Learning Broadcast Protocols

Dana Fisman
Department of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
dana@cs.bgu.ac.il

Noa Izsak
Department of Computer Science
Ben-Gurion University
Beer-Sheva, Israel
izsak@post.bgu.ac.il

Swen Jacobs
CISPA Helmholtz Center
for Information Security
Saarbrücken, Germany
jacobs@cispa.de

***Abstract*—The problem of learning a computational model from examples has been receiving growing attention. Models of distributed systems are particularly challenging since they encompass an added succinctness. While positive results for learning some models of distributed systems have been obtained, so far the considered models assume a fixed number of processes interact. In this work we look for the first time (to the best of our knowledge) at the problem of learning a distributed system with an arbitrary number of processes, assuming only that there *exists* a cutoff. Specifically, we consider *fine broadcast protocols*, these are broadcast protocols (BPs) with a finite cutoff and no hidden states. We provide a learning algorithm that given a sample consistent with a fine BP, can infer a correct BP, with the help of an SMT solver. Moreover we show that the class of fine BPs is teachable, meaning that we can associate a finite set of words $\mathcal{S}_B$ with each BP B in the class (a so-called characteristic set) so that the provided learning algorithm can correctly infer a correct BP from any consistent sample subsuming $\mathcal{S}_B$. On the negative size we show that (a) characteristic sets of exponential size are unavoidable, (b) the consistency problem for fine BPs is NP hard, and (c) fine BPs are not polynomially predictable.**

I. Introduction

Learning computational models has a long history starting with the seminal works of Gold [23], [24] and Angluin [3]. Questions regarding learning computational models have raised a lot of interest also in the verification community [33], [34]. Many results regarding the learnability of various computational models used in verification have already been obtained (e.g. [8], [10], [13], [5], [1], [14], [4], [35], [21]).

Particularly challenging is learning of concurrent computational models, as they offer another level of succinctness, and usually have no unique minimal model. Various results regarding learning concurrent models have already been obtained [11], [19], [31], but only for models with a fixed number of processes.

Broadcast protocols (BPs) are a powerful concurrent computational model, allowing the synchronous communication of the sender of an action with an arbitrary number of receivers [16]. BPs have mainly been studied in the context of parameterized verification, i.e., solving the question whether a given property holds for all systems where an arbitrary number of processes executes a given protocol. Esparza et al. [18] have shown that this problem is decidable for safety properties, and undecidable for liveness.

The challenge in verifying parameterized systems such as broadcast protocols, is that a parameterized system concisely represents an infinite family of systems: for each natural number n it includes the system where n processes interact. The system is correct only if it satisfies the specification for any number n of processes interacting. A variety of approaches has been investigated to overcome this. Some of these are based on the notion of *cutoff*. Generally speaking, a cutoff is a number c of processes such that a given property holds for any instance of the system with $n \geq c$ processes if and only if it holds for the cutoff system. Then, cutoffs provide a complete method for proving safety properties of parameterized systems [32], i.e., for every safety property there exists a cutoff.

In the literature, many results exist that provide cutoffs for whole *classes of properties* in a given computational model. For token rings, Emerson and Namjoshi provide cutoffs for branching-time safety and liveness properties [17]. For guarded protocols, cutoffs for linear-time safety and liveness properties have been studied [15], [7], [28]. For rendezvous systems, it has been shown that there are protocols that do not have a cutoff for all linear-time properties, but if a protocol has a cutoff then all of its executions are (ω-)regular [2]. Finally, cutoffs also enable the *synthesis* of implementations for parameterized systems from formal specifications [27], [26], [30], a problem closely related to learning.

In this paper, we develop a learning approach for broadcast protocols. Given the expressiveness of BPs and the complexity of the general problem, we make some assumptions to keep the problem manageable. In particular, we assume that the BP under consideration has no hidden states, i.e., every state has a least one broadcast sending action by which it can be recognized, and that there *exists* a cutoff, i.e., a number c such that the language derived by c processes is the same as the language derived by any number greater than c. We call such broadcast protocols *fine*. We note that not all broadcast protocols have a cutoff (whether or not they have hidden states), and that when a cutoff exists the derived language is regular.[1] The fact that the derived language is regular also holds in previous work on learning concurrent models (communicating automata [11], workfklow petri nets [19],

[1]The language of a BP in general need not be regular [20], [22] and this is true also with the restriction to no-hidden states.

and negotiation protocols [31]) merely since a finite number of essentially finite state machines is in consideration. We emphasize that this does not reduce the problem to learning a regular language, since the aim is to obtain the concurrent representation, which we show to be much more succinct than a respective DFA for the language. Moreover, the problem we consider goes way beyond what has been considered in previous work in the sense that our approach works if *there exists* a cutoff, but in contrast to existing approaches it does not require that the cutoff is known a priori, or that the system is assumed to consist of a known fixed number of processes.

We focus on passive learning paradigms [12]. Specifically, we consider the following problems 1) *Consistency* — whether there exists a BP with at most k states that agrees with a given sample, 2) *Inference* — given a sample consistent with a BP, return a BP that is consistent with the sample, 3) *Teachability* — whether there exists a finite sample $\mathcal{S}_B$, a so called *characteristic set*, that can be associated with any BP B such that a learning algorithm can correctly infer a BP equivalent to B from any sample subsuming $\mathcal{S}_B$ (and consistent with B), 4) *CS Polynomiality* — if the class is teachable, whether characteristic sets are of polynomial size, and 5) *Polynomial Predictability* — whether a learner can correctly classify an unknown word with high probability after asking polynomially many membership and draw queries.

We show, in Sec.III, that consistency is NP-hard for the class of fine BPs, and prove a few basic properties of BPs relevant to learning in Sec.IV. In Sec.V, we provide an inference algorithm that, given a sample of words that are consistent with a fine BP, can infer a correct BP. The inference approach is constraint-based and can be implemented using an SMT solver. In Sec.VI, we show that this class of BPs is *teachable*. In Sec.VII, we show that there exists a family of fine BPs for which a characteristic set of polynomial size cannot be obtained. The same family shows the succinctness of fine BPs, since the minimal corresponding DFA is exponentially larger than the fine BP. In Sec.VIII we show that fine BPs are not polynomially predictable. We have implemented the inference algorithm and report on some preliminary experimental data in Sec.IX.

II. Preliminaries

A. Broadcast Protocols

a) Broadcast Protocols (BP): A *broadcast protocol* $B = (S, s_0, L, R)$ consists of a set of states S with an initial state $s_0 \in S$, a set of labels L and a transition relation $R \subseteq S \times L \times S$, where $L = \{a!!, a?? \mid a \in A\}$ for some set of actions A. A transition labeled with $a!!$ is a *sending transition*, and a transition labeled with $a??$ is a *receiving transition*, also called a *response*.[2] For each action $a \in A$, a receiving transition should be enabled from every state. In addition, following [9], we assume that for each action a, there is a unique state s_a with an outgoing sending transition on $a!!$. A state s in a broadcast protocol is said to be *hidden* if it has no outgoing sending transition. In this paper we consider broadcast protocols with no hidden states.

b) The Counter System B^n: For systems composed of n instances of a given broadcast protocol B, we will assume some ordering $s_0, s_1, ...s_{|S|-1}$ on S, and identify global states with vectors from $[n]^{|S|}$ where $[n]$ denotes the set $\{0, 1, \ldots, n\}$. We sometimes refer to a global state as a state-vector. We use bold font for global states and $\mathbf{q}[i]$ to denote the entry in position i of a state-vector $\mathbf{q}$. For example, let $\mathbf{u}_j$ be the unit vector with $\mathbf{u}_j[j] = 1$ and $\mathbf{u}_j[i] = 0$ for all $i \neq j$. Then the global state where all n instances of B are in s_0 is the vector $n \cdot \mathbf{u}_0$. If $\mathbf{q}$ is a state vector with $\mathbf{q}[i] \geq 1$ we say that i is *lit* in $\mathbf{q}$. If local state i has a sending transition on action a we say that a is *enabled* from i; if i is lit in $\mathbf{q}$ we also say that a is *enabled* from $\mathbf{q}$.

With each action a we can associate the *broadcast matrix* of action a in P, denoted $\mathbf{M}_a$, which is an $|S| \times |S|$ matrix with $\mathbf{M}_a(k, m) = 1$ if $(s_k, a??, s_m) \in R$, and $\mathbf{M}_a(k, m) = 0$ otherwise. Then, the parallel composition of n copies of a broadcast protocol $B = (S, s_0, L, R)$ is the *counter system* $B^n = ([n]^{|S|}, n \cdot \mathbf{u}_0, A, T)$, where $(\mathbf{q}, a, \mathbf{q}') \in T$ iff there exists $(s_i, a!!, s_j) \in R$ with $\mathbf{q}[i] \geq 1$ and $\mathbf{q}'$ is obtained from $\mathbf{q}$ in the following way:

$$\begin{aligned} \mathbf{p} &= \mathbf{q} - \mathbf{u}_i \\ \mathbf{p}' &= \mathbf{p} \cdot \mathbf{M}_a \\ \mathbf{q}' &= \mathbf{p}' + \mathbf{u}_j, \end{aligned}$$

That is, given the current state vector $\mathbf{q}$, the state vector $\mathbf{p} = \mathbf{q} - \mathbf{u}_i$ corresponds to the sending process leaving the state i. The state vector $\mathbf{p}' = \mathbf{p} \cdot \mathbf{M}_a$ describes the situation after the other processes take the receiving transition on a. Finally, $\mathbf{q}' = \mathbf{p}' + \mathbf{u}_j$ is the resulting state-vector after the sending process arrives its target location.

An *execution* of B^n is a sequence $\mathbf{q}_0, a_1, \mathbf{q}_1, a_2, \ldots, a_m, \mathbf{q}_m$ such that $(\mathbf{q}_i, a_{i+1}, \mathbf{q}_{i+1}) \in T$ for every $0 \leq i \leq m-1$. We say that the execution is *based on* the sequence of actions $a_1, \ldots, a_m$ and that $B^n(a_1 \ldots a_m) = \mathbf{q}_m$. We say that a word $w \in A^*$ is *feasible* in B^n if there is an execution of B^n based on w. The *language* of B^n, denoted $L(B^n)$, is the set of all feasible words for n processes, and the language of B, denoted $L(B)$, is the union of $L(B^n)$ over all $n \in \mathbb{N}$. A broadcast protocol B is said to have a *cutoff* $k \in \mathbb{N}$ if for any $k' > k$ it holds that $L(B^k) = L(B^{k'})$. We say that a broadcast protocol with no hidden states is *fine* if it has a cutoff. We use $\mathcal{F}$ for the class of fine BPs. Let B_1 and B_2 be two BPs. We say that B_1 and B_2 are *equivalent* iff $L(B_1) = L(B_2)$. Note that unlike the case of DFAs, there is no unique minimal fine BP, as shown by the example in Fig.1.

B. Passive Learning and Characteristic Sets

A *sample* for a BP B is a set $\mathcal{S}$ of triples in $A^* \times \mathbb{N} \times \mathbb{B}$ where $\mathbb{B} = \{\mathrm{T}, \mathrm{F}\}$. A triple (w, n, T) is consistent with a BP B if w is feasible in B^n. Similarly, a triple (w, n, F) is consistent

[2]Some models of BPs also consider rendezvous transitions, usually labeled with $a!$ and $a?$, but these can be simulated by broadcast transitions with a quadratic blowup in the number of states.

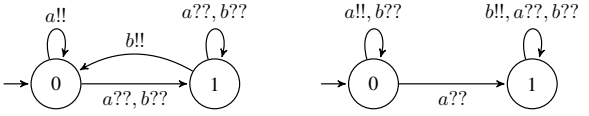

Fig. 1. Two non-isomorphic fine BPs for the same language: $a(a \cup b)^*$.

with a BP B if w is infeasible in B^n. A sample is *consistent* with a BP B if all triples in it are consistent with B.

We consider the following problems related to passive learning a class $\mathcal{C}$ of broadcast protocols.

Definition II.1 (Consistency)**.** *Given a sample $\mathcal{S}$ and $k \in \mathbb{N}$ determine whether there exists a BP $B \in \mathcal{C}$ consistent with $\mathcal{S}$ with at most k states.*

Definition II.2 (Inference)**.** *Devise an algorithm that given a sample $\mathcal{S}$ that is consistent with some BP in $\mathcal{C}$ returns a BP $B \in \mathcal{C}$ that is consistent with $\mathcal{S}$. We refer to such an algorithm as an* inference algorithm*.*

Definition II.3 (Teachability)**.** *We say that $\mathcal{C}$ is* teachable *if there exists an inference algorithm $\mathfrak{A}$ for $\mathcal{C}$ such that for every $B \in \mathcal{C}$ it is possible to construct a sample $\mathcal{S}_B$ such that $\mathfrak{A}$ will return a BP B' for which $L(B') = L(B)$ when applied to any sample $\mathcal{S}$ that subsumes $\mathcal{S}_B$ and is consistent with B. In this case we refer to the sample $\mathcal{S}_B$ as the* characteristic set *for B.*

Definition II.4 (Polynomial CS)**.** *Given $\mathcal{C}$ is teachable, determine whether every $B \in \mathcal{C}$ has a characteristic set of size polynomial in B.*

We use the number of states in B as a measure for its size. The size of a characteristic set is measured by the sum of the lengths of the words in it.

III. Consistency is NP-Hard for fine BPs

We show below that consistency is NP-hard even for fine BPs. We note that hardness is expected since DFA consistency is NP-hard, but it does not directly follow from hardness of DFA consistency. This is since a DFA is not a special case of a fine BP. Below we give a direct proof for the NP-hardness of BP consistency. In appendix A we give an alternative proof, that goes via a reduction from DFA consistency.

Regarding completeness, we note that given a BP B and a pair $(w, n) \in A^* \times \mathbb{N}$ it is possible to check in polynomial time whether w is feasible in B^n by developing the state vector $n \cdot \mathbf{u}_0$ along the word w in B. Consequently, and since a BP with m states over set of actions A can be described in size polynomial in m and $|A|$, if m is given in unary then BP-consistency is NP-complete.

Theorem III.1. *The consistency problem for fine BPs is NP-hard.*

Proof. The proof is by reduction from the problem of *all-eq-3SAT* and is inspired by a recent proof on the hardness of DFA consistency [29]. The problem of *all-eq-3SAT* asks given an all-eq-3CNF formula φ whether it has a satisfying assignment. Where an all-eq-3CNF formula is a 3CNF formula where in each clause either all literals are positive or all literals are negative. This problem is known to be NP-complete.

Let $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an all-eq-3CNF formula over a set of variables $V = \{x_1, x_2, \ldots, x_n\}$. We take the number k to be $n+m+4$. For the alphabet of the sample we take $A = \{a_i \mid 1 \le i \le m+n\} \cup \{a, b, c, d, e\}$. We devise a sample $\mathcal{S}$ using four disjoint sets of words: P_1, P_2, N_1, N_2. The sample then consists of the triples $\{(w, 1, \text{T}) \mid w \in P_1\} \cup \{(w, 2, \text{T}) \mid w \in P_2\} \cup \{(w, 1, \text{F}) \mid w \in N_1\} \cup \{(w, 2, \text{F}) \mid w \in N_2\}$. The sets are defined as follows

$$
\begin{aligned}
P_1 &= \{a_1 a_2 \ldots a_{m+n} aa\} \cup \{bcc\} \\
P_2 &= \{a_1 a_2 \ldots a_i bcdd \mid 1 \le i \le m, \ \ C_i \text{ is positive}\} \cup \\
&\quad \{a_1 a_2 \ldots a_i bcee \mid 1 \le i \le m, \ \ C_i \text{ is negative}\} \\
N_1 &= \{a_1 a_2 \ldots a_i a_j \mid 1 \le i < m+n, \ j \ne i+1\} \cup \\
&\quad \{a, ba, bb, c, d, e, bca, bcd, bce\} \\
N_2 &= \{a_1 \ldots a_i b a_j a_{j+1} \ldots a_{m+n} \mid i \le j \le m\} \cup \\
&\quad \{a_1 \ldots a_i b a_{j+1} \ldots a_{m+n} \mid i \le m, j > m, x_j \notin C_i\} \cup \\
&\quad \{a_1 a_2 \ldots a_i bcde \mid 1 \le i \le m\} \cup \\
&\quad \{a_1 a_2 \ldots a_i bced \mid 1 \le i \le m\} \cup \\
&\quad \{a_1 a_2 \ldots a_i x \mid 0 < i \le m+n, \ x \in \{c, d, e\}\} \cup \\
&\quad \{a_1 a_2 \ldots a_i a \mid 0 < i < m+n\} \cup \\
&\quad \{a_1 \ldots a_i b a_{j+1} \ldots a_k a \mid i \le m, \ j \ge m, \ k < m+n\}
\end{aligned}
$$

First we note that for every BP consistent with P_1, if for some $\sigma \in A$ there is a word in P_1 with $\sigma\sigma$ as an infix, there must be a state with self-loop on σ, since there is only one processes in the system. The first set in N_1 prescribes that in any consistent BP, with a single process, the action a_i may only be followed by a_{i+1}. Together with P_1 this guarantees that any BP consistent with $\mathcal{S}$ has at least $m+n+1$ states, one for each of the a_i actions, and one for the c action. Looking at the third and forth sets in N_2, we entail that in any consistent BP, d and e cannot be fired from the same state, nor from any of the previously identified states. Looking at the fifth set in N_2 we entail that in any consistent BP, a has a state of its own. Thus a consistent BP requires at least $k = m + n + 4$ states.

Suppose there is a satisfying assignment for φ. We can construct a BP with k states consistent with the sample as shown in Fig. 2. The states $1, 2, \ldots,$ M correspond to the clauses, the states 1', 2', ..., N' correspond to the variables, the states T and F correspond to truth assignments T and F, and there are two additional states C and A. The initial state 1 has transition with b!! to C which has a self transition on c!!. State A has a self-loop on a!!. A word $a_1 \ldots a_i$ for $i < m$ corresponds to the clause C_{i+1}. From each state I corresponding to a clause C_i we add a b?? transition to a state J' corresponding to a satisfying literal in the clause of C_i. All states J' corresponding to variables x_j with a T assignment have a c?? response to state T making the words $a_1 a_2 \ldots a_i bcdd$ feasible in B^2 for every positive clause C_i for which $x_j \in C_i$ (conditioned a corresponding b?? transition from C_i to x_j exists). All states J' corresponding to variables x_j with a F assignment have

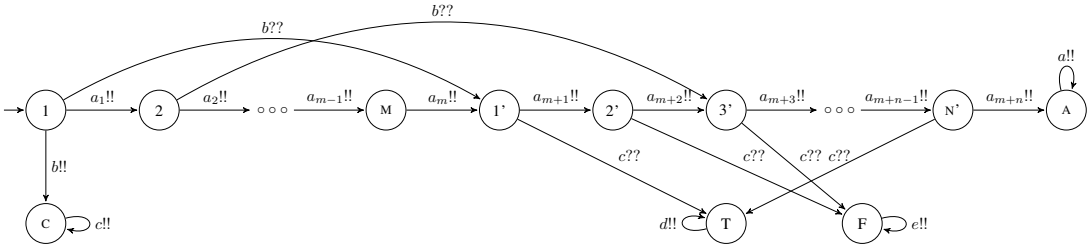

Fig. 2. A BP consistent with $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ where $C_1 = (x_1 \vee x_2 \vee x_5)$, $C_2 = (\overline{x_2} \vee \overline{x_3} \vee \overline{x_7})$ and the satisfying assignment $x_1 = \text{T}, x_2 = \text{F}, x_3 = \text{F}, \ldots, x_n = \text{T}$. Responses that are not shown are self-loops.

a $c??$ response to state F making the words $a_1 a_2 \ldots a_i bcee$ feasible in B^2 for every negative clause C_i for which $\overline{x_j} \in C_i$ (again, conditioned a corresponding $b??$ transition from C_i to x_j exists). The first set of N_2 guarantee that $b??$ from a state corresponding to certain clause does not land in a state corresponding to another clause. The second set of N_2 guarantee that $b??$ from a state corresponding to a certain clause does not land in a state corresponding to a variable that does not belong to the clause. Similar arguments show that if φ has no satisfying assignments then no BP with less than k states that agrees with the sample exists. □

We note that the BP constructed in the proof of Thm.III.1 has no hidden states and a cutoff (the cutoff is 2). It follows that BP consistency is NP-hard also for the class $\mathcal{F}_2$ of fine BPs with a cutoff 2.

IV. Properties of Broadcast Protocols

Below we establish some properties regarding broadcast protocols that will be useful in devising the learning algorithm.

Lemma IV.1 (Prefix-closedness and monotonicity)**.** *If B is a BP then $L(B)$ is prefix-closed. If $w \in A^*$ is feasible in B^k then w is feasible in B^ℓ, that is, $L(B^k) \subseteq L(B^\ell)$ for all $\ell > k$.*

Proof. Prefix-closedness holds since if $a_1 a_2 \ldots a_n$ is feasible, then for every $1 \leq i \leq n$ the action a_i is feasible after reading the prefix $a_1 a_2 \ldots a_{i-1}$.

For monotonicity, for two state vectors $\mathbf{p}$ and $\mathbf{q}$ we say that $\mathbf{q} \geq \mathbf{p}$ if for every $i \in [|S|]$ we have that $\mathbf{q}[i] \geq \mathbf{p}[i]$. Note that if $a \in A$ is enabled in $\mathbf{p}$ then it is also enabled in $\mathbf{q}$. Let $w = a_1 a_2 \ldots a_m$ and let $\mathbf{p}_0, a_1, \mathbf{p}_1, a_2, \ldots, a_m, \mathbf{p}_m$ be the execution of B^k on w. We can construct an execution $\mathbf{q}_0, a_1, \mathbf{q}_1, a_2, \ldots, a_m, \mathbf{q}_m$ of B^ℓ by induction on the length of w such that for every i we have that $\mathbf{q}_i > \mathbf{p}_i$ entailing w is feasible in B^ℓ as well. □

Lemma IV.2 (Step by step progress)**.** *Let $w \in A^*$, $a \in A$, and $m < n$. If $w \in L(B^m)$ and $wa \notin L(B^m)$ yet $wa \in L(B^n)$, then $wa \in L(B^{m+1})$.*

Proof. Suppose $w \in L(B^m)$ and $wa \notin L(B^m)$ yet $wa \notin L(B^{m+1})$. I.e., neither one of the processes that take some sending transitions when executing w in B^{m+1}, nor one of the processes only taking receiving transitions (including the additional process compared to B^m) is in the state that has the sending transition on a. Since any further additional process will behave in the same way as the additional process in B^{m+1}, wa can never become feasible. Contradicting that it is feasible in B^n. □

Recall that fine BPs have no canonical minimal representation, in the sense that, as shown in Fig.1 there could be two non-isomorphic BPs for the same language. The fact that there is no canonical minimal representation is often the main obstacle in obtaining a learning algorithm. The following important lemma asserts, that while two minimal fine BPs may not be isomorphic there is a tight correspondence between them.

Since every action can be fired from a unique state, and from every state at least one action can be fired, in a minimal BP the set of actions is partitioned between states, and if there is a state s_1 in B_1 whose set of sending transitions is $A' = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ then there should be a state s_2 in B_2 for which the set of sending transitions is exactly A'. So we can define such a mapping between the states of two minimal BPs, and it must be that on every word w if $\mathbf{p}_w$ and $\mathbf{q}_w$ are the state vectors B_1 and B_2 reach after reading w, resp., then if state s_1 is lit in $\mathbf{p}_w$ then the corresponding state s_2 (that agrees on the set of sending actions) is lit in $\mathbf{q}_w$. In the following we use $f^{\mathsf{act}}(s) = A'$ if A' is the set of sending actions from s.

Lemma IV.3 (Relation between two minimal equivalent BPs)**.** *Let B_1 and B_2 be two minimal BPs with sets of states S_1 and S_2 such that $L(B_1) = L(B_2)$. Then for every $m \in \mathbb{N}$ it holds that $L(B_1^m) = L(B_2^m)$ and there exists a bijection $h : S_1 \to S_2$ satisfying that $f^{\mathsf{act}}(s) = f^{\mathsf{act}}(h(s))$ for any $s \in S_1$; and for any $m \in \mathbb{N}$ and $w \in A^*$*

- *if reading w in B_1^m leads to global state $\boldsymbol{p}_w$ and reading w in B_2^m leads to global state $\boldsymbol{q}_w$ then for every state i, $\boldsymbol{p}_w[i]$ is lit if and only if $\boldsymbol{q}_w[h(i)]$ is lit.*

Proof. The proof is by induction on the number m of interacting processes and the length of w. For $m = 1$ and $w = \epsilon$, we have $\mathbf{p}_\epsilon = \mathbf{u}_i$ and $\mathbf{q}_\epsilon = \mathbf{u}_j$ for some $i, j \in \{0, \ldots, |S|-1\}$ where $n = |B_1| = |B_2|$. It follows that s_i and s_j are the initial states of B_1 and B_2, resp. Thus, $f^{\mathsf{act}}(s_i) = f^{\mathsf{act}}(s_j)$, as otherwise there is an action a that is enabled in B_1^1 and not in B_2^1 or vice versa (and therefore also in any B_1^m and B_2^m

with $m \geq 1$), contradicting that $L(B_1) = L(B_2)$. Hence we can set $h(i) = j$ and the claim holds.

For $m = 1$ and $w = ua$ for $u \in A^*$ and $a \in A$, by the induction hypothesis we know that for every state i, $\mathbf{p}_w[i]$ is lit if and only if $\mathbf{q}_w[h(i)]$ is lit. Since we have only one process all global states are unit vectors. Thus, it must be that $\mathbf{p}_u = \mathbf{u}_i$ and $\mathbf{q}_u = \mathbf{u}_j$ for some $i, j \in \{0, \ldots, |S|-1\}$. That is, i and j are the indices of the states in B_1, B_2 from which a is enabled, resp. Let $i', j' \in \{0, \ldots, |S|-1\}$ be the indices of the states that B_1, B_2 reach after reading ua. Then $\mathbf{p}_{ua} = \mathbf{u}_{i'}$ and $\mathbf{q}_{ua} = \mathbf{u}_{j'}$. Since this is true for any action b that is feasible in B_1 or B_2 after u, $h(i') = j'$ satisfies the claim. Note that this also shows that any word w that is feasible in B_1^1 is also feasible in B_2^1 and vice versa.

Assume the claim holds for m we show it holds for $m+1$. Consider a word w. Let $\mathbf{p}_w$ and $\mathbf{q}_w$ be the state vectors B_1^m and B_2^m reach after reading w. Then by the induction hypothesis, $L(B_1^m) = L(B_2^m)$ and for every state i we have that $\mathbf{p}_w[i]$ is lit if and only if $\mathbf{q}_w[h(i)]$ is lit. Let $\mathbf{p}'_w$ and $\mathbf{q}'_w$ be the state vectors B_1^{m+1} and B_2^{m+1} reach after reading w. Then $\mathbf{p}_w$ and $\mathbf{p}'_w$ are the same for every $i \in \{0, \ldots, |S|-1\}$ but one (and similarly for the $\mathbf{q}$'s). For all of these indices the claim holds by the induction hypothesis. Let j be the index with $\mathbf{p}_w[j] \neq \mathbf{p}'_w[j]$, i.e., $\mathbf{p}'_w[j] = \mathbf{p}_w[j] + 1$. If $\mathbf{p}_w[j] \geq 1$, then by the induction hypothesis for every state i we have that $\mathbf{p}_w[i]$ is lit if and only if $\mathbf{q}_w[h(i)]$ is lit. In particular, this holds for j.

Otherwise, $\mathbf{p}'_w[j] = 1$, which implies that there is at least one action a enabled from $\mathbf{p}'_w$ that is not enabled from $\mathbf{p}_w$. By induction hypothesis, $L(B_1^m) = L(B_2^m)$, and therefore we know that a is also not enabled from $\mathbf{q}_w$. Moreover, by Lemma IV.2, if $w \in L(B_2^m)$, $wa \in L(B_2)$ and $wa \notin L(B_2^m)$, then $wa \in L(B_2^{m+1})$. Thus, there must be a local state k enabling a such that $\mathbf{q}_w[k] = 0$ and $\mathbf{q}'_w[k] = 1$. Hence, letting $h(j) = k$ satisfies the claim. □

V. Inferring a Broadcast Protocol from a Sample

Let $\mathcal{S}$ be a sample. The inference algorithm $\mathfrak{I}$ we devise will construct a BP that agrees with $\mathcal{S}$, such that moreover, if $\mathcal{S}$ subsumes a characteristic set for a fine BP B then the inference algorithm will return a minimal BP equivalent to B.

Let $A_{\mathcal{S}}$ be the set of actions that appear in the sample at least once as feasible. In order to return a BP with no hidden states, we allow the resulting BP to have more actions than in $A_{\mathcal{S}}$. We use A for the set of actions used by the constructed BP, S for the set of states of the constructed BP, and s_0 for its initial state.

We will construct a set of constraints that define a BP. More precisely, we will construct constraints regarding the behavior of three partial functions $f^{\mathsf{st}} : A \to S$, $f^{!!} : A \to S$, and $f_a^{??} : S \to S$ for every $a \in A$ so that any valuation of these functions that satisfies the constraints implement a BP consistent with the sample. Formally, we say that functions $f^{\mathsf{st}}, f^{!!}, \{f_a^{??} \mid a \in A\}$ *implement* the broadcast protocol $B = (S, s_0, L, R)$ if for every $(s_i, a!!, s_j) \in R$ we have $f^{\mathsf{st}}(a) = s_i$ and $f^{!!}(a) = s_j$ and for every $(s_i, a??, s_j) \in R$ we have $f^{??}(s_i, a) = s_j$. We also use $f^{\mathsf{act}}(s) = A'$ if $A' = \{a \in A \mid f^{\mathsf{st}}(a) = s\}$.

We turn to introduce some terminology regarding the sample. Let $\mathcal{P}_i$ be the set of words $\{w \mid (w, i, \mathrm{T}) \in \mathcal{S}\}$. Let $\mathcal{N}_i$ be the set of words $\{w \mid (w, i, \mathrm{F}) \in \mathcal{S}\}$. We note that it follows from lemma IV.1 that if w is in $\mathcal{P}_i$ then it is feasible in B^j for every $j \geq i$. Similarly if w is in $\mathcal{N}_i$, then w is infeasible in B^j for every $j \leq i$.

We define a relation between actions as follows. Let $a, b \in A_{\mathcal{S}}$. We say that $a \#_{\mathcal{S}} b$ if there exist a word $w \in A_{\mathcal{S}}^*$ and naturals $n' \geq n$ such that $(wa, n, \mathrm{T}) \in \mathcal{S}$ and $(wb, n', \mathrm{F}) \in \mathcal{S}$ or vice versa (switching the roles of a and b). Following the observation in IV.3, $a \#_{\mathcal{S}} b$ means that the sample $\mathcal{S}$ has information contradicting that a and b are enabled from the same state.

1) Our first constraints are therefore that for every $a, b \in A$ such that $a \#_{\mathcal{S}} b$ it holds that $f^{\mathsf{st}}(a) \neq f^{\mathsf{st}}(b)$.
2) Since we assume there are no hidden states, we need to demand that at least one action is associated with every state. We collect the set of states that are the target of a sending or a receiving transition, and add to them the initial state, as follows:

$$\begin{aligned} S = \; & \{f^{\mathsf{st}}(a) \colon \exists b \in A.\ f^{!!}(b) = f^{\mathsf{st}}(a)\} \cup \\ & \{f^{\mathsf{st}}(a) \colon \exists b, c \in A.\ f_c^{??}(f^{\mathsf{st}}(b)) = f^{\mathsf{st}}(a)\} \cup \\ & \{f^{\mathsf{st}}(a) \colon a \in \mathcal{P}_1\} \end{aligned}$$

   Note that the last set refers to the initial state since it considers a word of length one, that must be fired from the initial state. Now we demand that each state has an action:

$$\forall s \in S : \exists a \in A : f^{\mathsf{st}}(a) = s.$$

3) The rest of the constraints are gathered by scanning the words first by length. For every word of length one, i.e. action a, if for some i, $a \in \mathcal{P}_i$ then we add $f^{\mathsf{st}}(a) = s_0$, and if $a \in \mathcal{N}_i$ then we add $f^{\mathsf{st}}(a) \neq s_0$.
4) Next, we scan recursively for every word w for the minimal i such that the word is in $\mathcal{P}_i$ or $\mathcal{N}_i$. For the base case $i = 0$ all words are infeasible, and there are no processes in the system, so no requirement is added.
   For the sake of readability we spell the case of $i = 1$ as well. Let $w \in \mathcal{P}_1 \cup \mathcal{N}_1$ and let $w = a_1 a_2 \ldots a_m$. We define $m + 1$ variables $p_0, p_1, \ldots p_m$. The variable p_k indicates the state the single process reaches after the system reads $a_1 \ldots a_k$, and p_0 indicates the initial state of the process. To this aim we add the following constraint

$$(p_0 = s_0) \wedge \textstyle\bigwedge_{0 \leq \ell < m} \left(p_{\ell+1} = f^{!!}(a_{\ell+1})\right)$$

   Next, if $w \in \mathcal{P}_1$ then we add the following requirement:

$$\bigwedge_{0 \leq \ell < m} \left(p_\ell = f^{\mathsf{st}}(a_{\ell+1})\right)$$

   This requires that the next letter $a_{\ell+1}$ to be executed is enabled in the state the process reached after $a_1 a_2 \ldots a_\ell$ was executed.
   If $w \in \mathcal{N}_1$ then we add the following requirement

$$\bigvee_{0 \leq \ell < m} \left(p_\ell \neq f^{\mathsf{st}}(a_{\ell+1})\right)$$

This requires that at least one of the letters in the word is not enabled in the state the process reached, implying the entire word is infeasible with one process.

5) For the induction step $i > 1$, let $w \in \mathcal{P}_i \cup \mathcal{N}_i$ and assume $w = a_1 a_2 \dots a_m$. We define $i(m+1)$ variables $p_{1,0}, p_{2,0}, \dots p_{i,m}$. The variable $p_{j,k}$ indicates the state the j-th process reaches after the system reads $a_1 \dots a_k$. Accordingly, we set $p_{j,0} = s_0$ for every $1 \leq j \leq i$. The state of the processes after reading the next letter, a_{l+1}, depends on their state after reading a_l.
Let $w \in \mathcal{P}_i$ and let $w = a_1 a_2 \dots a_m$, we add the constraint $\psi_{w,i}$ defined as follows.

$$\psi_{w,i} = \bigwedge_{1 \leq \ell \leq m} \left(\bigvee_{1 \leq j \leq i} \left(\left(p_{j,\ell-1} = f^{\mathsf{st}}(a_\ell) \right) \wedge \varphi_{j,\ell} \right) \right)$$

where

$$\varphi_{j,\ell} = \left(\begin{array}{c} p_{j,\ell} = f^{!!}(a_\ell) \quad \wedge \\ \bigwedge_{\substack{1 \leq j' \leq i \\ j' \neq j}} p_{j',\ell} = f^{??}(p_{j',\ell-1}, a_\ell) \end{array} \right)$$

Intuitively, $\psi_{w,i}$ requires that for every letter a_ℓ of w one of the processes, call it j, reached a state from which a_ℓ is enabled. The formula $\varphi_{j,\ell-1}$ states that the j-th process took the sending transition on $a_{\ell-1}$ and the rest of the processes took the respective receiving transition.
Let $w \in \mathcal{N}_i$ and let $w = a_1 a_2 \dots a_m$. We then add the following requirement

$$\bigvee_{0 \leq \ell < m} \left(\psi_{w[..\ell],i} \wedge \bigwedge_{1 \leq j \leq i} \left(p_{j,\ell} \neq f^{\mathsf{st}}(a_{\ell+1}) \right) \right)$$

where $w[..\ell]$ denotes the ℓ'th prefix of w, namely $a_1 a_2 \dots a_\ell$, and we let $\psi_{\epsilon,i} = \mathrm{T}$ for every i.
Intuitively, if w is infeasible with i processes, then there exists a (possibly empty) prefix $w[..\ell]$ which is feasible with i processes, therefore $\psi_{w[..\ell],i}$ holds, while $w[..\ell{+}1]$ is infeasible, meaning none of the i processes is in a state where $a_{\ell+1}$ is enabled.

Theorem V.1. *Let $\mathcal{S}$ be a sample that is consistent with some fine BP. Let $\Psi_{\mathcal{S}}$ be the prescribed constraints with respect to $\mathcal{S}$. Let B be a BP that satisfies $\Psi_{\mathcal{S}}$. Then B is a BP consistent with $\mathcal{S}$.*

Proof. We prove that if $w \in \mathcal{P}_i$ then w is feasible in B^i, and if $w \in \mathcal{N}_i$ then w is infeasible in B^i by induction first on the length of w and then on i. For w of length 1, this holds by the constraints in item (3). Let $w = a_1 a_2 \dots a_n \in \mathcal{P}_i$. If $i = 1$ then this holds by induction on w thanks to constraint (4). Next we consider words in the sample of the form w that are in $\mathcal{P}_i \cup \mathcal{N}_i$. If w is already in $\mathcal{P}_{i-1}$ then by the induction hypothesis it is already feasible for $i-1$ processes in the constructed BP, and by Lemma.IV.1, it is also feasible with i processes. Otherwise, $w \in \mathcal{P}_i \setminus \mathcal{P}_{i-1}$. In this case, constraint (5) makes sure that every prefix of w is feasible with i processes by going letter by letter, and requiring that for the next letter a_ℓ one of the i processes reached the state enabling a_ℓ after reading the prefix up to $a_{\ell-1}$.

If $w \in \mathcal{N}_i$ then w is infeasible with i processes. In this case, there exists a letter a_ℓ for $1 \leq \ell \leq m$ such that while $w[..\ell-1]$ is feasible, a_ℓ is not enabled from any of the states that the i processes reach after reading (the possibly empty) prefix $w[..\ell-1]$. This is exactly what constraint (5) stipulates. □

Corollary 1. *There exists an inference algorithm $\mathfrak{I}$ for the class of fine BPs.*

VI. The class of fine BPs is teachable

In order to show that the class of fine BPs is teachable we have to show that every BP B in the class can be associated with a sample $\mathcal{S}_B$ so that there exists an inference algorithm $\mathfrak{A}$ that when applied to any sample $\mathcal{S}$ that subsumes $\mathcal{S}_B$ and is consistent with B, returns a minimal fine BP that is equivalent to B. Recall that when the class is teachable, the sample $\mathcal{S}_B$ associated with B is called a characteristic set.

We start by describing in Sec.VI-A a procedure $\mathfrak{G}$ that generates a sample $\mathcal{S}_B$ from a fine BP B. Then in Sec.VI-B we prove that an inference algorithm $\mathfrak{A}$ can correctly infer a minimal BP B' equivalent to B from any sample subsuming $\mathcal{S}_B$.

A. Generating a Characteristic Set

The characteristic set generation algorithm $\mathfrak{G}$ builds a sequence of trees $\mathcal{T}_i$ starting with $i = 0$ and incrementing i by one until $\mathcal{T}_{i+1} = \mathcal{T}_i$. The edges of the tree are action symbols. The name of a node is taken to be the unique sequence of actions w that leads to it. Thus the root is named ε and a child of a node $w \in A^*$ is named wa for some $a \in A$. A node $w \in A^*$ in tree $\mathcal{T}_i$ is annotated with $\mathbf{p}_{w,i}$, the state-vector B^i reaches when reading w, if w is feasible in B^i, and with the special symbol $\perp$ otherwise. We call a node in the tree *positive* if it is annotated with a state-vector, and *negative* otherwise. All nodes are either leaves or have exactly $|A|$ children. Negative nodes are always leaves.

The tree $\mathcal{T}_0$ consists of only a root ε and is annotated with the state vector of all zeros. The tree $\mathcal{T}_{i+1}$ is constructed from the tree $\mathcal{T}_i$ by first re-annotating all its nodes: The annotation of a positive $\mathbf{p}_{w,i}$ is replaced by $\mathbf{p}_{w,i+1}$, a negative node w in $\mathcal{T}_i$ may become positive in $\mathcal{T}_{i+1}$ (if w is feasible with $i+1$ processes) and will be annotated accordingly with $\mathbf{p}_{w,i+1}$. Then we check, from every positive node, whether further exploration is needed. A positive node will be declared a leaf if it is of the form va and it has an ancestor u, a prefix of v, for which $\mathbf{p}_{u,i+1} = \mathbf{p}_{v,i+1}$. Otherwise its $|A|$ children are constructed. That is, once we reach a node whose state-vector is the same as one of its ancestors, we develop its children, but the children are not developed further.

The entire process terminates when $\mathcal{T}_{i+1} = \mathcal{T}_i$. Note that given the BP has a cutoff, such an i must exist. We use $\mathcal{T}$ for the last tree constructed, namely $\mathcal{T}_{i+1}$. The sample is then produced as follows. For $n \in [1..i+1]$, let $\mathcal{P}_n = \{(u, n, \mathrm{T}) \mid n$ is the minimal for which u is positive in $\mathcal{T}_n\}$, $\mathcal{N}_n = \{(u, n, \mathrm{F}) \mid$

u is the maximal for which u is negative in $\mathcal{T}_n\}$. Then the sample is the union of all these, i.e., $\mathcal{S}_B = \bigcup_{n=1}^{i+1} (\mathcal{P}_n \cup \mathcal{N}_n)$.

B. Proving that $\mathfrak{G}$ generates characteristic sets

We first note that for any state s of the original BP, there exists at least one node v in the tree where s is lit (i.e. the entry for the s in the state vector annotating the node is at least one).

Lemma VI.1. *Let $\boldsymbol{p}$ be a state vector that is reachable in B^m. Then for every shortest word w that reaches $\boldsymbol{p}$ in B^m there exists a node w in $\mathcal{T}_m$ such that $\boldsymbol{p}_w = \boldsymbol{p}$.*

Proof. The proof is by induction, first on m then on the length of w. For $m = 1$, the construction of the tree clearly guarantees that all states reachable with one process have a respective node in the tree.

Suppose $\mathbf{p}$ is reachable with w in B^m for $m > 1$. By Lemma IV.2, there exists a prefix u of w, such that u is feasible in B^{m-1}. By the induction hypothesis, u is a node of $\mathcal{T}_{m-1}$. Suppose $w = ua_1a_2 \dots a_n$. We can show by induction on $1 \leq n$ that $ua_1a_2 \dots a_i$ is a node of $\mathcal{T}_m$ for every $i \leq n$ by simply following the tree construction. □

Next we claim that if the sample $\mathcal{S}$ subsumes a characteristic set then $\#_{\mathcal{S}}$ induces an equivalence relation between the actions.

Lemma VI.2. *For two actions a and b define $a \sim_{\mathcal{S}} b$ iff it is not the case that $a \#_{\mathcal{S}} b$. If $\mathcal{S}$ subsumes a characteristic set then $\sim_{\mathcal{S}}$ is an equivalence relation.*

Proof. Clearly $\sim_{\mathcal{S}}$ is symmetric and reflexive. To see that it is transitive, we first refer to Lem. VI.1 to deduce that there exists a node v in $\mathcal{T}$ in which s is lit. Let m be the minimal for which v is in $\mathcal{T}_m$. It follows that for every action a that is feasible after v with m processes we have $(wa, n, \mathrm{T}) \in \mathcal{S}$ and for every action a that is infeasible after v with m processes we have $(wa, n, \mathrm{F}) \in \mathcal{S}$. Therefore, for any two actions we have $a \#_{\mathcal{S}} b$ iff a and b are not enabled from the same state, and $a \sim_{\mathcal{S}} b$ otherwise. Assume now that $a \sim_{\mathcal{S}} b$, $b \sim_{\mathcal{S}} c$. Then a and b are enabled from the same state, and b and c are enabled from the same state, implying a and c are enabled from the same state, i.e., $a \sim_{\mathcal{S}} c$ as required. □

Theorem VI.3. *Let B be a fine minimal BP, and let $\mathcal{S}_B$ be the sample generated for it as above. There is an inference algorithm $\mathfrak{A}$ such that if B' is the result of $\mathfrak{A}$ on when applied to any set subsuming $\mathcal{S}_B$ and consistent with B then B' is minimal and $L(B') = L(B)$.*

Proof. The inference algorithm $\mathfrak{A}$ we use to prove this claim runs in two steps. First it runs a variation $\mathfrak{I}'$ of the inference algorithm $\mathfrak{I}$ presented in Sec.V that turns the constraint (1) into an iff constraint. I.e. adding that $f^{\mathsf{st}}(a) = f^{\mathsf{st}}(b)$ unless $a \#_{\mathcal{S}} b$. If running $\mathfrak{I}'$ returns that there is no satisfying assignment then it runs $\mathfrak{I}$. In both cases Thm. V.1 guarantees that the returned BP is consistent with the given sample. Therefore $\mathfrak{A}$ is an inference algorithm.

Next we claim that if the given sample subsumes $\mathcal{S}_B$ then B', the resulting BP, is minimal. This holds since Lem. VI.2 ensures that $\#_{\mathcal{S}}$ defines the desired equivalence $\sim_{\mathcal{S}}$ between actions, and the revised constraint (1) guarantees that actions are not enabled from the same state if and only if the sample separates them. (Note that any word consistent with the BP cannot separate actions a and b if they are enabled from the same state.) Therefore $\mathfrak{I}'$ will not return that there is no satisfying assignment.

Next we note that by Lem. VI.1 for every state vector $\mathbf{p}$ that is reachable in B^m. And for every shortest word w that reaches $\mathbf{p}$ in B^m there exists a node w in $\mathcal{T}_m$ such that $\mathbf{p}_w = \mathbf{p}$. If $w = a_1a_2 \dots a_n$ then for each $1 \leq i \leq n$ one process took the sending transition $a_i!!$ and the rest of the processes responded with $a_i??$. Constraint (5) makes sure the assignment to f^{st}, $f^{!!}$ and $f^{??}$ respect all the possible options that enabled this, making sure that for every two options for enabling w that result in state vectors $\mathbf{p}_1$ and $\mathbf{p}_2$, resp., the same states are lit in both $\mathbf{p}_1$ and $\mathbf{p}_2$.

It follows that for any BP B' that adheres to the constraints there exists a mapping h between the states of B and B' satisfying the requirements of Lem. IV.3. Therefore $L(B) = L(B')$. □

Corollary 2. *The class of fine broadcast protocols is teachable.*

VII. Characteristic sets may be inevitably large

In this section we show that there exist fine BPs for which there is no characteristic set of polynomial size.

We start by showing that there exist fine BPs with cutoff of size quadratic in the size of the BP. We adapt a family of BPs used in [25] for showing a quadratic cutoff for BPs without the restriction of no-hidden states, and for a slightly different definition of cutoff (reaching a particular state). The adaptation for no-hidden states is seamless. To work with our definition of cutoff we needed to introduce some auxiliary states. The family is given in Fig.3.

The family is parameterized by three natural numbers m, n, and ℓ. The BP $B_{m,n,\ell}$ has n states in the lower loop, m states in the upper loop, and ℓ helper states (overall $n + m + \ell + 4$ states). We use $H??$ and $A??$ as a shortcuts for $\{h_i?? \mid i \in [1..\ell]\}$ and $\{a_i?? \mid i \in [1..n'{-}1]\}$, respectively. From all of the states except for M we assume $c??$ takes to the $\perp$ state, and $a_\top??$ takes all of the states to the $\top$ state (we didn't add these transitions to avoid clutter). For any action in $\{b_j : j \in [1..m]\}$ and any state, $b_j??$ is a self loop. One can see that traversing the lower loop requires at least n processes: one process for each of the a_i transitions, for $1 \leq i \leq n-1$, and one process for the H transition. Each H transition requires that one of the h_i transitions to be taken. The structure of the h_i sending transitions thus restrict the number of times N' can be reached to $\ell + 1$ (since it can be reached once without executing an h_i). In order to enable $a_\top$ there must be one process in state N' that sends $c!!$ and one process in state M responding to it. Therefore if n and m are co-prime, this can occur only

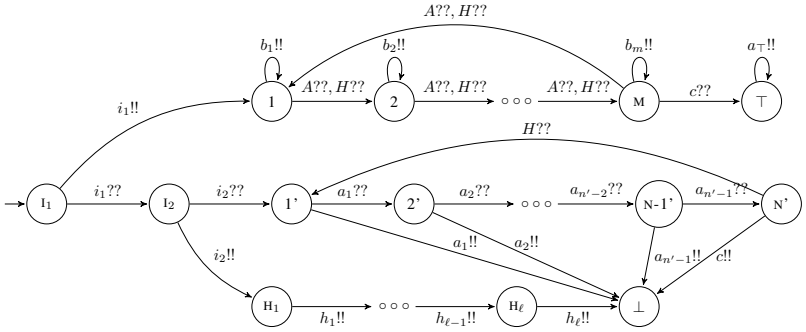

Fig. 3. The BP $B_{m,n,\ell}$ from a family of fine BP's with quadratic cutoffs.

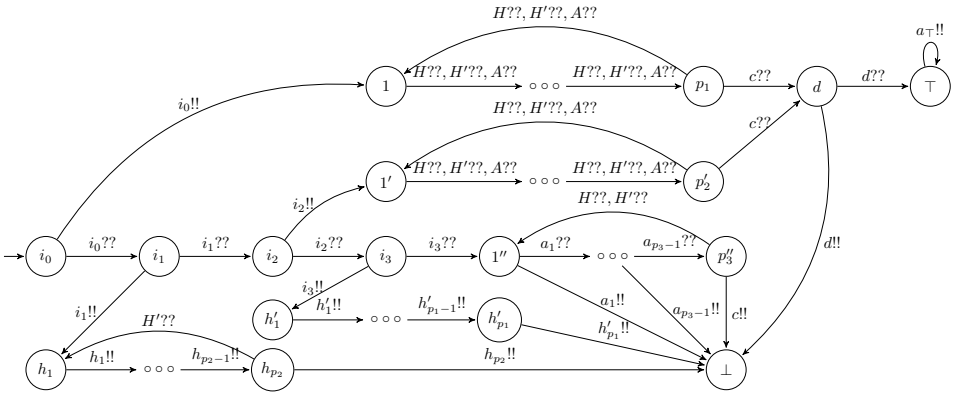

Fig. 4. The BP B_{p_1,p_2,p_3}. From state d responding to any action except for $d??$ will take us to the sink state. For state 1 to p_1 as well as $1'$ to p_2, their actions is a self-loop and responding to them will be a self-loop as well from any action. Responding to $c!!$ will take every one to the sink state except states p_1 and p_2.

once the lower loop is traversed m times (which requires m helper states). Thus an overall of $n \cdot m$ processes are required to make a word using action $a_\top$ feasible. Accordingly, the shortest word using the $a_\top$ action is of length at least $n \cdot m$. This shows that there exist no characteristic sets for this family of size less than quadratic in the size of the given BP.

To obtain larger lower bounds we can introduce additional loops. Let $p_1 < p_2 < p_3$ be three different primes, and consider the BP with three loops of sizes p_1, p_2 and p_3 given in Fig. 4. To allow the p_3 loop to be traversed at most $p_1 \cdot p_2$ times, we can introduce $p_1 + p_2$ helper auxiliary states and arrange them in a line of size p_1 and a loop of size p_2, respectively. We also need to introduce an additional state (d in the figure) to make the transition to $\top$ conditioned on the synchronization of the three loops, and an additional initialization state (i_3 in the figure).

To obtain an exponential size lower bound we extend this idea as follows. Instead of having two such loops, we have k such loops of sizes $p_1, p_2, \ldots, p_k$, where p_i is the i-th prime number. The overall size of the BP will be $2(p_1 + p_2 + \ldots + p_{k-1}) + p_k + 2(k-1) + k$. This is since we need $p_1 + p_2 + \ldots + p_k$ states for the main loops; $p_1 + p_2 + \ldots + p_{k-1}$ for the helper states (arranged in $k-2$ loops and one line); $2(k-1)$ states for the initialization (to place one process in each of the k main loops, and the $k-1$ helper loops/line); and k states for the synchronization of the main loops (these includes $d, \top, \bot$ in Fig.4).

Let $\{P_n\}_{n \in \mathbb{N}}$ be the family of fine BPs constructed as above with loops for all primes up to size n. Then the size of the BP P_n is quadratic in n. Since all primes are of size 2 at least, and since the number of primes of size n or less is $\Theta(n/\log n)$ the number of processes required to enable $a_\top$ is $2^{\Theta(n \log n)}$, and so is the size of a shortest word that includes the $a_\top$ action. Hence a word of length exponential in the size of the BP is required to be in the sample, entailing this family has no characteristic set of polynomial size.

Corollary 3. *There exists a family of fine BPs with no characteristic set of polynomial size.*

We note that the same family also shows that fine BPs can be exponentially smaller than the minimal DFA accepting the same language. Indeed, any DFA for the family P_n has to include some words that uses $a_\top$ as one of the letters, and the smallest such word is of size exponential in n. Therefore, the DFA must have at least that many states.

Corollary 4. *There exists a family of fine BPs for which the corresponding DFA is of exponential size.*

VIII. BPs ARE NOT POLYNOMIALLY PREDICTABLE

In this section we show that fine BPs are not polynomially predictable with membership queries. The learning paradigm of polynomial predictability of a class $\mathcal{C}$ can be explained as follows. The learner has access to an oracle answering two types of queries with regard to the target concept $C \in \mathcal{C}$: *membership queries* (MQ) and *draw queries* (DR). A membership query receives a word w as input and answers whether w is or is not in C. A draw query receives no inputs and returns a pair (w, b) where w is a word that is randomly chosen according to some probability distribution D and b is MQ(w). We assume some bound ℓ on the length of the relevant examples, so that D is a probability distribution on the set of relevant words. We assume the learner knows ℓ but D is unknown to her. At some point, the learner is expected to ask for a word whose membership it needs to predict, in which case it is handed a word w (drawn randomly according to the same distribution D) and it should then answer whether w is or is not in C. We say that the class $\mathcal{C}$ is *polynomially predictable* with membership queries, if given a bound s on the size of the target language, the mentioned bound ℓ on the length of relevant examples, and an accuracy parameter $\epsilon > 0$, there exists a learner that will classify the word to predict correctly with probability at least $(1 - \epsilon)$, after asking a number of queries that is polynomial in the size of the minimal BP of the target language.

We show that under plausible cryptography assumptions fine BPs (and hence BPs in general) are not polynomially predictable.

Theorem VIII.1. *Assuming the intractability of any of the following three problems: testing quadratic residues modulo a composite, inverting RSA encryption, or factoring Blum integers, fine broadcast protocols are not polynomially predictable with membership queries.*

Proof. The proof is via a reduction from the class $\mathcal{D}$ of intersection of DFAs, for which Angluin and Kharitonov have shown that $\mathcal{D}$ is not polynomially predictable under the same assumptions [6].

We show that given a predictor $\mathfrak{B}$ for fine BPs we can construct a predictor $\mathfrak{D}$ for the intersection of DFAs as follows. First, we show how to associate with any given set $D_1, D_2, \ldots, D_k$ of DFAs a particular BP B. Let $D_i = (\Sigma, Q_i, \iota_i, \delta_i, F_i)$ and assume without loss of generality that the states of the DFAs are disjoint and the DFAs are complete (i.e. from every state there is an outgoing transition on every letter). We construct a fine BP $B = (A, S, s, R)$ with cutoff $k + 1$ as follows (see Fig.5).

Let $Q = \cup_{i\in[1..k]} Q_i$. The set of states S is $Q \cup \{\bot, \mathrm{S}, \mathrm{C}, \mathrm{X}\} \cup \{\mathrm{G}_i, \mathrm{H}_i \mid i \in [1..k]\}$. The initial state is H_1. Let $\Sigma_h = \{h_j \mid j \in [1..k]\}$ and $\Sigma_g = \{g_j \mid j \in [1..k]\}$. The set of actions A is $\Sigma \cup \Sigma_h \cup \Sigma_g \cup \{\$, \bot, x, s\}$.

The transitions are as follows: for every $i \in [1..k]$ we have $(\mathrm{H}_i, h_i!!, \mathrm{G}_i)$. The receiving transitions on h_i are $(\mathrm{H}_i, h_i??, \mathrm{H}_{i+1})$ for every $i \in [1..k{-}1]$, and $(\mathrm{H}_k, h_k??, \mathrm{S})$. For state S, the transitions are as follows: $(\mathrm{S}, s!!, \mathrm{C})$, $(\mathrm{S}, s??, \mathrm{C})$. For every $\sigma \in \Sigma$ we have $(\mathrm{C}, \sigma!!, \mathrm{C})$. The transitions from the G_i's states are $(\mathrm{G}_i, s??, \iota_i)$ for every $i \in [1..k]$. For every $(q, \sigma, q') \in \delta_i$ we have $(q, \sigma??, q')$. For the $\bot$ state we have $(\bot, \bot!!, \mathrm{X})$ and $(\mathrm{X}, x!!, \mathrm{X})$. Finally, for every $i \in [1..k]$ and $q \in Q_i$ we have $(q,\$??, \mathrm{X})$ if $q \in F_i$ and $(q,\$??, \bot)$ otherwise. Response transitions that are not specified are self-loops.

To satisfy the requirement of no-hidden states, we can assume that the G_i states have a self loop on $g_i!!$, and every state q of one of the DFAs has a self loop on $q!!$ (not shown in the figure to avoid clutter). The set of actions used would be $A \cup Q$. Note that this structure enforces the prefix to be of the form $h_1u_1h_2u_2 \ldots h_ku_k$ where $u_i \in \{g_1, g_2, \ldots, g_i\}^*$ and k processes are required to enable h_k. In order to enable s an additional process is required. At this point any word v from Σ^* is feasible, and there will be exactly one process in the initial state of each of the DFAs, simulating its run on v. Upon the letter \$ only $\bot$ or x are feasible. Moreover, if there exists a DFA that rejects the word then $\bot$ is feasible, otherwise, no $\bot$ is feasible, only x.

Next we show how the predictor $\mathfrak{D}$ for the intersection of DFAs uses the predictor $\mathfrak{B}$ for BPs to satisfy his task. Note that $\mathfrak{D}$ has an oracle for the intersection of the DFAs $D_1, \ldots, D_k$ at his disposal, whereas $\mathfrak{B}$ expects answers regarding B. When $\mathfrak{B}$ asks a MQ about (w, n), i.e. whether w is feasible in B^n then $\mathfrak{D}$ behaves as follows.

Let w' be the word obtained from w by removing letters that are not in A. If w' is not a prefix of $h_1u_1h_2u_2 \ldots u_{k-1}h_ku_k s\Sigma^*\$\{\bot, x\}^*$ where $u_i \in \{g_1, g_2, \ldots, g_i\}^*$, or the number of $\bot$ letters exceeds one, then $\mathfrak{D}$ returns the answer "no" to $\mathfrak{B}$. Else, if w is such a prefix and w does not contain $\bot$, \$ or x, then $\mathfrak{D}$ returns to $\mathfrak{B}$ the answer "yes" iff $n > k$ or w' is a prefix of $h_1u_1h_2u_2 \ldots u_{n-1}h_nu_n$ where the u_i's are as above. Otherwise, w is such a prefix that contains \$. If $n \leq k$ we return "no". Otherwise, let v be the maximal infix of w' that is in Σ^*. The predictor $\mathfrak{D}$ asks a MQ about v. Assume it receives the answer b. If w has $\bot$, it answers "yes" iff b is "no". Otherwise it answers "yes".

We claim that $\mathfrak{B}$ receives correct answers. Indeed, the first couple of checks verify that w is feasible according to the construction that builds B from the given DFAs. For the last check, first note that if w does not contain $\bot$ then w is feasible. Otherwise, if w does contain $\bot$ then it is infeasible if v is accepted by all the DFAs and is feasible if at least one DFA rejects it.

The next ingredient is to show how draw queries of $\mathfrak{B}$ are simulated by $\mathfrak{D}$. When $\mathfrak{B}$ makes a DR query, then $\mathfrak{D}$ makes a DR query. Suppose it receives the answer (v, b). Then it passes $((h_1h_2 \ldots h_ksv\$\bot, k+1), \neg b)$ to $\mathfrak{B}$. Last, when $\mathfrak{B}$ asks for a word w to predict, then $\mathfrak{D}$ asks for a word v to predict and flips the answer of the prediction of $\mathfrak{B}$ on $(h_1h_2 \ldots h_ksv\$\bot, k{+}1)$. Note that if v is accepted by the intersection of the DFAs, and $w = h_1h_2 \ldots h_ksv\$\bot$ then w is infeasible, and if v is rejected then w is feasible. It follows that given $\mathfrak{B}$ classifies the predicted word correctly then so does $\mathfrak{D}$. Therefore, if $\mathfrak{B}$ is a polynomial predictor for fine BPs then $\mathfrak{D}$ is a polynomial

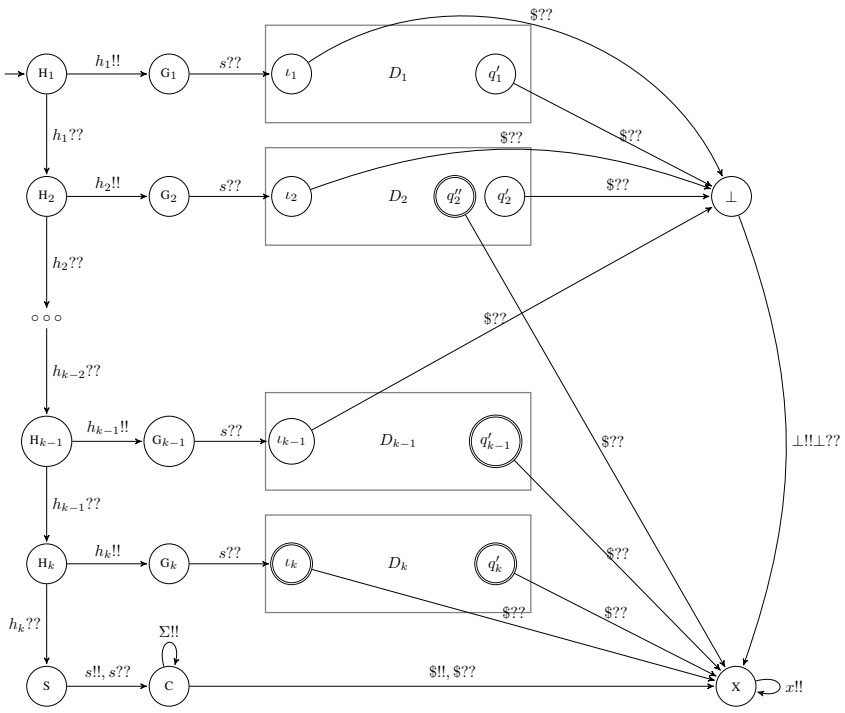

Fig. 5. A state s for which no sending transitions is shown, has a self-loop on $s!!$ (for a unique action s). Responses that are not shown are self-loops.

predictor for the intersection of DFAs. □

IX. Experimental Results

We have implemented our approach in a prototype tool. It uses the $Z3$ Theorem Prover (v4.12.2) as its underlying SMT solver via PyCharm, with Python 3.9. All experiments were run on a 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, with 32.0 GB RAM.

In the first experiment, we randomly generated BPs with no hidden states as follows. A number s between 2 to 10 is chosen randomly for the number of states. Then a number between 1 to 6 of actions are chosen and distributed randomly between the states. Then additional actions are added to states who weren't associated with an action to make sure the BP has no hidden states.

Using this method we randomly generated over 1000 BPs with no hidden states, and applied $\mathfrak{G}$ (see Sec. VI-A) to generate a characteristic set for them with a bound on 20 for the cutoff. Out of those, 422 successfully terminated. The rest either may have a bigger cutoff or no cutoff at all. On the produced characteristic sets, we ran the inference algorithm $\mathfrak{I}$ (see Sec. V, not assuming that the sample subsumes a characteristic set), limiting the SMT solving time to 2h.[3] The two graphs on top of Fig.6 show the results of this experiment. The x-axis shows the cutoff of the BP, the y-axis the number

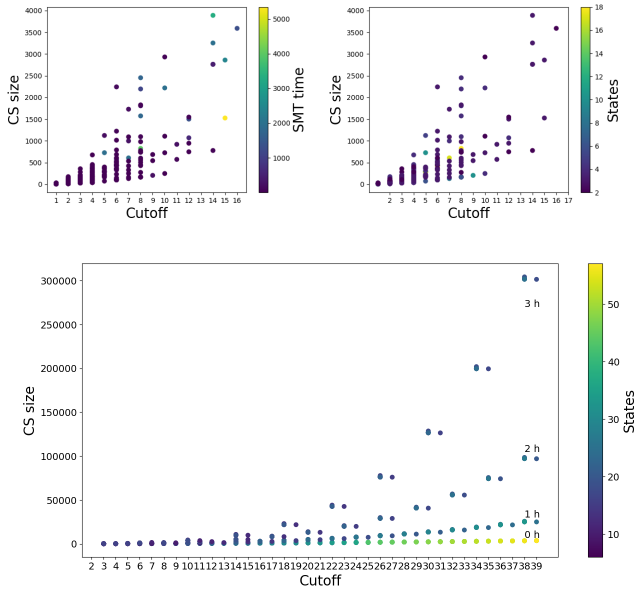

Fig. 6. Experimental Results

of words in the CS, the left color-bar shows the running time of the SMT solver in seconds, and the right color bar the number of states. Correctness of the obtained BP was verified by means of exhaustive membership queries. We observe that 92% of the instances run in less than 2 minutes, 4% run in

[3]Note that all variables in our SMT constraints are over finite domains with known size, implying that our constraints are decidable, and Z3 provides a complete decision procedure by reduction to SAT.

2 to 15 minutes. The hardest instance, which took about one and a half hour, has 18 states and a cutoff of 8. Other hard instances had a cutoff of 14 to 15.

In the second experiment we wanted to test the algorithms on fine BPs with a large cutoff. To this aim, we used the family $\{B_{n,m,\ell}\}$ of fine BPs from Fig.3. For n we used values between 2 to 20, for m we used values between 2 to 7, for ℓ between 0 to 3. Note that here m and n need not be prime or co-prime. Here as well we used 40 to bound the size of the cutoff. One can clearly see in Fig.6 (bottom) the exponential growth in the size of the characteristic set: looking e.g. at $x = 38$ the size of the characteristic set grow from $3K$, $25K$, $97K$, $311K$ as the number of helper states grow from 0 to 3.

X. Conclusion

We investigated the learnability of the class of fine broadcast protocols. To the best of our knowledge, this is the first work on learning concurrent models that does not assume a fixed number of processes interact.

On the positive we have shown a passive learning algorithm that can infer a BP consistent with a given sample, and have proved that this class is teachable in the sense that a system of characteristic sets (CS) can be associated with each BP in the class, so that an inference algorithm can correctly identify the BP, when given a consistent sample that subsumes a CS.

On the negative side we have shown that the consistency problem for fine BPs is NP-hard, that characteristic sets may be inevitably of exponential size, and that the class is not polynomially predictable.

References

[1] Fides Aarts, Paul Fiterau-Brostean, Harco Kuppens, and Frits W. Vaandrager. Learning register automata with fresh value generation. In *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings*, pages 165–183, 2015.

[2] Benjamin Aminof, Tomer Kotek, Sasha Rubin, Francesco Spegni, and Helmut Veith. Parameterized model checking of rendezvous systems. *Distributed Comput.*, 31(3):187–222, 2018.

[3] Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[4] Dana Angluin, Timos Antonopoulos, and Dana Fisman. Strongly unambiguous büchi automata are polynomially predictable with membership queries. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, pages 8:1–8:17, 2020.

[5] Dana Angluin, Sarah Eisenstat, and Dana Fisman. Learning regular languages via alternating automata. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3308–3314, 2015.

[6] Dana Angluin and Michael Kharitonov. When won't membership queries help? *J. Comput. Syst. Sci.*, 50(2):336–355, 1995. doi:10.1006/jcss.1995.1026.

[7] Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov. Tight cutoffs for guarded protocols with fairness. In *VMCAI*, volume 9583 of *Lecture Notes in Computer Science*, pages 476–494. Springer, 2016.

[8] Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000.

[9] Michael Blondin, Javier Esparza, and Stefan Jaax. Expressive power of broadcast consensus protocols. In *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, pages 31:1–31:16, 2019.

[10] Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A fresh approach to learning register automata. In *Developments in Language Theory - 17th International Conference, DLT 2013, Marnela-Vallée, France, June 18-21, 2013. Proceedings*, pages 118–130, 2013.

[11] Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, and Martin Leucker. Learning communicating automata from mscs. *IEEE Trans. Software Eng.*, 36(3):390–408, 2010.

[12] Colin de la Higuera. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, 2010. doi:10.1017/CBO9781139194655.

[13] Normann Decker, Peter Habermehl, Martin Leucker, and Daniel Thoma. Learning transparent data automata. In *Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings*, pages 130–149, 2014.

[14] Samuel Drews and Loris D'Antoni. Learning symbolic automata. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, pages 173–189, 2017.

[15] E. Allen Emerson and Vineet Kahlon. Reducing model checking of the many to the few. In *CADE*, volume 1831 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2000.

[16] E. Allen Emerson and Kedar S. Namjoshi. On model checking for non-deterministic infinite-state systems. In *LICS*, pages 70–80. IEEE Computer Society, 1998.

[17] E. Allen Emerson and Kedar S. Namjoshi. On reasoning about rings. *Int. J. Found. Comput. Sci.*, 14(4):527–550, 2003.

[18] Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 352–359, 1999.

[19] Javier Esparza, Martin Leucker, and Maximilian Schlund. Learning workflow petri nets. *Fundam. Informaticae*, 113(3-4):205–228, 2011.

[20] Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

[21] Dana Fisman and Sagi Saadon. Learning and characterizing fully-ordered lattice automata. In *Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings*, pages 266–282, 2022.

[22] Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin. Well-structured languages. *Acta Informatica*, 44(3-4):249–288, 2007.

[23] E. Mark Gold. Language identification in the limit. *Inf. Control.*, 10(5):447–474, 1967.

[24] E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978.

[25] Nouraldin Jaber, Swen Jacobs, Christopher Wagner, Milind Kulkarni, and Roopsha Samanta. Parameterized verification of systems with global synchronization and guards. In *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I*, pages 299–323, 2020.

[26] Nouraldin Jaber, Christopher Wagner, Swen Jacobs, Milind Kulkarni, and Roopsha Samanta. Synthesis of distributed agreement-based systems with efficiently-decidable verification. In *TACAS (2)*, volume 13994 of *Lecture Notes in Computer Science*, pages 289–308. Springer, 2023.

[27] Swen Jacobs and Roderick Bloem. Parameterized synthesis. *Log. Methods Comput. Sci.*, 10(1), 2014.

[28] Swen Jacobs and Mouhammad Sakr. Analyzing guarded protocols: Better cutoffs, more systems, more expressivity. In *VMCAI*, volume 10747 of *Lecture Notes in Computer Science*, pages 247–268. Springer, 2018.

[29] Jonas Lingg, Mateus de Oliveira, and Petra Wolf. Learning from positive and negative examples: New proof for binary alphabets. In *4th edition of Learning and Automata, Paris, France*, 2022.

[30] Nahal Mirzaie, Fathiyeh Faghih, Swen Jacobs, and Borzoo Bonakdarpour. Parameterized synthesis of self-stabilizing protocols in symmetric networks. *Acta Informatica*, 57(1-2):271–304, 2020.

[31] Anca Muscholl and Igor Walukiewicz. Active learning for sound negotiations. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, pages 21:1–21:12, 2022.

[32] Kedar S. Namjoshi. Symmetry and completeness in the analysis of parameterized systems. In *VMCAI*, volume 4349 of *Lecture Notes in Computer Science*, pages 299–313. Springer, 2007.

[33] Doron A. Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black box checking. *J. Autom. Lang. Comb.*, 7(2):225–246, 2002.
[34] Frits W. Vaandrager. Model learning. *Commun. ACM*, 60(2):86–95, 2017.
[35] Frits W. Vaandrager, Roderick Bloem, and Masoud Ebrahimi. Learning mealy machines with one timer. In *Language and Automata Theory and Applications - 15th International Conference, LATA 2021, Milan, Italy, March 1-5, 2021, Proceedings*, pages 157–170, 2021.

APPENDIX A
REDUCING DFAS TO BPS

Let Γ, Γ' be alphabets such that $\Gamma' \supseteq \Gamma$. Let w' be a word over Γ'. We use $\pi_\Gamma(w')$ for the word obtained from w' by removing letters in $\Gamma' \setminus \Gamma$. If B is a BP over $A' \supseteq A$, we refer to the words in $\{\pi_A(w) \mid w \in L(B)\}$, abbreviated $\pi_A(L(B))$, as the A-feasible words of B.

Theorem A.1. *Let L be a non-trivial regular language over Σ (i.e., neither the empty language nor Σ^*), and assume n is the number of states in the minimal DFA for L. Let $A = \Sigma \cup \{i, \$, \top, \bot, x\}$.*

1) *There exists a fine BP B with $n+5$ states over a set of actions A' subsuming A satisfying that the only A-feasible word of B with one process is i, and the set of A-feasible words with two or more processes is subsumed in $i(\Sigma)^*\$x^*(\top^* \cup \bot^*)$.*
2) *In addition, for every $w \in \Sigma^*$:*
   - *$w \in L \Leftrightarrow (iw\$\top, 2)$ is feasible in B.*
   - *$w \notin L \Leftrightarrow (iw\$\bot, 2)$ is feasible in B.*
3) *Moreover, for every B satisfying the above it holds that B has at least $n+5$ states.*

Proof. We start with the first item. Let $D = (\Sigma, Q, \iota, \delta, F)$ be a minimal DFA for L. We build the BP $B = (A, S, s, R)$ as follows, see Fig. 7. The states are $S = Q \cup \{\text{I}, \text{C}, \text{X}, \top, \bot\}$, the initial state is $s = \text{I}$. The actions $A = \Sigma \cup Q \cup \{i, \$, \bot, \top, x\}$. The transitions are as follows: $(\text{I}, i!!, \iota)$, and $(\text{I}, i??, \text{C})$. For every $\sigma \in \Sigma$: $(\text{C}, \sigma!!, \text{C})$ and $(\text{C}, \$!!, \text{X}), (\text{C}, \$??, \text{X})$. For every $(q, \sigma, q') \in \delta : (q, \sigma??, q')$. For every $q \in Q \setminus F$: $(q, \$??, \bot)$ and for every $q \in F$: $(q, \$??, \top)$. Finally, $(\text{X}, x!!, \text{X}), (\text{X}, x??, \text{X})$, $(\text{X}, \top??, \top), (\text{X}, \bot??, \bot)$, $(\top, \top!!, \top)$ and $(\bot, \bot!!, \bot)$. To adhere to the no-hidden states requirement we can add $(q, q!!, q)$ for all $q \in Q$ and every response that isn't defined is a self loop.

First we note that with one process the set of A-feasible words is i. With two processes, after executing i we have one process in ι and one in C. The process in state C can execute any word over Σ and the other process would have to simulate the DFA on this word. After $\$$ is fired a sequence of x's is feasible followed by either a sequence of $\top$'s or a sequence of $\bot$'s, depending whether the DFA has reached an accepting

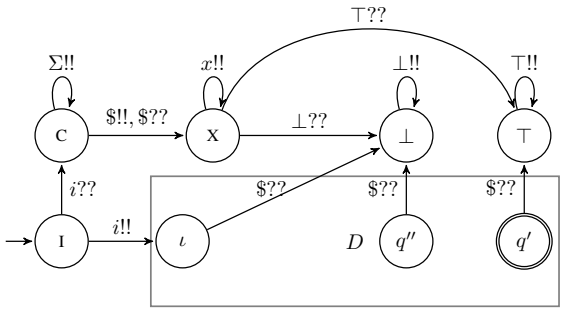

Fig. 7. Reduction from DFA-consistency to BP-consistency.

state or not. Note that with three or more processes the same set of words is feasible, therefore the cutoff is 2 and B is fine. It follows that $w \in L$ if and only if $(iw\$\top, 2)$ is feasible in B and similarly $w \notin L$ if and only if $(iw\$\bot, 2)$ is feasible in B. This proves the first two items.

For the third item, we first claim that the requirement of the A-feasible words require at least 5 states. Indeed, since with one process only i is A-feasible, the only action feasible from the initial state, call it I is i. Since ii is infeasible, $i!!$ from the initial state must reach a new state, and since no other A-feasible words are feasible with one process, this state has no A-feasible actions. With two processes, after i we can also fire all actions in $\Sigma \cup \{\$\}$, and so $i??$ from the initial state must reach a state that enables all of them, call it C. Afterwards, once a \$ has been fired neither i nor Σ or \$ are feasible, but only x and either $\top$ or $\bot$. So $\$!!$ as well as $\$??$, must have reached new states that enables these. Due to the fact that after x nothing but $\top$ or $\bot$ is feasible, we get that x is fired from a different state. Finally, since after $\top$ only $\top$'s are feasible, and similarly after $\bot$, these actions have to be enabled from different states as well, call them $\top$ and $\bot$. So there are at least 5 states, and they possess transitions as shown in Fig. 7.

Now, assume towards contradiction that the minimal DFA for L has n states, and there is a BP B satisfying these requirements with less than $n+5$ states. We claim that if B has less than $n+5$ states then we can build a DFA for L with less than n states. First note that as argued earlier, after firing i there exists a process in a state, call it ι, from which none of the actions in A can be fired. Moreover, if there is more than one process, then the rest of processes after executing i are in a different state, the one we called C. Assuming first, the responses on actions in Σ do not reach one of the identified 5 states, it is clear that they must simulate the transitions of the DFA, as otherwise we would get a contradiction to the minimality of the DFA. Removing this assumption, we note that out of the five states, responses on Σ actions can only get to C as otherwise the set of A-feasible words will not be a prefix of $i(\Sigma)^*\$x^*(\top^* \cup \bot^*)$ as required. Assuming that this happens for some $\sigma \in \Sigma$, then at that point all the processes will be in C, which means that after taking \$ neither $\bot$ nor $\top$ is feasible, contradicting requirement (2). This concludes that the responses on actions in Σ do behave the same as in the minimal DFA. Regarding the actions in $A' \setminus A$, we can assume that sending and receiving transitions on every $a \in A' \setminus A$ behave such that, when starting from a global state where only states C and $f^{\mathsf{st}}(a)$ are lit, then also after a the same states are lit.[4] Otherwise we would get a contradiction either to requirement (2) or to minimality of the DFA:

- First, note that if C is not lit after a, then requirement (2) cannot be satisfied.
- If a transition on a from $f^{\mathsf{st}}(a)$ or C goes to an auxiliary state other than C, then we also get a contradiction to requirement (2).
- If a transition on a from $f^{\mathsf{st}}(a)$ goes to a non-auxiliary state other than $f^{\mathsf{st}}(a)$, then we get a contradiction to minimality of the DFA, since then from this state exactly the same words in Σ^* would be accepted as from $f^{\mathsf{st}}(a)$.

This concludes the proof. □

[4]This means that either the transitions on a are all self-loops, or the response from C goes to $f^{\mathsf{st}}(a)$, and at least one of the sending and receiving transitions on a from $f^{\mathsf{st}}(a)$ goes to C (and is a self-loop otherwise).

Theorem A.2. *BP consistency is NP-hard.*

Proof. The proof is by reduction from DFA-consistency which was shown to be NP-hard in [24], using the idea in Thm. A.1. Given an input to DFA-consistency, namely a sample $\mathcal{S}$ and a number k, we produce a sample $\mathcal{S}'$ and $k' = k + 5$ as an input to BP-consistency as follows.

We start by putting $(i, 1, \text{T})$, $(ii, 2, \text{F})$, $(ix, 2, \text{F})$, $(i\top, 2, \text{F})$, $(i\bot, 2, \text{F})$ and for all $\sigma \in \Sigma$ $(i\sigma i, 2, \text{F}), (i\sigma x, 2, \text{F}), (i\sigma\$xi, 2, \text{F})$ to $\mathcal{S}'$. Then, for each $(w, \text{T}) \in \mathcal{S}$ we add $(iw\$\top\top, 2)$ and $(iw\$xx\top\top, 2)$ with positive label to $\mathcal{S}'$, and the following words with negative label to $\mathcal{S}'$: $(iw\$\top\bot, 2)$, $(iw\$\bot\top, 2)$, $(iw\top, 2)$, $(iw\$\bot, 2)$, $(iw\$\top i, 2)$, $(iw\$\top x, 2)$, and $(iw\$\top\sigma, 2)$, $(iw\$\sigma, 2)$ for any $\sigma \in \Sigma$.

For each $(w, \text{F}) \in \mathcal{S}$ we add $(iw\$\bot\bot, 2)$ and $(iw\$xx\bot\bot, 2)$ with positive label to $\mathcal{S}'$, and the following words with negative label to $\mathcal{S}'$: $(iw\$\top\bot, 2)$, $(iw\$\bot\top, 2)$, $(iw\bot, 2)$, $(iw\$\top, 2)$, $(iw\$\bot x, 2)$, and $(iw\$\bot\sigma, 2)$, $(iw\$\sigma, 2)$ for any $\sigma \in \Sigma$.

From similar arguments as the proof for the third item in Thm. A.1 we can show that the reduction is valid, namely there is a DFA for the given sample $\mathcal{S}$ with less than k sates if and only if there is a BP for the constructed sample $\mathcal{S}'$ with less than $k + 5$ states. □