Data-driven abstractions via adaptive refinements and a Kantorovich metric

Adrien Banse, Licio Romao, Alessandro Abate and Raphaël M. Jungers

***Abstract*— We introduce an adaptive refinement procedure for smart and scalable abstraction of dynamical systems. Our technique relies on partitioning the state space depending on the observation of future outputs. However, this knowledge is dynamically constructed in an adaptive, asymmetric way. In order to learn the optimal structure, we define a Kantorovich-inspired metric between Markov chains, and we use it to guide the state partition refinement. Our technique is prone to data-driven frameworks, but not restricted to.**

We also study properties of the above mentioned metric between Markov chains, which we believe could be of broader interest. We propose an algorithm to approximate it, and we show that our method yields a much better computational complexity than using classical linear programming techniques.

I. Introduction

Feedback control of dynamical systems is at the core of several techniques that have caused tremendous impact in several industries, being essential to important advancements in e.g. aerospace and robotics. Traditionally, these control techniques were model-based, relying on a complete mathematical model to perform controller design. With recent technological advancements, however, where a vast amount of data can be collected online or offline, the interest within the control community to study methods that leverage available data for feedback controller design has been reignited [1], [2], [3], [4].

In this paper, we focus on data-driven techniques for building *abstractions of dynamical systems.* Abstraction methods create a symbolic model [5], [6] that approximates the behavior of the original (the "concrete") dynamics in a way that controllers designed for such a symbolic representation can be refined to a valid controller for the original dynamics. The main advantage of abstraction methods with respect to standard control techniques is that one can transfer formal properties from the abstract system to the concrete one in a rigorous manner. Moreover, one can enforce complex temporal properties [7], [8], such as those described by LTL, STL, or PCTL temporal logics, using standard automata or MDP-based algorithms widely studied within the formal verification and control communities [9], [10], [11], [12], [13]. Classical abstraction methods, however, do not scale well with the state space dimension of the original dynamics, as they usually require a partitioning of the state space whose complexity grows exponentially with the underlying dimension. Besides, most of the existing abstraction techniques have been designed for when a full mathematical representation of the dynamics is available.

Several recent research efforts started exploring the possibility of generating data-driven abstractions for stochastic dynamical systems [14], [15], [16], [17]. In [14], we show that memory-based Markov models can be built from trajectory data. Memory has been classically used as a tool to mitigate non-Markovian behaviors of the original dynamics [11], [13], a feature also explored in recent papers [11], [16]. Increasing memory allows us to create more precise representations of the original dynamics using Markov decision processes or Markov chains. In [14], we also propose a heuristic to validate this observation, and prove a theoretic result showing convergence of the behavior of the discrete model to the original dynamics when memory increases (under observability and ergodicity assumptions). Despite promising results, [14] does not offer an adaptive mechanism to compute the generated abstraction, and thus it faces the curse of dimensionality, as the number of possible observations grows exponentially with the memory length.

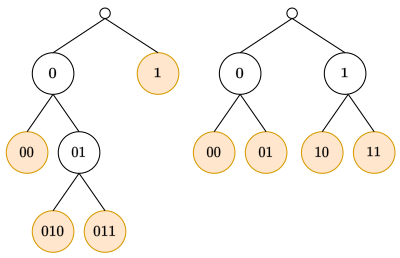

Fig. 1. Difference between an adaptive and a brute-force approach. On the left, an adaptive approach: only some of the possible observations are expanded. On the right, a brute-force approach: every observation is expanded. The two resulting abstractions have four states, but the adaptive one provides a better abstraction (see Example 1).

We follow-up on our results in [14] and improve them in this aspect. At the core of the approach in this paper is the construction of a novel metric between two Markov chains; this metric is then exploited to adaptively increase memory in certain regions of the state space, in view of taming the complexity of the generated abstraction. An illustration of the difference between these two approaches in depicted in

R. M. Jungers is a FNRS honorary Research Associate. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No 864017 - L2C. R. M. Jungers is also supported by the Walloon Region and the Innoviris Foundation. He is currently on sabbatical leave at Oxford University, Department of Computer Science, Oxford, UK. Adrien Banse is supported by the French Community of Belgium in the framework of a FNRS/FRIA grant. Adrien Banse and Raphaël M. Jungers are with ICTEAM, UCLouvain. E-mail adresses: {adrien.banse, raphael.jungers}@uclouvain.be. Licio Romao and Alessandro Abate are with the Department of Computer Science, Oxford University. E-mail adresses: {licio.romao, alessandro.abate}@cs.ox.ac.uk.

Manuscript 1509 submitted to 2023 62nd IEEE Conference on Decision and Control (CDC) . Received March 31, 2023.

Figure 1. As opposed to [14], where the states of the chain are built using past memory, the abstractions we construct in this paper are based on forward memory. In order to define a metric between two Markov models, we leverage the *Kantorovich* metric (also known as the Wasserstein or Earth's mover distance) between the induced probability on words of a fixed length and let the word length go to infinity. To define the Kantorovich metric, we equip the space of words with the *Baire distance* [18], turning it into a metric space. The Baire distance is classically used in symbolic dynamics [5], and we argue that it is a natural and relevant choice for control purposes, which is confirmed by numerical experiments. Not only we show that the proposed metric between Markov models is well-defined, but we also present an efficient algorithm that avoids solving linear programs of increasing size, which would lead to a prohibitive computational burden.

Computing metrics between Markov models has been an active research topic within the computer science community [19]. Our construction on the metric between Markov chains resembles the one presented in [20], however with another metric, namely the Kantorovich metric induced by the Baire distance. In [21] computability and complexity results are shown for the total variation metric. Kantorovich metrics for Markov models have been studied in [22], [23], [24], [25], but their underlying distance is different from ours. Our choice of the Baire distance is crucial both for computational aspects and for building smart abstractions of dynamical systems.

Summarizing, our approach is the first one to provide a data-driven, adaptive abstraction method for dynamical systems that leverages memory to alleviate non-Markorvian behaviors associated with the original dynamics. Overall, our main contribution is threefold. First, we propose a new metric to measure distance between Markov models. Second, we develop an efficient algorithm that approximates arbitrarily well the proposed metric. Third, we exploit the proposed metric to adaptively improve abstractions in specific regions of the state space.

Outline: The rest of this paper is organized as follows. In Section II, we introduce the Kantorovich metric between two Markov chains, and propose an efficient algorithm to approximate it with arbitrary precision. In Section III, we apply this metric to build data-driven abstractions of dynamical systems using a greedy strategy that leads to the refinement of the state-space partitioning. We also demonstrate the quality of our procedure on an example.

Notations: Let $\mathcal{A}$ be a finite alphabet. We denote the set of n-long sequences of this alphabet by $\mathcal{A}^n$, and the set of countably infinite sequences by $\mathcal{A}^*$. The symbol Λ stands for the empty sequence and, for any $w_1 \in \mathcal{A}^{n_1}$, $w_2 \in \mathcal{A}^{n_2}$, the sequence $w_1 w_2 \in \mathcal{A}^{n_1+n_2}$ is the concatenation of w_1 and w_2. Let $c(x)$ be a number of operations with respect to some attributes x. We say that an algorithm has a computational complexity $\mathcal{O}(f(x))$ if there exists $M > 0$, x_0 such that, for all $x \geq x_0$, $c(x) \leq Mf(x)$. For any bounded set $X \subset \mathbb{R}^d$, let $\sigma(X)$ be the induced σ-algebra of X, and λ be the Lebesgue measure on $\mathbb{R}^d$, then $\lambda_X : \sigma(X) \to [0,1]$ is defined as $\lambda_X(A) = \lambda(A)/\lambda(X)$. Finally, for any set X and function F, the set $F(X) = \{F(x) : x \in X\}$.

II. A Kantorovich metric between Markov chains

A. Preliminaries

Using a similar formalism as in [14], we define a labeled Markov chain as follows.

Definition 1 (Markov chain)**.** A *Markov chain* is a 5-tuple $\Sigma = (\mathcal{S}, \mathcal{A}, P, \mu, L)$, where $\mathcal{S}$ is a finite set of *states*, $\mathcal{A}$ is a finite *alphabet*, P is the *transition matrix* on $\mathcal{S} \times \mathcal{S}$, μ is the *initial measure* on $\mathcal{S}$, and $L : \mathcal{S} \to \mathcal{A}$ is a *labelling function*.

In Definition 1, the entry of the transition matrix $P_{s,s'}$ represents the probability $\mathbb{P}(X_{k+1} = s' | X_k = s)$. The labelling L induces a partition of the states. Consider the equivalence relation on $\mathcal{S}$ defined as $s \sim s'$ if and only if $L(s) = L(s')$. For any $a \in \mathcal{A}$, the notion of *equivalent classes* is defined as

$$[a] = \{s \in \mathcal{S} : L(s) = a\}. \tag{1}$$

We also define the *behavior of a Markov chain* $\mathcal{B}(\Sigma) \subseteq \mathcal{A}^*$ as follows. A sequence $w^* = (a_1, a_2, \dots) \in \mathcal{B}(\Sigma_{\mathcal{W}})$ if there exists $s_1, s_2, \dots \in \mathcal{S}$ such that $\mu_{s_1} > 0$, $P_{s_i, s_{i+1}} > 0$ and $L(s_i) = a_i$.

In the present work, we focus on a notion of metric between probabilities on label sequences. Let $w = (a_1, \dots, a_n)$ be a n-long sequence of labels, and define $p^n : \mathcal{A}^n \to [0,1]$ as

$$p^n(w) = \sum_{s_1 \in [a_1]} \mu_{s_1} \sum_{s_2 \in [a_2]} P_{s_1,s_2} \cdots \sum_{s_n \in [a_n]} P_{s_{n-1},s_n}, \tag{2}$$

that is the probability induced by the Markov chain on n-long sequences.

Remark 1. Classical procedures are well-known in the literature allowing to compute the probabilities p^n for increasing n, with a complexity proportional to $|\mathcal{S}|^2$ at every step [26].

We endow the set of n-long sequences of labels with the Baire distance d_B.

Definition 2 (Baire's distance, [18])**.** The *Baire distance* $d_B : \mathcal{A}^n \times \mathcal{A}^n \to \mathbb{R}$ is defined as $d_B(w_1, w_2) = 2^{-l}$, where l is the length of the longest common prefix. In other words, let $w_1 = (a_1, \dots, a_n)$ and $w_2 = (b_1, \dots, b_n)$, then $d_B(w_1, w_2) = 2^{-l}$, where $l = \inf\{k : a_k \neq b_k\}$.

B. The Kantorovich metric

Consider two Markov chains $\Sigma_1 = (\mathcal{S}_1, \mathcal{A}, P_1, \mu_1, L_1)$ and $\Sigma_2 = (\mathcal{S}_2, \mathcal{A}, P_2, \mu_2, L_2)$ defined on the same alphabet $\mathcal{A}$. For a fixed n, they respectively generate the distributions p_1^n and p_2^n on the metric space $(\mathcal{A}^n, d_B)$ as described in (2). We can now define the Kantorovich metric between p_1^n and p_2^n.

Definition 3 (Kantorovich metric)**.** The *Kantorovich metric* between the probability distributions p_1^n and p_2^n is given by

$$K(p_1^n, p_2^n) = \min_{\pi^n \in \Pi(p_1^n, p_2^n)} \sum_{w_1, w_2 \in \mathcal{A}^n} d_B(w_1, w_2)\pi^n(w_1, w_2), \tag{3}$$

where $\Pi(p_1^n, p_2^n)$ is the set of *couplings* of p_1^n and p_2^n, which contains the joint distributions $\pi^n : \mathcal{A}^n \times \mathcal{A}^n \to [0, 1]$ whose marginal distributions are p_1^n and p_2^n, that is,

$$\begin{aligned} &\forall w_1, w_2 \in \mathcal{A}^n : \pi^n(w_1, w_2) \geq 0, \\ &\forall w_1 \in \mathcal{A}^n : \sum_{w_2 \in \mathcal{A}^n} \pi^n(w_1, w_2) = p_1^n(w_1), \\ &\forall w_2 \in \mathcal{A}^n : \sum_{w_1 \in \mathcal{A}^n} \pi^n(w_1, w_2) = p_2^n(w_2). \end{aligned} \tag{4}$$

The Kantorovich metric is often interpreted as an optimal transport problem. Indeed one can see problem (3) as the problem of finding the optimal way to satisfy "demands" p_2^n with "supplies" p_1^n, where the cost of moving $\pi^n(w_1, w_2)$ probability mass from w_1 to w_2 amounts to $\pi^n(w_1, w_2)d_B(w_1, w_2)$. An illustration is provided in Figure 2.

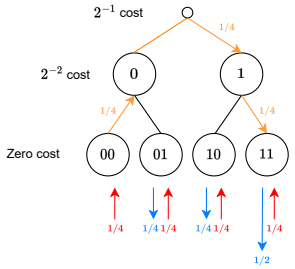

Fig. 2. Interpretation of the Kantorovich distance as an optimal transport problem. In this example, the alphabet $\mathcal{A} = \{0, 1\}$, $p_1^2(w) = 1/4$ for all $w \in \mathcal{A}^2$, and $p_2^2(00) = 0$, $p_2^2(01) = p_2^2(10) = 1/4$, and $p_2^2(11) = 1/2$. One can see that the optimal way to satisfy the demands p_2^n with the supplies p_1^n is to move $1/4$ of probability mass from 00 to 11, that is $\pi^2(00, 11) = 1/4$. Since $d_B(00, 11) = 1/2$, the Kantorovich distance is $K(p_1^2, p_2^2) = 1/8$.

A naïve computation of $K(p_1^n, p_2^n)$ in (3) entails solving a linear program. However, standard techniques, such as interior point methods and network simplex result in some cases in a complexity of $\mathcal{O}(n|\mathcal{A}|^{3n}\log(|\mathcal{A}|))$, and therefore scale very poorly with the number labels. In this section, we show that it is possible to compute $K(p_1^n, p_2^n)$ in $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|^{n+1})$ operations.

We present in Theorem 1 a key result for writing an efficient algorithm. Due to space constraints, its proof is omitted here but can be found in the extended version of this paper[1].

Theorem 1. *For any $n \geq 1$, let π^n be the solution of* (3)*. Then the following holds:*

$$\begin{aligned} K(p_1^{n+1}, p_2^{n+1}) &= K(p_1^n, p_2^n) \\ &+ 2^{-(n+1)} \sum_{w \in \mathcal{A}^n} \left[r(w) - \sum_{a \in \mathcal{A}} r(wa) \right], \end{aligned} \tag{5}$$

where

$$\begin{aligned} r(w) &= \min\{p_1^n(w), p_2^n(w)\}, \\ r(wa) &= \min\{p_1^{n+1}(wa), p_2^{n+1}(wa)\}. \end{aligned}$$

Theorem 1 allows to prove that Algorithm 1 efficiently computes the Kantorovich metric between p_1^n and p_2^n.

Algorithm 1 KANT(k, m, w, n)

for $i = 1, \ldots, |\mathcal{A}|$ **do**
  Compute $p_1^n(wa_i)$ and $p_2^n(wa_i)$ (see Remark 1)
  $r_i \leftarrow \min\{p_1^n(wa_i), p_2^n(wa_i)\}$
RES $= 2^{-(k+1)}(m - \sum_{i=1,\ldots,|\mathcal{A}|} r_i)$
if $k + 1 = n$ **then**
  return RES
for $i = 1, \ldots, |\mathcal{A}|$ **do**
  if $r_i \neq 0$ **then**
    RES $\leftarrow$ RES + KANT$(k + 1, r_i, wa_i, n)$
return RES

Corollary 1. *Let* KANT *be the algorithm described in Algorithm 1, then*

$$K(p_1^n, p_2^n) = \text{KANT}(0, 1, \Lambda, n). \tag{6}$$

Moreover KANT *terminates in $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|^{n+1})$ operations.*

Proof. We will prove that (6) holds by induction on the level of the execution tree of Algorithm 1. Let us prove that case $n = 1$. The constraints (4) imply that

$$\sum_{a_1, a_2 \in \mathcal{A}} \pi^1((a_1), (a_2)) = 1.$$

Therefore,

$$\begin{aligned} K(p_1^1, p_2^1) &= 2^{-1} \sum_{\substack{a_1, a_2 \in \mathcal{A} \\ a_1 \neq a_2}} \pi^1((a_1), (a_2)) \\ &= 2^{-1} \left[1 - \sum_{a \in \mathcal{A}} \pi^1((a), (a)) \right]. \end{aligned} \tag{7}$$

Moreover, for all $a \in \mathcal{A}$, the solution of $\pi^n((a), (a)) = r(wa)$. A formal proof of this can be found in the extended version of this paper. Therefore (7) is the result of KANT$(0, 1, \Lambda, 1)$. Now, assume that (6) holds for n. By Theorem 1,

$$\begin{aligned} K(p_1^{n+1}, p_2^{n+1}) &= \text{KANT}(0, 1, \Lambda, n) \\ &+ 2^{-(n+1)} \sum_{w \in \mathcal{A}^n} \left[r(w) - \sum_{a \in \mathcal{A}} r(wa) \right]. \end{aligned} \tag{8}$$

[1] https://adrienbanse.github.io/assets/pdf/cdc23_extended.pdf

Following the notations of Algorithm 1, let $m^w = r(w)$, and let $r_i^w = r(wa_i)$. One can re-write (8) as

$$K(p_1^{n+1}, p_2^{n+1}) = \textsc{Kant}(0, 1, \Lambda, n) + \sum_{w \in \mathcal{A}^n} 2^{-(n+1)} \left[m^w - \sum_{i=1,\dots,|\mathcal{A}|} r_i^w \right].$$

One can recognize $\textsc{Kant}(0, 1, \Lambda, n+1)$ in the right hand side of the equation above, which conludes the proof of (6).

In terms of computational complexity, the bottleneck of Algorithm 1 is the computation of p_1^n and p_2^n at each node of the execution tree. Following Remark 1, this can be done in $\mathcal{O}(|\mathcal{S}|^2)$ operations. Since there are $\mathcal{O}(|\mathcal{A}|^{n+1}|)$ nodes in the execution tree, the total number of operations is $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|^{n+1})$. ■

C. A metric between Markov chains

Let Σ_1 and Σ_2 be two Markov chains defined on the same alphabet $\mathcal{A}$. We define a metric between them as

$$d(\Sigma_1, \Sigma_2) = \lim_{n \to \infty} K(p_1^n, p_2^n),$$

where p_1^n and p_2^n are the distributions on $\mathcal{A}^n$ induced by each Markov chain on n-long label sequences.

Remark 2. The Baire distance 2^{-l} can be interpreted as a discount factor. Therefore, the metric $d(\Sigma_1, \Sigma_2)$, if well-defined, can be interpreted as a discounted measure of the difference between the behaviors $\mathcal{B}(\Sigma_1)$ and $\mathcal{B}(\Sigma_2)$.

We now prove that this distance is well-defined. A proof of Theorem 2 can be found in Appendix A.

Theorem 2. *The metric $d(\Sigma_1, \Sigma_2)$ is well-defined. Moreover, for any $n \geq 1$,*

$$0 \leq d(\Sigma_1, \Sigma_2) - K(p_1^n, p_2^n) \leq 2^{-n}.$$

Theorem 2 provides a guarantee on the approximation of $d(\Sigma_1, \Sigma_2)$ that we will be able to compute. Indeed, for any $\varepsilon > 0$, if $n \geq \lceil \log_2(\varepsilon^{-1}) \rceil$, then

$$0 \leq d(\Sigma_1, \Sigma_2) - K(p_1^n, p_2^n) \leq \varepsilon.$$

Following Corollary 1, for a fixed number of labels and states, this implies that an ε-solution can be found in $\mathcal{O}(\varepsilon^{-1})$ computational complexity.

III. Application: data-driven model abstractions

We now show how the metric $d(\Sigma_1, \Sigma_2)$ enables an adaptive refinement procedure for dynamical systems abstraction.

A. Abstractions with adaptive refinement

In this section, we introduce a new abstraction based on adaptive refinements. Even though our approach can be generalized to stochastic systems, in this preliminary work we focus on deterministic ones, which we now define.

Definition 4 (Dynamical system)**.** A *dynamical system* is the 4-tuple $S = (X, \mathcal{A}, F, H)$ that defines the relation

$$x_{k+1} = F(x_k), \quad y_k = H(x_k),$$

where $X \subseteq \mathbb{R}^d$ is the *state space*, $\mathcal{A}$ is a finite alphabet called the *output space*, $F : X \to X$ is a *transition function*, and $H : X \to \mathcal{A}$ is the *output function*. The variables x_k and y_k are called the *state* and the *output* at time k.

Also, in parallel to the definition of behavior of a Markov chain, we define the *behavior of a dynamical system* $\mathcal{B}(S) \subseteq \mathcal{A}^*$ as follows. A sequence $w^* = (a_1, a_2, \dots) \in \mathcal{B}(S)$ if there exists $x_1, x_2, \dots \in X$ such that $x_{i+1} = F(x_i)$ and $H(x_i) = a_i$. Also, in parallel to equivalent classes (1) on Markov chains, we define equivalent classes on the continuous state space X. A subset of states is an *equivalent class* if it satisfies the recursive relation

$$[wa]_S = \{x \in [w]_S \mid H^n(x) = a\}, \quad [\Lambda]_S = X,$$

for any $w \in \mathcal{A}^n$ and $a \in \mathcal{A}$. In other words, for a given sequence $w = (a_1, \dots, a_n)$, a state $x \in [w]_S$ if $H(x) = a_1$, $H(F(x)) = a_2$, …, and $H(F^{n-1}(x)) = a_n$. In this work, we impose the following assumption on dynamical systems.

Assumption 1. The dynamical system S as defined as Definition 4 is such that, for any $w \in \mathcal{A}^n$ and $a \in \mathcal{A}$, the following two conditions hold:

- If $\lambda_X([w]_S) = 0$, then $[w]_S = \emptyset$.
- If $\lambda_X([wa]_S) = \lambda_X([w]_S)$, then $[w]_S = [wa]_S$.

Informally, Assumption 1 requires that any possible trajectory has a nonzero probability to be sampled.

Definition 5 (Adaptive partitioning)**.** Let $w_1 \in \mathcal{A}^{n_1}$, $w_2 \in \mathcal{A}^{n_2}$, …, $w_k \in \mathcal{A}^{n_k}$ be k sequences of labels of different lengths. The set of sequences $\mathcal{W} = \{w_i\}_{i=1,\dots,k}$ is an *adaptive partitioning* for S if

$$\bigcup_{w \in \mathcal{W}} [w]_S = X, \quad \forall i \neq j,\ [w_i]_S \cap [w_j]_S = \emptyset.$$

We now introduce an abstraction procedure based on an adaptive partitioning refinements.

Definition 6 (Abstraction based on adaptive refinements)**.** Let $S = (X, \mathcal{A}, F, H)$ be a dynamical system as defined in Definition 4, and let $\mathcal{W}$ be an adaptive partitioning for S as defined in Definition 5. Then the corresponding *abstraction based on adaptive refinements* is the Markov chain $\Sigma_{\mathcal{W}} = (\mathcal{S}, \mathcal{A}, P, \mu, L)$ defined as follows:

- The states are the partitions, that is $\mathcal{S} = \mathcal{W}$.
- μ_w is the Lebesgue measure of equivalent class $[w]_S$ on X, that is $\mu_w = \lambda_X([w]_S)$.
- For $w_1 = (a_1, \dots, a_{n_1})$, and $w_2 = (b_1, \dots, b_{n_2})$, let $k = \min\{n_1 - 1, n_2\}$, $w_1' = (a_2, \dots, a_{k+1})$, and $w_2' = (b_1, \dots, b_k)$. If $w_1' \neq w_2'$ or $\lambda_X([w_1]_S) = 0$, then $P_{w_1, w_2} = 0$. Else
$$P_{w_1, w_2} = \frac{\lambda_X([a_1 w_2]_S)}{\lambda_X([w_1]_S)}.$$
- For $w = (a_1, \dots, a_n)$, $L(w) = a_1$.

Informally, for a given adaptive partitioning $\mathcal{W}$, the abstraction $\Sigma_{\mathcal{W}}$ can be interpreted as follows. The initial probability to be in the state w in the Markov chain is

the proportion of $[w]_S$ in X, and the probability to jump from the state w_1 to the state w_2 is the proportion of $[w_1]_S$ that goes into $[w_2]_S$ given the dynamics. We now provide a result that gives a sufficient condition for the abstraction to have the same behavior as the original system. A proof of Proposition 1 can be found in Appendix B.

Proposition 1. *Given a dynamical system S satisfying Assumption 1, consider abstraction $\Sigma_{\mathcal{W}}$ as per Definition 6. If for all $w_1, w_2 \in \mathcal{W}$, $P_{w_1,w_2} \in \{0,1\}$, then $\mathcal{B}(\Sigma_{\mathcal{W}}) = \mathcal{B}(S)$.*

B. A data-driven abstraction

In this section, we propose a method to construct an abstraction based on adaptive refinements, from a data set comprising outputs sampled from the dynamical model S. Given an adaptive partitioning $\mathcal{W}$, we propose to construct $\Sigma_{\mathcal{W}}$ using empirical probabilities (see [14] for more details). We make the following assumption, which considers an idealised situation where one has an infinite number of samples. We leave for further work the relaxation of this assumption.

Assumption 2. For any abstraction $\Sigma_{\mathcal{W}} = (\mathcal{W}, \mathcal{A}, P, \mu, L)$, the transition probabilities P and the initial distribution μ are known exactly.

Now we are able to use the tool investigated in Section II to find a smart adaptive partitioning. Indeed, one can construct two abstractions $\Sigma_{\mathcal{W}_1}$ and $\Sigma_{\mathcal{W}_2}$ corresponding to two different partitionings, and efficiently compute the Kantorovich metric $d(\Sigma_{\mathcal{W}_1}, \Sigma_{\mathcal{W}_2})$ up to some accuracy ε following Corollary 1. This gives a discounted measure of the difference between $\mathcal{B}(\Sigma_{\mathcal{W}_1})$ and $\mathcal{B}(\Sigma_{\mathcal{W}_2})$ (see Remark 2). This reasoning leads to the greedy procedure $\textsc{Refine}(S, N, \varepsilon)$ described in Algorithm 2.

Algorithm 2 $\textsc{Refine}(S, N, \varepsilon)$

```
W ← {(a)}_{a∈A}
Construct Σ_W from samples of S
while ∃w1, w2 ∈ W : P_{w1,w2} ∈ (0,1) do
    if N = 0 then
        return Σ_W
    for i = 1, ..., |W| do
        W'_i ← W \ {w_i}
        W'_i ← W'_i ∪ {w_i a}_{a∈A}
        Construct Σ_{W'_i} from samples of S
        d_i ← d(Σ_W, Σ_{W'_i}) with precision ε
    j = arg max_{i=1,...,|W|} d_i
    W ← W'_j
    Σ_W ← Σ_{W'_j}
    N ← N − 1
return Σ_W
```

An interpretation of Algorithm 2 goes as follows. Let $\mathcal{W}$ be a coarse partitioning, and $\mathcal{W}'_1$ and $\mathcal{W}'_2$ be two more refined partitionings. If $d(\Sigma_{\mathcal{W}}, \Sigma_{\mathcal{W}'_1}) > d(\Sigma_{\mathcal{W}}, \Sigma_{\mathcal{W}'_2})$, then one could argue that it is more interesting to choose $\mathcal{W}'_1$ over $\mathcal{W}'_2$, since the discounted measure between the behaviors corresponding to the coarse partitioning and the refined partitioning is larger. Moreover, if at some point $\Sigma_{\mathcal{W}}$ is such that $P_{w,w'} \in \{0,1\}$ for all $w, w' \in \mathcal{A}^n$, then one has a sufficient condition to stop the algorithm following Proposition 1, otherwise the algorithm stops after N iterations. If $N = \infty$, then the algorithm only stops in such case. An execution step of the algorithm can be found in Figure 3. A complexity analysis of Algorithm 2 can be found in Corollary 2, whose proof is in Appendix C.

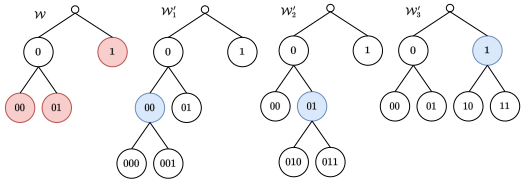

Fig. 3. Illustration of the execution of Algorithm 2. Consider a current partitioning $\mathcal{W} = \{00, 01, 1\}$, with the corresponding abstraction $\Sigma_{\mathcal{W}}$. Then the algorithm will explore the partitionings $\mathcal{W}'_1 = \{000, 001, 01, 1\}$, $\mathcal{W}'_2 = \{00, 010, 011, 1\}$ and $\mathcal{W}'_3 = \{00, 01, 10, 11\}$. For each one, it will compute $\Sigma_{\mathcal{W}'_i}$, and $d(\Sigma_{\mathcal{W}}, \Sigma_{\mathcal{W}'_i})$, and choose the one for which the distance to $\Sigma_{\mathcal{W}}$ is the largest.

Corollary 2. *The algorithm $\textsc{Refine}(S, N, \varepsilon)$ terminates in $\mathcal{O}(|\mathcal{A}|^{n+4} N^4)$ operations, with $n = \lceil \log_2(\varepsilon^{-1}) \rceil$. Moreover, for S satisfying Assumption 1, if $\Sigma_{\mathcal{W}} = \textsc{Refine}(S, \infty, \varepsilon)$ terminates, then $\mathcal{B}(\Sigma_{\mathcal{W}}) = \mathcal{B}(S)$.*

C. Numerical examples

In this section, we demonstrate on an example that our greedy algorithm converges to a smart partitioning[2], and we show how to use the proposed framework for controller design.

Example 1. Consider $S = (X, \mathcal{A}, F, H)$ with $X = [0,2] \times [0,1]$, $\mathcal{A} = \{0,1\}$. Let F be defined as

$$F(x) = \begin{cases} x & \text{if } x \in P_1 \cup P_4, \\ (x_1/2 + 1/2, x_2 + 1/2) & \text{if } x \in P_2, \\ (x_1 - 1/2, x_2) & \text{if } x \in P_3, \\ (2x_1 + 1, 4x_2 - 3/4) & \text{else,} \end{cases}$$

where P_i are depicted in Figure 4, and $H(x) = 0$ if $x \in P_1$, else $H(x) = 1$. An illustration and interpretation of S is given in Figure 4.

The result of the algorithm applied to Example 1 at all iterations k is depicted in Table I.

The final partitioning is illustrated in Figure 5. Observe that the generated partitioning aligns well with the dynamics, and that our algorithm generates an emerging structure which is not trivial. The algorithm stops at the third iteration since the obtained data-driven abstraction is such that $P_{w,w'} \in \{0,1\}$, which is a stopping criterion following Proposition 1, and has much less states than the brute force approach of [14].

[2]All the code corresponding to this section can be found at `https://github.com/adrienbanse/KantorovichAbstraction.jl`.

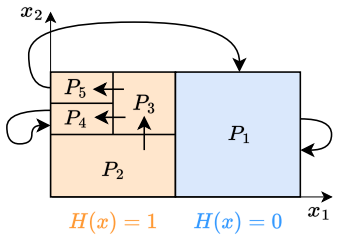

Fig. 4. Illustration and description of the transition function F of Example 1. F has to be understood in the following way: P_1 is mapped to itself, P_2 is mapped to P_3, the lower half of P_3 is mapped to P_4, the upper half of P_3 is mapped to P_5, P_5 is mapped to P_1, and P_4 to itself.

	$\mathcal{W}$	$d\left(\Sigma_{\mathcal{W}}, \Sigma_{\mathcal{W}'_j}\right)$	$P_{w,w'} \in \{0,1\}$
$k=0$	$\{0, \mathbf{1}\}$	0.0015	No
$k=1$	$\{0, 10, \mathbf{11}\}$	0.0059	No
$k=2$	$\{0, 10, 110, \mathbf{111}\}$	0.0039	No
$k=3$	$\{0, 10, 110, 1110, 1111\}$	-	**Yes**

TABLE I. Results of Algorithm 2 for Example 1. For each iteration k, the current model is the abstraction corresponding to $\Sigma_{\mathcal{W}}$, and the chosen model is $\Sigma_{\mathcal{W}'_j}$, with the largest distance $d(\Sigma_{\mathcal{W}}, \Sigma_{\mathcal{W}'_j})$. With P the transition matrix of the current model, if for all $w, w' \in \mathcal{A}^n$, $P_{w,w'} \in \{0,1\}$, the algorithm stops.

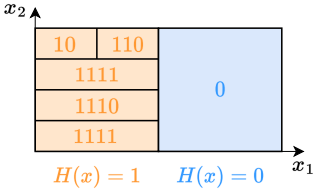

Fig. 5. Illustration of the last partitioning $\mathcal{W}$ given by Algorithm 2 for Example 1.

We further demonstrate the quality of the obtained abstractions by designing a controller for a similar dynamical system.

Example 2. Consider the dynamical system S as described in Example 1, except that the dynamics is controlled as follows:

$$\tilde{x}_k = x_k + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_k, \quad x_{k+1} = F(\tilde{x}_k)$$

where $u_k = K(x_k) \in \{0, 1/4, 1/2\}$ is an input to the system. Consider the reward $r(x) = 1$ if $H(x) = 0$, else $r(x) = 0$, and a discounted expected reward maximization objective, that is

$$\max_K \mathbb{E}_{x_1 \sim \mu(X)} \sum_k \gamma^k r(x_k), \tag{9}$$

where $\mu(X)$ is the uniform distribution on X, and $\gamma = 0.95$ is a discount factor.

To solve this optimal control problem, we will use the abstractions constructed by Algorithm 2. For each partitioning $\mathcal{W}$ in Table I, we will construct the data-driven abstraction $\Sigma^u_{\mathcal{W}}$ corresponding to the actions given in Example 2, that is $u = 0$, $u = 1/4$ and $u = 1/2$. We will then solve a Markov Decision Process (or MDP for short, see [27] for an introduction) maximizing the expected reward of the MDP. For this, we used the implementation of the value iteration algorithm implemented in the `POMDPs.jl` Julia package [28]. Now, let $P(s)$ be the optimal policy for the state s, we design the controller for the system 2 as follows: for $x_k \in [w]_S$, then

$$u_k = K(x_k) = P(w). \tag{10}$$

For the different abstractions found by Algorithm 2, the corresponding expected rewards (9) for the original system controlled by (10) are given in Table II. One can see that the expected reward increases as our algorithm refines the state-space.

Iteration	Controller (10)	**Expected reward** (9)
$k=0$	$K(x) = \begin{cases} 0 & \text{if } x \in [0]_S \\ 0 & \text{if } x \in [1]_S \end{cases}$	**14.4784**
$k=1$	$K(x) = \begin{cases} 0 & \text{if } x \in [0]_S \\ 0 & \text{if } x \in [10]_S \\ 1/4 & \text{if } x \in [11]_S \end{cases}$	**18.8726**
$k=2$	$K(x) = \begin{cases} 0 & \text{if } x \in [0]_S \\ 0 & \text{if } x \in [10]_S \\ 0 & \text{if } x \in [110]_S \\ 1/4 & \text{if } x \in [111]_S \end{cases}$	**19.0311**
$k=3$	$K(x) = \begin{cases} 0 & \text{if } x \in [0]_S \\ 0 & \text{if } x \in [10]_S \\ 0 & \text{if } x \in [110]_S \\ 1/2 & \text{if } x \in [1110]_S \\ 1/4 & \text{if } x \in [1111]_S \end{cases}$	**19.1022**

TABLE II. Expected rewards (9) for the Example 2 controlled by (10). The iterations k correspond to the iterations of Algorithm 2 represented in Table I. The optimal policy is found by solving MDPs corresponding to the three possible actions $u_k \in \{0, 1/4, 1/2\}$, and the expected reward (9) is approximated by sampling 5000 trajectories of length 1000. One can observe that the expected reward increases.

IV. Conclusion and Further Research

We now summarize the contributions of this work. First, we proposed a Kantorovich metric between two Markov chains, defined on an underlying space defined by the Baire distance. We showed that one can arbitrarily approximate this metric with an efficient algorithm, and therefore efficiently compute a discounted measure between the behaviors of two Markov chains. We then applied this metric to the construction of data-driven abstractions based on adaptive refinement. More precisely, we propose a greedy procedure using the Kantorovich metric to assess the difference between two abstractions. We showed that, in some cases, the obtained abstraction has exactly the same behavior as the original dynamical system. We also demonstrated the quality of our procedure by designing a controller for the original system, from the obtained finite model.

As further research, we would like to investigate other underlying distances than the Baire distance, and characterize those for which the Kantorovich metric can be efficiently computed. We would also like to investigate even more efficient algorithms to compute the proposed metric between

Markov chains, or to establish rigorous connections with alternative metrics in the litterature. Finally, we plan to design a smart stopping criterion for our refinement procedure. For example, we could quantify a difference between the behavior of the abstraction at any iteration, and the behavior of the original system.

ACKNOWLEDGMENT

We thank Prof. Franck van Breugel and Prof. Prakash Panangaden for their insightful comments and the interesting conversations about this work.

REFERENCES

[1] C. D. Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, pp. 909–924, 3 2020.

[2] Z. Wang and R. M. Jungers, "A data-driven method for computing polyhedral invariant sets of black-box switched linear systems," *IEEE Control Systems Letters*, vol. 5, pp. 1843–1848, 11 2021.

[3] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, 6 2021.

[4] A. Banse, Z. W. Raphaël, and M. Jungers, "Black-box stability analysis of hybrid systems with sample-based multiple lyapunov functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 7284–7289.

[5] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 2003.

[6] A. van der Schaft, "Equivalence of dynamical systems by bisimulation," *IEEE Transactions on Automatic Control*, vol. 49, pp. 2160–2172, 2004.

[7] C. Baier and J. P. Katoen, *Principles of Model Checking*. MIT Press Books, 2008.

[8] P. Tabuada, *Verification and Control of Hybrid Systems*. Springer, 2009.

[9] M. Kwiatkowska, D. Parker, and G. Norman, "Prism 4.0: Verification of probabilistic real-time systems," 2011, pp. 1–6.

[10] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," R. Majumdar and V. Kunčak, Eds., vol. 10427. Springer International Publishing, 2017.

[11] R. Majumdar, N. Ozay, and A. K. Schmuck, "On abstraction-based controller design with output feedback." Association for Computing Machinery, Inc, 4 2020.

[12] T. Badings, L. Romao, A. Abate, D. Parker, H. Poonawala, M. Stoelinga, and N. Jensen, "Robust control for dynamical systems with non-gaussian via formal abstractions," *Journal of Artificial Intelligence Research*, vol. 76, pp. 341–391, 2023.

[13] A. K. Schmuck and J. Raisch, "Asynchronous l-complete approximations," *Systems and Control Letters*, vol. 73, pp. 67–75, 2014.

[14] A. Banse, L. Romao, A. Abate, and R. M. Jungers, "Data-driven memory-dependent abstractions of dynamical systems," 2022. [Online]. Available: https://arxiv.org/abs/2212.01926

[15] A. Devonport, A. Saoud, and M. Arcak, "Symbolic abstractions from data: A pac learning approach," 4 2021. [Online]. Available: http://arxiv.org/abs/2104.13901

[16] R. Coppola, A. Peruffo, and M. M. Jr., "Data-driven abstractions for verification of deterministic systems," 2023. [Online]. Available: https://arxiv.org/abs/2211.01793

[17] A. Lavaei, S. Soudjani, E. Frazzoli, and M. Zamani, "Constructing mdp abstractions using data with formal guarantees," 6 2022. [Online]. Available: http://arxiv.org/abs/2206.14402

[18] R. Baire, "Sur la représentation des fonctions discontinues: Première partie," *Acta Mathematica*, vol. 30, no. none, pp. 1–48, Jan. 1906, publisher: Institut Mittag-Leffler.

[19] Y. Deng and W. Du, "The Kantorovich Metric in Computer Science: A Brief Survey," *Electronic Notes in Theoretical Computer Science*, vol. 253, no. 3, pp. 73–82, Nov. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066109004265

[20] Z. Rached, F. Alajaji, and L. Campbell, "The Kullback–Leibler Divergence Rate Between Markov Sources," *Information Theory, IEEE Transactions on*, vol. 50, pp. 917–921, Jun. 2004.

[21] S. Kiefer, "On Computing the Total Variation Distance of Hidden Markov Models," Apr. 2018, arXiv:1804.06170 [cs]. [Online]. Available: http://arxiv.org/abs/1804.06170

[22] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, "Metrics for labelled Markov processes," *Theoretical Computer Science*, vol. 318, no. 3, pp. 323–354, Jun. 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397503006042

[23] F. van Breugel, B. Sharma, and J. Worrell, "Approximating a Behavioural Pseudometric Without Discount for Probabilistic Systems," in *Foundations of Software Science and Computational Structures*, ser. Lecture Notes in Computer Science, H. Seidl, Ed. Berlin, Heidelberg: Springer, 2007, pp. 123–137.

[24] N. Madras and D. Sezer, "Quantitative bounds for Markov chain convergence: Wasserstein and total variation distances," *Bernoulli*, vol. 16, no. 3, pp. 882–908, Aug. 2010, publisher: Bernoulli Society for Mathematical Statistics and Probability. [Online]. Available: https://projecteuclid.org/journals/bernoulli/volume-16/issue-3/Quantitative-bounds-for-Markov-chain-convergence--Wasserstein-and-total/10.3150/09-BEJ238.full

[25] D. Rudolf and N. Schweizer, "Perturbation theory for Markov chains via Wasserstein distance," *Bernoulli*, vol. 24, no. 4A, pp. 2610–2639, Nov. 2018, publisher: Bernoulli Society for Mathematical Statistics and Probability. [Online]. Available: https://projecteuclid.org/journals/bernoulli/volume-24/issue-4A/Perturbation-theory-for-Markov-chains-via-Wasserstein-distance/10.3150/17-BEJ938.full

[26] P. Bikash, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, Jan. 1989. [Online]. Available: https://www.academia.edu/1806671/A_tutorial_on_hidden_Markov_models_and_selected_applications_in_speech_recognition

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[28] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty," *Journal of Machine Learning Research*, vol. 18, no. 26, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-300.html

APPENDIX

A. Proof of Theorem 2

Let $K_n = K(p_1^n, p_2^n)$. First, we prove that, for all $n \geq 1$,

$$0 \leq K_{n+1} - K_n \leq 2^{-n}. \tag{11}$$

Following Theorem 1, it suffices to show that

$$0 \leq \sum_{w \in \mathcal{A}^n} \left[r(w) - \sum_{a \in \mathcal{A}} r(wa) \right] \leq 1, \tag{12}$$

By the law of total probability, we have that

$$p_1^n(w) = \sum_{a \in \mathcal{A}} p_1^{n+1}(wa),$$

and similarly for $p_2^n(w)$. Hence

$$0 \leq r(w) - \sum_{a \in \mathcal{A}} r(wa) \leq r(w),$$

which shows that (12) holds. Now, notice that (11) implies that the sequence $(K_n)_{n \geq 1}$ is monotone, and bounded since

$$\lim_{n \to \infty} K_n \leq \sum_{n \geq 1} (K_{n+1} - K_n) \leq \sum_{n \geq 1} 2^{-n} = 1.$$

By the monotone convergence theorem, the limit exists and is equal to

$$\lim_{n \to \infty} K_n = \sup_{n \geq 1} K_n.$$

Moreover, since K_n is a distance for every $n \geq 1$, and the limit exists, we have that $\lim_{n\to\infty} K_n$ is also a distance. Finally, by (11),

$$\lim_{n\to\infty} K_n - K_p \leq \sum_{n\geq p} K_n = 2^{-p},$$

for any $p \geq 1$, which concludes the proof of the theorem.

B. Proof of Proposition 1

We first prove that, if there are $w_1, w_2 \in \mathcal{W}$ such that $P_{w_1,w_2} = 1$, then

$$F([w_1]_S) \subseteq [w_2]_S. \tag{13}$$

Let w_1, w_2, k, w_1' and w_2' be as in Definition 6. Let us note that

$$\begin{aligned} F([w_1]_S) &= \left\{ F(x) \in X \;\middle|\; \begin{array}{l} H(x) = a_1, \\ H(F(x)) = a_2 \\ \dots \\ H(F^{n-1}(x)) = a_{n_1} \end{array} \right\} \\ &= F([(a_1)]_S) \cap [(a_2, \dots, a_{n_1})]_S \end{aligned} \tag{14}$$

Now, since $P_{w_1,w_2} > 0$, then $w_1' = w_2'$. There are two cases, either $w_1' = (a_2, \dots, a_{k+1})$ and $w_2' = w_2$, or $w_1' = w_1$ and $w_2' = (b_1, \dots, b_k)$. Let us investigate these separately. In the first case, $(a_2, \dots, a_{k+1}) = w_2$. By definition,

$$[(a_2, \dots, a_{n_1})]_S \subseteq [(a_2, \dots, a_{k+1})]_S = [w_2]_S.$$

Therefore (14) implies

$$F([w_1]_S) \subseteq F([(a_1)]_S) \cap [w_2]_S \subseteq [w_2]_S,$$

which is (13). In the second case, assume that

$$[w_1]_S = [a_1 w_2]_S, \tag{15}$$

then

$$\begin{aligned} F([w_1]_S) &= F([a_1 w_2]_S) \\ &= F([(a_1)]_S) \cap [w_2]_S \\ &\subseteq [w_2]_S, \end{aligned}$$

where a very similar as in (14) was used. It remains to show that (15) holds. Since we are in the second case cited above, then

$$a_1 w_2 = w_1(b_{k+1}, \dots, b_{n_2}),$$

which implies that $[a_1 w_2]_S \subseteq [w_1]_S$. Moreover, $P_{w_1,w_2} = 1$, that is

$$\lambda_X([a_1 w_2]_S) = \lambda_X([w_1]_S).$$

Following Assumption 1, it means that $[a_1 w_2]_S = [w_1]_S$, which proves (13).

Now we prove that (13) implies $\mathcal{B}(\Sigma_\mathcal{W}) = \mathcal{B}(S)$. We first prove that $\mathcal{B}(S) \subseteq \mathcal{B}(\Sigma_\mathcal{W})$. Let $w^* = (a_1, a_2, \dots) \in \mathcal{A}^*$ such that $w^* \notin \mathcal{B}(\Sigma_\mathcal{W})$, then there are a_i, a_{i+1} such that, for all $w_1, w_2 \in \mathcal{W}$ for which $L(w_1) = a_i$ and $L(w_2) = a_{i+1}$, $P_{w_1,w_2} = 0$. This could mean three things.

1) For all such w_1, $\lambda_X([(a_i)]_S) = 0$. Following Assumption 1, it means that $[(a_i)]_S = \emptyset$.
2) Let $w_2 = (a_{i+1}, b_2, \dots, b_{n_2})$. For all such w_2,

$$\lambda_X([(a_i, a_{i+1}, b_2, \dots, b_{n_2})]_S) = 0,$$

which means by Assumption 1 that

$$[(a_i, a_{i+1}, b_2, \dots, b_{n_2})]_S = \emptyset.$$

By Definition 5, it implies that $[a_i a_{i+1}]_S = \emptyset$.

In any case, it means that $w^* \neq \mathcal{B}(S)$. Now we prove $\mathcal{B}(\Sigma_W) \subseteq \mathcal{B}(S)$. Let $w^* = (a_1, a_2, \dots) \in \mathcal{B}(\Sigma_\mathcal{W})$. It means that there exists $w_1, w_2, \dots \in \mathcal{W}$ such that $L(w_i) = a_i$ and $P_{w_i,w_{i+1}} = 1$. Following (13), this implies that $F([w_i]_S) \subseteq [w_{i+1}]_S$. This implies that $[a_i a_{i+1}]_S \neq \emptyset$, which means $w^* \in \mathcal{B}(S)$.

C. Proof of Corollary 2

Proposition 1 gives a sufficient condition to stop the algorithm, hence the second part of the claim. It remains to prove that the computational complexity is the claimed one. Let $\mathcal{W}^{(k)}$ and $\mathcal{W}_i^{'(k)}$ be the abstractions $\mathcal{W}$ and $\mathcal{W}_i'$ at iteration k in Algorithm 2. First we note that

$$|\mathcal{W}^{(k)}| = k|\mathcal{A}| - (k-1), \quad |\mathcal{W}_i^{'(k)}| = (k+1)|\mathcal{A}| - k \tag{16}$$

At each iteration k, one has to compute $|\mathcal{W}^{(k)}|$ times the ε-accurate Kantorovich distance between two models of sizes given by (16). By Corollary 1, such computational complexity is

$$\begin{aligned} &\mathcal{O}\left(|\mathcal{W}^{(k)}|\left(|\mathcal{A}|^{n+1}\left(|\mathcal{W}^{(k)}|^2 + |\mathcal{W}_i^{'(k)}|^2\right)\right)\right) \\ =&\mathcal{O}\left(|\mathcal{A}|^{n+1}|\mathcal{W}_i^{'(k)}|^3\right) \\ =&\mathcal{O}\left(|\mathcal{A}|^{n+4}(k+1)^3\right). \end{aligned}$$

Now, the worst-case is when the algorithm does not converge to an abstraction where there exists w, w' such that $P_{w,w'} \in (0,1)$. Therefore, the total computational complexity is

$$\sum_{k=1}^{N} \mathcal{O}\left(|\mathcal{A}|^{n+4}(k+1)^3\right) = \mathcal{O}\left(|\mathcal{A}|^{n+4}N^4\right),$$

which is the claim.