using the control inputs in $\mathcal{U}_\tau$ can drive (1) to a desired state $x^\star \in X$.

This work focuses on modeling a dynamic system using a transition system. In the former, the evolution of a state over time is described by differential equations. The latter is a discrete-time system whose behavior is modeled as a sequence of states and transitions between them according to the actions taken. By using a transition system, the problem of designing a controller for the dynamic system can be formulated as a decision-making problem.

IV. State space partitioning

This section addresses the challenge of partitioning the state space into a finite set of states $\mathcal{S}$. First, we introduce the notion of quantization embedding, proposed in [15]. A *mesh* $\mathcal{M}$ is a finite collection of pairs represented as

$$\mathcal{M} = \{\{\xi_0, C_0\}, \{\xi_1, C_1\}, ..., \{\xi_{|\mathcal{S}|}, C_{|\mathcal{S}|}\}\}$$

where each C_s is a *cell* of the mesh $\mathcal{M}$, each ξ_s is called the *discrete point* of the s-th *cell* C_s, the pair $\{\xi_s, C_s\}$ defines a *state* $s \in \mathcal{S}$, and moreover the set $C = \{C_0, C_1, ..., C_{|\mathcal{S}|}\}$ forms a partition of X. The mesh $\mathcal{M}$ defines a quantization function $Q_\mathcal{M} : X \to \{\xi_0, \xi_1, \ldots, \xi_{|\mathcal{S}|}\}$ that maps an arbitrary point of the state space $x \in X$ to a discrete

point ξ_s whose corresponding cell C_s includes x, formally $Q_{\mathcal{M}}(x) = \xi_s \quad \text{if } x \in C_s$. The quantization function allows the conversion of continuous states $x \in X$ into discrete states $s \in \mathcal{S}$.

A. Grid-based approach

The most simple approach is to divide the state space X into a grid with resolution μ chosen according to the desired level of precision. Formally, $\mathcal{S} = [X]_\mu$, where $[\cdot]_\mu$ is the quantizer $Q_{\mathcal{M}}$. In this way, each cell C_s of the grid represents a discrete state of $\mathcal{S}$, and the centroid ξ_s is its representative point. In this partitioning, given a state $x \in X$, the quantizer $Q_{\mathcal{M}}$ compares its coordinates with the cell boundaries and determines to which cell the state belongs, namely $Q_{\mathcal{M}}(x) = \xi_s$ such that

$$\|x - \xi_s\| \leq \frac{\mu}{2} = \delta_\mu. \tag{2}$$

This grid-based approach provides a standard representation of the state space that is simple to manage and implement. However, large state spaces may suffer from the curse of dimensionality as the number of states grows exponentially to obtain an accurate system representation. More exhaustive details are reported in [8].

B. Trajectory-based approach

The novel approach we present is based on the idea that the trajectories of the system provide a natural way to partition the state space. Specifically, we propose to populate the state space with the system's trajectories and use the resulting partitions as the mesh $\mathcal{M}$ that defines the symbolic model. The essential advantage of this technique is that it can precisely represent the system's behavior, which may not be possible using an agnostic grid-based partitioning approach, where the shape of the cells is fixed and predetermined.

Algorithm 1 describes in detail the sequential steps required to execute the trajectory-based partitioning. The approach is based on iteratively branching and simulating trajectories forward and backward in time to explore the whole state space. Since the goal is to steer the system's state to $x^\star \in X$, this is the natural starting point to simulate the system's evolution. From $x^\star$, a new set of trajectories is generated based on control inputs $u \in \mathcal{U}$. The branching is performed so that, given an existing trajectory $\mathbf{x}$, if its length is greater than μ, it is divided at its midpoint $\bar{x}$ into two sub-trajectories $\mathbf{x}_1$ and $\mathbf{x}_2$. Each time a new trajectory is branched from an existing one, starting from $\bar{x}$, its branching count is incremented. The trajectories to be evaluated are sorted in increasing order according to the branching count. Using a priority queue $\mathcal{Q}$ ensures that the trajectories generated with fewer branches are processed first, and the state space is explored more efficiently. When two trajectories intersect, each is divided into two sub-trajectories at the intersection points $\tilde{x}$. This process is repeated until the priority queue is empty, that is, the length of all the trajectories is less than μ.

Upon completion of this process, it becomes evident that the set of generated trajectories provides a partition of the state space X. Therefore, the boundary of each cell C_s is determined by portions of the system's trajectories. It is worth noticing that, unlike the grid-based approach, the cells obtained following this trajectory-based partitioning do not have fixed and regular shapes but are variable and not necessarily convex.

Algorithm 1: Trajectory-based partitioning

input : a starting state $x^\star$, a threshold μ
output : a mesh $\mathcal{M}$

initialize a set $\mathcal{T}$ to store trajectories and a priority queue $\mathcal{Q}$ for those to be evaluated
foreach $a \in \mathcal{A}$ **do**
  $\mathbf{x} \leftarrow$ gen_traj($x^\star$, a, 0)
  ($\mathcal{T}$, $\mathcal{Q}$) $\leftarrow$ add_traj($\mathbf{x}$, $\mathcal{T}$, $\mathcal{Q}$);
repeat
  $(\mathbf{x}, p) \leftarrow$ pop the first trajectory of $\mathcal{Q}$
  $\bar{x} \leftarrow$ middle point of $\mathbf{x}$
  ($\mathcal{T}$, $\mathcal{Q}$) $\leftarrow$ split_traj($\mathbf{x}$, $\bar{x}$, $\mathcal{T}$, $\mathcal{Q}$)
  foreach $a \in \mathcal{A}$ **do**
    $\mathbf{x}' \leftarrow$ gen_traj($\bar{x}$, a, $p+1$)
    foreach $\mathbf{x} \in \mathcal{T}$ **do**
      if $\mathbf{x}$ *intersects* $\mathbf{x}'$ **then**
        $\tilde{x} \leftarrow$ intersection points of $\mathbf{x}$ and $\mathbf{x}'$
        ($\mathcal{T}$, $\mathcal{Q}$) $\leftarrow$ split_traj($\mathbf{x}$, $\tilde{x}$, $\mathcal{T}$, $\mathcal{Q}$)
        ($\mathcal{T}$, $\mathcal{Q}$) $\leftarrow$ split_traj($\mathbf{x}'$, $\tilde{x}$, $\mathcal{T}$, $\mathcal{Q}$)
until $\mathcal{Q}$ *is not empty*
define the cells of C according to the trajectories of $\mathcal{T}$
foreach $C_s \in C$ **do**
  $\xi_s \leftarrow$ point of inaccessibility of C_s
return $\mathcal{M} = \{\{\xi_0, C_0\}, ..., \{\xi_{|\mathcal{S}|}, C_{|\mathcal{S}|}\}$

Function *gen_traj(x, a, p)*:
  $T \leftarrow [-\min_{\mathbf{x}(-t,x,a)\notin X} t, \min_{\mathbf{x}(t,x,a)\notin X} t]$
  return $\{\mathbf{x}(t, x, a) : t \in T\}$ with priority p

Function *add_traj($\mathbf{x}$, $\mathcal{T}$, $\mathcal{Q}$)*:
  $\mathcal{T}$.insert($\mathbf{x}$)
  if *length*($\mathbf{x}$) $> \mu$ **then** $\mathcal{Q}$.insert($\mathbf{x}$)
  return $\mathcal{T}$, $\mathcal{Q}$

Function *split_traj($\mathbf{x}$, x, $\mathcal{T}$, $\mathcal{Q}$)*:
  $(\mathbf{x}_1, \mathbf{x}_2) \leftarrow$ split $\mathbf{x}$ at x
  ($\mathcal{T}$, $\mathcal{Q}$) $\leftarrow$ add_traj($\mathbf{x}_1$, $\mathcal{T}$, $\mathcal{Q}$)
  ($\mathcal{T}$, $\mathcal{Q}$) $\leftarrow$ add_traj($\mathbf{x}_2$, $\mathcal{T}$, $\mathcal{Q}$)
  return $\mathcal{T}$, $\mathcal{Q}$

Without guaranteeing the convexity property of the cells, the centroid is not used as their reference point ξ_s, as in non-convex shapes, it can fall outside of them. Thus, it is chosen as the point farthest from its boundary, called *Chebyshev center*, as in [16], formally,

$$\xi_s = \arg\max_{x \in C_s} \min_{y \in \partial C_s} \|x - y\|. \tag{3}$$

Similarly to the grid-based approach, the quantization function $Q_{\mathcal{M}}$ converts continuous states $x \in X$ to discrete states ξ_s considering the cell C_s to which the state belongs. Verifying if a point is inside a convex shape is a straightforward problem,

in contrast, a non-convex shape can have one or more concave regions or holes, making it more challenging to solve. One of the most common algorithms to solve this problem is the ray-casting method, as described in [17].

Thus, it is possible to state that for each state $s \in \mathcal{S}$

$$\|x - \xi_s\| \leq \delta_\mu \quad \forall x \in C_s, \tag{4}$$

where δ_μ is defined as $\delta_\mu = \max_{s \in \mathcal{S}} \max_{y \in \partial C_s} \|\xi_s - y\|$.

Note that in the worst-case scenario, a cell C_s is defined by at most η trajectories that have a length less than or equal to μ each, yielding a boundary of length $p \leq \eta\mu$.

Proposition 1: Let Assumption 1 hold and consider system (1). If the number of trajectories that constitute the boundary of cell C_s is lower than $\eta^\star > 0$ for all $s \in \mathcal{S}$, then $\lim_{\mu \to 0} \delta_\mu = 0$.

Although more computationally expensive, trajectory-based partitioning has several advantages over the traditional grid-based approach. Firstly, it allows for a more flexible state space partitioning that can more accurately describe system dynamics, even if complex and non-linear. Moreover, it reduces the number of cells because the number of partitions required to capture the dynamics of a system using an agnostic grid-based approach can be prohibitively large.

V. Definition of transitions and optimal policy

This section involves modeling the system's behavior, that is, defining the transitions between the set of states $\mathcal{S}$ based on the set of actions $\mathcal{A} = \mathcal{U}_\tau$, regardless of the state space partitioning. The procedure followed to achieve such an objective consists of generating, for each state s, namely, for each cell C_s of the mesh $\mathcal{M}$, a set of samples $\mathcal{X}_s \subset C_s$. Then, the evolution of the system (1) from each of the samples belonging to $\mathcal{X}_s$ is simulated for a finite time τ with each action $a \in \mathcal{A}$. The next state $s' \in \mathcal{S}$ is calculated according to the quantizer $Q_\mathcal{M}$, namely, $s' = Q_\mathcal{M}(\mathbf{x}(\tau, x, a))$, $x \in \mathcal{X}_s$ and the transition $s \xrightarrow{a} s'$ is determined.

Theorem 1: Consider system (1), let Assumption 1 hold, let $\mu > 0$ and $\tau > 0$ be given. If system (1) is δ-FC, it is possible to build a symbolic model with parameters (μ, τ) such that, considering a transition $s \xrightarrow{a} s'$ the following inequality holds

$$\|\mathbf{x}(\tau, \xi_s, a) - \xi_{s'}\| \leq \beta(\delta_\mu, \tau) + \delta_\mu, \tag{5}$$

where β is the function that satisfies the δ-FC condition for the system.

Corollary 1: Consider system (1) and let Assumption 1 hold. If system (1) is δ-ISS, then for any $\varepsilon > 0$, there exist $\mu > 0$ and time $T > 0$ such that it is possible to build a symbolic model with parameters (μ, τ), with $\tau > T$, where each transition $s \xrightarrow{a} s'$ satisfies the following inequality

$$\|\mathbf{x}(\tau, \xi_s, a) - \xi_{s'}\| \leq \varepsilon. \tag{6}$$

It is worth pointing out that estimating transitions using analytic approaches is a complex and challenging problem, particularly in large and highly nonlinear systems. This work uses a data-driven technique based on Monte Carlo methods. Therefore, the quality of the samples is a crucial aspect to consider, as it impacts the accuracy of the results. Since states are defined as a partition of X and each sample corresponds to a specific state $x \in X$, it is essential to ensure that the sampling process covers the entire state space to represent the whole system. The generation of samplings $\mathcal{X}_s$ is performed using the Poisson disk sampling method, as described in [18], which ensures efficiently obtaining a set of points uniformly distributed and without clustering.

A. Finite Transition System

In the case of an FTS, each time a new state is reached, the corresponding transition $s \xrightarrow{a} s'$ is added to the set of transitions $\longrightarrow$. In general, it is a *non-deterministic* FTS because given a pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, there can be multiple possible transitions. Therefore, since it is impossible to determine the next state with certainty, calculating an exact value function is not feasible. Two value functions related to the worst-case and best-case scenarios are defined, namely $\underline{V}$ and $\overline{V}$. [9] provides a detailed description of how to modify the classic Value Iteration algorithm, as in [12], to obtain the *minimally* and *maximally optimal value function*, namely $\underline{V}_\star$ and $\overline{V}_\star$.

B. Markov Decision Process

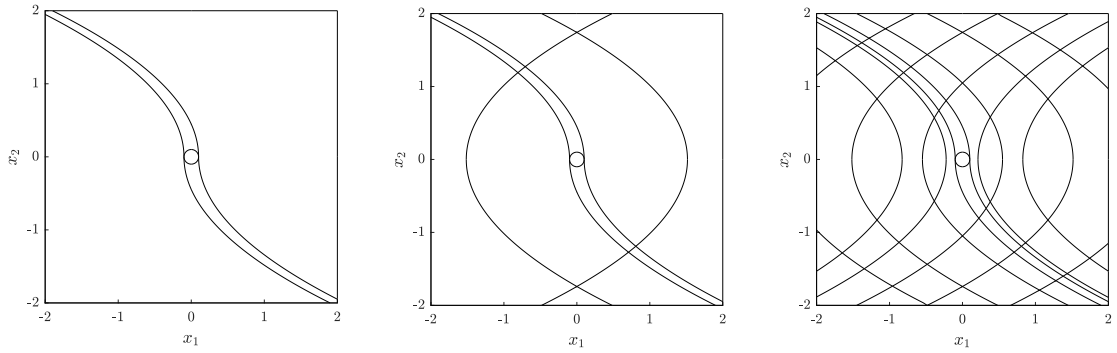
Treating all transitions as equally probable could lead to erroneous actions since outliers can affect the behavior of the entire state. The novelty proposed in this work is to estimate the probability of the occurrence of the transitions and model the symbolic system through an MDP. The most straightforward method is to count the number of times a transition to a state occurs for all state-action pairs and divide it by the total number of samples. Repeating this procedure for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, we can estimate the transition probability matrix P and then model an MDP. Once an MDP has been modeled, the *optimal value function* $V_\star$ and the *optimal policy* $\pi_\star$ can be found using the classic Value Iteration algorithm [12].

C. Bounded-parameter Markov Decision Process

When probabilities are calculated based on a limited number of samples, the related estimates have inherent uncertainty. To mitigate this problem, a confidence interval CI is calculated for the probabilities of the transitions, providing a range of values likely to include the true value with a certain confidence level. The probability distribution is modeled as a multinomial distribution, which, for large values of n, can be approximated by a normal distribution. Hence, the confidence interval CI is computed as

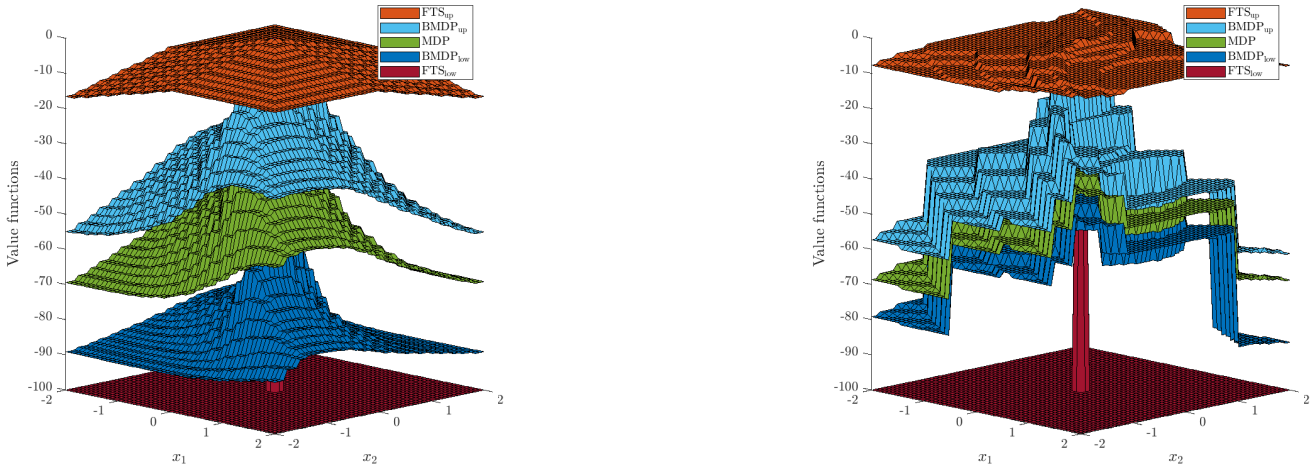
$$CI = \hat{p} \pm z^\star \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}}, \tag{7}$$

where $\hat{p}$ is the probability estimate, and $z^\star$ is the critical value of the standard normal distribution corresponding to the desired confidence level. As explained previously, a BMDP is based on *interval value functions* $V_\updownarrow$ so to find the *optimal value interval function* $V_{\updownarrow_\star}$ it is necessary to adapt the Value Iteration algorithm, as described in [13].



(a) Initial stage. (b) Second stage. (c) Final stage.

Fig. 1: Evolution of the state space partitioning using the trajectory-based approach.



(a) Value functions for grid-based partitioning. (b) Value functions for trajectory-based partitioning.

Fig. 2: Comparison of the value function obtained with FTS, MDP, and BMDP defined for grid-based and trajectory-based state space partitioning.

VI. Numerical Simulations

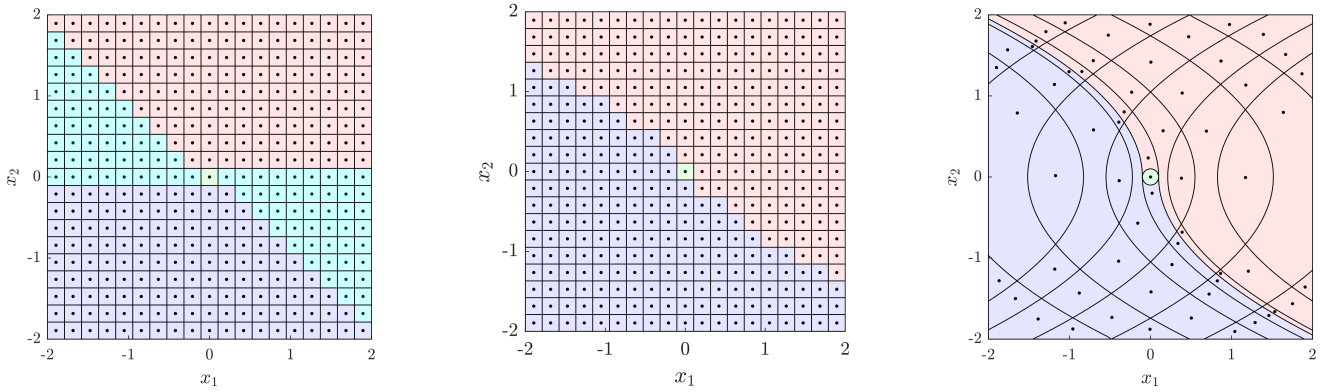
This section compares combinations of partitioning approaches and modeling frameworks through numerical simulations. The tests have been performed on the double integrator model, governed by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u. \tag{8}$$

The state set is given as $X = [-2, 2] \times [-2, 2]$, and the input set is limited to a binary control $u \in U = \{-1, 1\}$. The goal is to control the system from a generic initial state $x_0 \in X$ to $x^\star = [0, 0]'$.

Following the approach described in Section IV, two state space partitions are considered. In the case of the grid-based approach, the cell size used is $\mu = 0.2$ on each dimension, resulting in a grid of 19×19 states. In the case of the trajectory-based approach, the length threshold used for branching is $\mu = 1$. Figure 1 shows the evolution of partitioning at different stages of the algorithm. The first panel of Fig. 1a illustrates the initialization phase, where the starting point is simulated only backward in time and is embedded in an ad hoc cell to adjust the final error bound. The second panel of Fig. 1b shows the branching of trajectories as Algorithm 1 progresses. New cells from the initial ones are established as the trajectories explore new regions of the state space. The third panel of Fig. 1c represents the state space's final partition, where each trajectory's length is less than the desired threshold, resulting in a mesh $\mathcal{M}$ of 60 states.

Figure 2a shows the optimal value functions obtained using grid-based partitioning combined with the introduced frameworks. The MDP optimal value function $V_\star$ is in the middle, while the extremes of the optimal interval value function $V_{\updownarrow_\star}$ obtained using the BMDP are above and below it, respectively $V_{\downarrow_\star}$ and $V_{\uparrow_\star}$. Lastly, the outermost value functions are derived from FTS, below the minimally optimal value function $\underline{V}_\star$ and above the maximally optimal value function $\overline{V}_\star$. The optimal value functions obtained using trajectory-based partitioning are represented in Fig. 2a, which



(a) FTS with grid partitioning. (b) MDP with grid partitioning. (c) BMDP with trajectory partitioning.

Fig. 3: Comparison of grid-based and trajectory-based state space partitioning and optimal policies obtained with FTS, MDP, and BMDP. The cell color represents the optimal action: red is for action $u = -1$, blue is for action $u = 1$, and cyan represents that both $u = -1$ and $u = 1$ are optimal.

follow the same order described above. It is worth noticing that the ordering of the value functions obtained reflects a more conservative system modeling. The MDP uses an exact probability distribution P, while the BMDP uses an interval probability distribution $P_{\updownarrow}$. Hence, since $P \in P_{\updownarrow}$, the corresponding optimal value functions meet the same relation, that is, $V_\star \in V_{\updownarrow_\star}$. Moreover, the FTS further simplifies the probability interval of BMDP by reducing it to binary conditions, from which $V_{\updownarrow_\star} \subset [\underline{V}_\star, \overline{V}_\star]$. Figure 3 compares optimal policies obtained through different pairs of partitioning algorithms and modeling frameworks. In Fig. 3a illustrates the optimal policy obtained by partitioning the state space with a grid and modeling the symbolic system with an FTS. This solution is far from the optimal control theory solution. Figure 3b exhibits a better solution obtained using the BMDP framework, which captures the system's behavior more accurately and can be improved by reducing the cell size. Lastly, Fig. 3c highlights the effectiveness of the presented approaches, as trajectory-based partitioning combined with the BMDP framework allows capturing the system dynamics accurately and returns the exact optimal solution.

VII. Conclusions

In this paper, we have introduced two novel methodologies for symbolic control of dynamic systems. The trajectory-based approach for defining the states of the symbolic model has been shown to provide a more accurate representation of the system's dynamics than the traditional grid-based technique. By dividing the state space across trajectories, we can identify more significant cells in terms of control, resulting in better performance than using a grid-based approach. Additionally, using a BMDP rather than an FTS to model the symbolic model has been demonstrated to be effective and robust in handling the stochastic behavior of the system and its uncertainties. The numerical results have further validated the effectiveness of combining these innovative approaches. Future work will focus on extending these techniques to more complex systems with higher-dimensional state spaces.

References

[1] D. M. R. Park, "Concurrency and automata on infinite sequences." *Theoretical computer science*, vol. 104, pp. 167–183, 1981.

[2] R. Milner, *Communication and concurrency*. Prentice Hall Englewood Cliffs, 1989, vol. 84.

[3] P. Tabuada, "Symbolic models for control systems," *Acta Informatica*, vol. 43, no. 7, pp. 477–500, 2007.

[4] A. Borri, G. Pola, and M. D. Di Benedetto, "A symbolic approach to the design of nonlinear networked control systems," in *ACM International Conference on Hybrid Systems*, 2012, pp. 255–264.

[5] ——, "Symbolic models for nonlinear control systems affected by disturbances," *International Journal of Control*, vol. 85, no. 10, pp. 1422–1432, 2012.

[6] A. Girard and G. J. Pappas, "Approximate bisimulations for nonlinear dynamical systems," in *IEEE Conference on Decision and Control*. IEEE, 2005, pp. 684–689.

[7] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[8] 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, 2011.

[9] A. Borri and C. Possieri, "Reinforcement learning for non-deterministic transition systems with an application to symbolic control," *IEEE Control Systems Letters*, vol. 7, pp. 1610–1615, 2023.

[10] A. Arnold, *Finite transition systems: semantics of communicating systems*. Prentice Hall International (UK) Ltd., 1994.

[11] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] R. Givan, S. Leach, and T. Dean, "Bounded-parameter Markov decision processes," *Artificial Intelligence*, vol. 122, no. 1-2, pp. 71–109, 2000.

[14] E. D. Sontag, *Mathematical control theory: deterministic finite dimensional systems*. Springer Science & Business Media, 2013.

[15] Y. Tazaki and J.-i. Imura, "Finite abstractions of discrete-time linear systems and its application to optimal control," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 201–10 206, 2008.

[16] L. Menini, C. Possieri, and A. Tornambe, "Distance to internal instability of linear time-invariant systems under structured perturbations," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 1941–1956, 2020.

[17] E. Haines, "Point in polygon strategies." *Graphics Gems*, vol. 4, pp. 24–46, 1994.

[18] R. Bridson, "Fast Poisson disk sampling in arbitrary dimensions." *SIGGRAPH sketches*, vol. 10, no. 1, p. 1, 2007.