Learning Broadcast Protocols

Dana Fisman[*1], **Noa Izsak**[*1†], **Swen Jacobs**[*2]
[1]Ben-Gurion University
[2]CISPA Helmholtz Center for Information Security
dana@cs.bgu.ac.il, izsak@post.bgu.ac.il, jacobs@cispa.de

Abstract

The problem of learning a computational model from examples has been receiving growing attention. For the particularly challenging problem of learning models of distributed systems, existing results are restricted to models with a *fixed* number of interacting processes. 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, i.e., a number of processes that is sufficient to produce all observable behaviors. 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 can infer a correct BP from a sample that is consistent with a fine BP, and a minimal equivalent BP if the sample is sufficiently complete. On the negative side we show that (a) characteristic sets of exponential size are unavoidable, (b) the consistency problem for fine BPs is NP hard, and (c) that fine BPs are not polynomially predictable.

1 Introduction

Learning computational models has a long history starting with the seminal works of Gold (1967; 1978) and Angluin (1987). Questions regarding learning computational models have raised a lot of interest both in the artificial intelligence community and the verification community. (Peled et al. (2002), Vaandrager (2017)). Many results regarding the learnability of various computational models used in verification have already been obtained (Beimel et al. 2000; Bollig et al. 2013; Decker et al. 2014), Angluin et al. (2015), Nitay et al. (2023), Roy et al. (2023).

Particularly challenging is learning of concurrent computational models. Compared to most sequential models, they offer another level of succinctness, and they usually have no unique minimal model. Both of these aspects can make learning significantly more difficult. Various results regarding learning concurrent models have already been obtained (Bollig et al. 2010), Esparza et al. (2011), Muscholl et al. (2022). However, these results are limited to models with a fixed number of processes, and therefore cannot reliably learn models for (distributed) protocols that are expected to run correctly for *any* 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 (Emerson and Namjoshi 1998). The basic model assumes that communication and processes are reliable, i.e., it does not consider communication failures or faulty processes. BPs have mainly been studied in the context of parameterized verification, i.e., proving functional correctness according to a formal specification, for all systems where an arbitrary number of processes execute a given protocol.

The challenge in reasoning about parameterized systems such as BPs is that a parameterized system concisely represents an infinite family of systems: for each natural number n it includes the system where n indistinguishable processes interact. The system is correct only if it satisfies the specification for any number n of processes interacting. In the context of verification, a variety of approaches has been investigated to overcome this challenge.

Some of these approaches are based on the notion of *cutoff*, i.e., 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, where the *cutoff system* is a system with exactly c processes interacting. In the literature, many results exist that provide cutoffs for certain classes of properties in a given computational model (Emerson and Namjoshi 2003; Emerson and Kahlon 2000), Außerlechner et al. (2016). Moreover, cutoffs also enable the *synthesis* of implementations for parameterized systems from formal specifications (Jacobs and Bloem 2014), a problem closely related to learning.

In this paper, we develop a learning approach for BPs. 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 (1) that the BP under consideration has no hidden states, i.e., every state has at least one broadcast sending action by which it can be recognized; and (2) that there *exists* a cutoff, i.e., a number c such that the language (of finite words over actions) derived by c processes is the same as the language derived by any number greater than c. We call such BPs *fine*, and note that many BPs studied in the literature are fine.

[*]These authors contributed equally.

[†]This author was supported by ISF grant 2507/21 & Frankel Center for Computer Science, BGU.

Moreover, the restriction to fine BPs does not overly simplify the problem, as even under this assumption we obtain negative results for some basic learning problems. We note that not all BPs have a cutoff (whether or not they have hidden states), and that when a cutoff exists the derived language is regular. The fact that the derived language is regular also holds in previous works on learning concurrent models (communicating automata (Bollig et al. 2010), workflow Petri nets (Esparza et al. (2011)), and negotiation protocols (Muscholl et al. (2022))) merely since a finite number of essentially finite state machines is in consideration.[1] 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 DFA for the language. Moreover, the problem we consider goes way beyond what has been considered in previous works in the sense that our approach works if *there exists* a cutoff, but it does not require that the cutoff is known a priori, which is equivalent to the assumption of a (given) fixed number of processes in existing approaches.

We focus mainly on passive learning paradigms (de la Higuera 2010). Specifically, we consider the following problems: 1. *Inference* — given a sample consistent with a BP, return a BP that is consistent with the sample, 2. *Consistency* — whether there exists a BP with at most k states that agrees with a given sample, 3. *Polynomial data* — whether characteristic sets are of polynomial size, and 4. *Polynomial Predictability* — whether a learner can correctly classify an unknown word with high probability after asking polynomially many membership (MQ) and draw queries(DR).

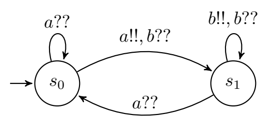
We prove a few properties of fine BPs relevant to learning in §3. In §4, we provide an inference algorithm that, given a sample of words that are consistent with a fine BP, infers a correct BP. In §5, we show that the inference algorithm produces a minimal equivalent BP when the sample subsumes a characteristic set, and that characteristic sets of exponential size are unavoidable. In §6, we show that consistency is NP-hard for the class of fine BPs. In §7 we show that fine BPs are not polynomially predictable. For complete proofs see the extended version (Fisman, Izsak, and Jacobs 2023).

2 Preliminaries

Broadcast Protocols

In the following we define broadcast protocols as introduced by Emerson and Namjoshi (1998) and studied in the seminal paper by Esparza et al. (1999). Broadcast protocols are one of the most powerful computational models for which some parameterized verification problems are still decidable, and are strictly more powerful than other standard communication primitives such as pairwise rendezvous or disjunctive guards (Emerson and Kahlon 2003).

Broadcast Protocols (BPs) A *broadcast protocol* $B = (S, s_0, L, R)$ consists of a finite 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 broadcast *sending transition*, and a transition labeled with $a??$ is a broadcast *receiving transition*, also called a *response*.[2] For each action $a \in A$, there must be exactly one outgoing response from every state.



$$\mathbf{M}_a = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \quad \mathbf{v}_a = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{v}'_a = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathbf{M}_b = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \quad \mathbf{v}_b = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{v}'_b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Figure 1: A simple BP (left) and its algebraic representation.

Given a BP $B = (S, s_0, L, R)$ we consider systems B^n, composed of n identical processes that execute B. Let $[n]$ denote the set $\{0, 1, \ldots, n\}$. A *configuration* of B^n is a function $\mathbf{q} : S \to [n]$, assigning to each state a number of processes. The *initial configuration* $\mathbf{q}_0$ is the configuration with $\mathbf{q}_0(s_0) = n$ and $\mathbf{q}_0(s) = 0$ for all $s \neq s_0$. In a *global transition*, all processes make a move: One process takes a sending transition (labeled $a!!$), modeling that it broadcasts the value a to all the others processes in the system. Simultaneously, all of the other processes take the receiving transition (labeled $a??$) from their current state.[3]

Example 2.1. Fig.1 (left) depicts a simple BP B. In the initial configuration of the system B^9 we have 9 processes in s_0. If a process broadcasts a, it moves to s_1 via the transition label $a!!$. The other processes respond following the $a??$ transition from their current state, hence they remain in s_0. Now we are in a configuration $\mathbf{q}'$ with one process in s_1 and 8 in s_0. If from $\mathbf{q}'$ another process broadcasts a, it moves from s_0 to s_1. The processes in s_0 stay there (following $a??$ from s_0), and the process in s_1 moves back to s_0 (following $a??$ from s_1). Thus, we return to $\mathbf{q}'$. If from $\mathbf{q}'$ the process in s_1 broadcasts b then it stays in s_1 (following $b!!$), while all other processes move to s_1 via $b??$ i.e., all will be in s_1.

Following Esparza et al. (1999), we make the standard assumption 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. Note that the additional assumption of no hidden states is modest, since many examples from the literature satisfy it (e.g., the MESI protocol in Esparza et al. (1999) or the last-in first-served protocol in Delzanno et al. (1999)), and every protocol that does not satisfy this restriction can easily be modified to satisfy it without changing its functionality.

Semantics of B^n We can represent transitions of a system B^n algebraically. Assuming some ordering $s_0, s_1, \ldots s_{|S|-1}$ on the set of states S, we can identify configurations of B^n with vectors from $[n]^{|S|}$, also called *state-vectors*. We use

[1]The language of a BP in general need not be regular (Finkel and Schnoebelen 2001; Geeraerts, Raskin, and Begin 2007) and this is true also with the restriction to no hidden states.

[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.

[3]We give a formal semantics of B^n below.

$\mathbf{q}[i]$ 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 configuration where all n processes 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 state s_i has an outgoing sending transition on action a we say that a is *enabled* in s_i; if i is lit in $\mathbf{q}$ we also say that a is *enabled* in $\mathbf{q}$.

With each action a we can associate (1) two unit-vectors $\mathbf{v}_a = \mathbf{u}_i$ for the origin and $\mathbf{v}'_a = \mathbf{u}_j$ for the destination, following its sending transition $(s_i, a!!, s_j)$, and (2) a *broadcast matrix* $\mathbf{M}_a$, which is an $|S| \times |S|$ matrix with $\mathbf{M}_a(m,k) = 1$ if there is a response $(s_k, a??, s_m) \in R$, and $\mathbf{M}_a(m,k) = 0$ otherwise. Every column of such a matrix is a unit vector.

Then the transitions T of B^n are defined as follows: there is a transition between configurations $\mathbf{q}$ and $\mathbf{q'}$ on action a in B^n, denoted $(\mathbf{q}, a, \mathbf{q}') \in T$, iff there exists $(s_i, a!!, s_j) \in R$ with $\mathbf{q}[i] \geq 1$ and: $\mathbf{q}' = \mathbf{M}_a \cdot (\mathbf{q} - \mathbf{v}_a) + \mathbf{v}'_a$.

To see this, note that the state-vector $\mathbf{q} - \mathbf{v}_a$ corresponds to the sending process leaving the state s_i. The state-vector $\mathbf{M}_a \cdot (\mathbf{q} - \mathbf{v}_a)$ describes the situation after the other processes take the responses on a. Finally, $\mathbf{q}'$ is the resulting state-vector after the sending process arrives at its target location.

Example 2.2. Consider again the BP in Fig.1, depicted with the broadcast matrices for the two actions $\mathbf{M}_a$ and $\mathbf{M}_b$, and the associated origin and destination vectors $\mathbf{v}_a$, $\mathbf{v}'_a$, $\mathbf{v}_b$, $\mathbf{v}'_b$. In configuration $\mathbf{q} = \left[\begin{smallmatrix}2\\2\end{smallmatrix}\right]$, a is enabled (s_0 is lit). Computing the effect of a, we first get $\mathbf{q} - \mathbf{v}_a = \left[\begin{smallmatrix}1\\2\end{smallmatrix}\right]$, then $\mathbf{M}_a \cdot \left[\begin{smallmatrix}1\\2\end{smallmatrix}\right] = \left[\begin{smallmatrix}3\\0\end{smallmatrix}\right]$, and finally, $\mathbf{q}' = \left[\begin{smallmatrix}3\\0\end{smallmatrix}\right] + \mathbf{v}'_a = \left[\begin{smallmatrix}3\\1\end{smallmatrix}\right]$.

An *execution* of B^n is a finite sequence $e = \mathbf{q}_0, a_1, \mathbf{q}_1, a_2, \ldots, a_m, \mathbf{q}_m$ such that $(\mathbf{q}_i, a_{i+1}, \mathbf{q}_{i+1}) \in T$ for every $i \in [m-1]$. We say that e 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 words that are feasible in B^n, and the language of B, denoted $L(B)$, is the union of $L(B^n)$ over all $n \in \mathbb{N}$.

Let B_1 and B_2 be two BPs. We say that B_1 and B_2 are *equivalent* iff $L(B_1){=}L(B_2)$. A BP B is said to have a *cutoff* $k \in \mathbb{N}$ if for any $k' > k$ it holds that $L(B^k) = L(B^{k'})$. A BP with no hidden states is termed *fine* if it has a cutoff. We measure the size of a BP by its number of states. Thus, a BP is termed *minimal* if there is no equivalent BP with fewer states. Note that unlike the case of DFAs, there is no unique minimal fine BP, as shown by the following example.

Example 2.3. Fig.2 shows two BPs B_1 and B_2. Note that $L(B_1^1) = a^*$, since with a single process, a is the only possible action from $\mathbf{q}_0$, and we arrive in $\mathbf{q}_0$ after executing it. With 2 processes, after executing a we arrive in state-vector $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, and we can execute either a or b, and each of them brings us to $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$ again. Therefore, $L(B_1^2) = a(a \cup b)^*$. Moreover, adding more processes does not change the language, i.e., $L(B_1^n) = L(B_1^2)$ for all n. Hence $L(B_1) = L(B_1^2)$ and the cutoff of B_1 is 2. Similarly, we can show that $L(B_2) = a(a \cup b)^*$ and the cutoff of B_2 is 2. Note that B_1 and B_2 are not isomorphic, but they are equivalent.

We note that the aforementioned examples from the literature (Esparza et al. (1999) and Delzanno et al. (1999)) also have a cutoff, and thus are fine BPs.

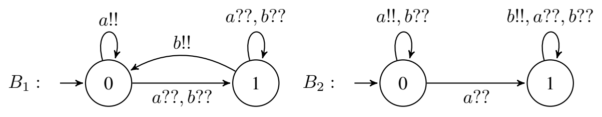


Figure 2: Two non-isomorphic equivalent fine BPs

Learning Problems

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) (resp. (w, n, F)) is consistent with B if w is feasible (resp. infeasible) in B^n. A sample is *consistent* with B if all triples in it are consistent with B. The size of $\mathcal{S}$ is defined as the sum of length of words in it.

We consider the following problems related to learning a class $\mathcal{C}$ of computational models, phrased for BPs.

Problem 2.1 (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.

Obviously one would prefer the returned BP to be minimal or sufficiently small. Phrased as a decision problem this is the consistency problem.

Problem 2.2 (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.*

The consistency problem is NP-hard even for DFAs (Gold 1978). Thus, inference algorithms are often based on SAT or SMT solvers In §4 we provide such an inference algorithm for fine BPs. Note that in many cases it is possible to devise a trivial inference algorithm (e.g., for DFAs the prefix-tree automaton) that is correct on the sample but makes no generalization and does not attempt to minimize the returned result. We show in §5 that if the sample is sufficiently complete (subsumes a *characteristic set*), the inference algorithm we provide in fact returns the minimal BP that agrees with the sample. One may thus ask how large should a characteristic set be. We show that, unfortunately, it can be of size exponential in the number of states of the BP.

Problem 2.3 (Polynomial data)**.** *Does there exist an inference algorithm $\mathfrak{A}$ such that one can associate with every BP $B \in \mathcal{C}$ a sample $\mathcal{S}_B$ of size polynomial in B so that $\mathfrak{A}$ correctly infers $L(B)$ from $\mathcal{S}_B$ or any sample subsuming it.*

The last problem we consider is in the active learning paradigm. Its definition is quite long and deferred to §7.

Problem 2.4 (Polynomial Predictability)**.** *Can a learner correctly classify an unknown word with high probability after asking polynomially many membership and draw queries.*

3 Properties of Broadcast Protocols

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

It is not hard to see that the set of feasible words $L(B)$ of a given BP is prefix-closed. That is, if uv is feasible for

$u, v \in A^*$ then u is feasible as well. Additionally, if $w \in A^*$ is feasible in B^k then w is feasible in B^ℓ for all $\ell > k$.

Lemma 3.1 (Prefix-closedness and monotonicity)**.** *If B is a BP then $L(B)$ is prefix-closed. Moreover, $L(B^k) \subseteq L(B^\ell)$ for all $\ell > k$.*

The following lemma asserts that if wa is feasible with n processes but not with $m < n$, while w is feasible with m processes (for $w \in A^*$ and $a \in A$) then wa must be feasible with $m + 1$ processes. Intuitively this is since m processes are enough to execute all actions in w, and therefore any additional processes will take only receiving transitions along w, and will all arrive in the same local state. Thus, if a is not enabled after w with $m + 1$ processes, then the additional process did not lit the state s_a enabling a, and the same is true if we add any bigger number of processes.

Lemma 3.2 (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 $wa \in L(B^{m+1})$.*

Recall that fine BPs have no canonical minimal representation, in the sense that, as shown in Ex.2.3, there could be two non-isomorphic BPs for the same language. The lack of a canonical minimal representation often makes it difficult to achieve a learning algorithm. The following important lemma asserts, that while two minimal fine BPs may be non-isomorphic there is a tight correlation between them.

Since every action is enabled by a unique state, and every state enables at least one action, in a minimal fine BP the set A is partitioned between states, and if there is a state s_1 in B_1 whose set of enabled actions 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 enabled actions is exactly A'. So we can define such a mapping between the states of two minimal fine 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 enabled actions) is lit in $\mathbf{q}_w$. Moreover, the fact that $L(B_1) = L(B_2)$ guarantees that $L(B_1^m) = L(B_2^m)$ for any $m \in \mathbb{N}$. This bundle of claims can be proven together by induction first on the number of processes m, and second the length of the word w. In the following we use $f^{\mathsf{act}}(s) = A'$ if A' is the set of actions enabled in s.

Lemma 3.3 (Relation between minimal fine equivalent BPs)**.** *Let B_1 and B_2 be minimal fine BPs with 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 $w \in A^*$ if $B_1^m(w) = \boldsymbol{p}_w$ and $B_2^m(w) = \boldsymbol{q}_w$ then $\boldsymbol{p}_w[i]$ is lit if and only if $\boldsymbol{q}_w[h(i)]$ is lit, for every state i.*

4 Inferring a BP from a Sample

Let $\mathcal{S}$ be a sample. The inference algorithm $\mathfrak{I}$ we devise constructs a BP $B_{\mathcal{S}}$ that agrees with $\mathcal{S}$.

Let $A_{\mathcal{S}}$ be the set of actions that appear in $\mathcal{S}$ in at least one feasible word. In order to return a BP $B_{\mathcal{S}}$ with no hidden states, we allow $B_{\mathcal{S}}$ to have a set of actions $A \supseteq A_{\mathcal{S}}$.[4] We use S for the set of states of $B_{\mathcal{S}}$, and s_0 for its initial state.

We construct a set of constraints that define the BP $B_{\mathcal{S}}$. More precisely, we construct a set of constraints $\Psi_{\mathcal{S}}$ 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 $\Psi_{\mathcal{S}}$ implement a BP consistent with the sample. Formally, we say that functions $f^{\mathsf{st}}, f^{!!}, \{f^{??}_a \mid a \in A\}$ *implement* a BP $B = (S, s_0, L, R)$ if for every $(s_i, a!!, s_j) \in R$ we have $f^{\mathsf{st}}(a) = s_i$, $f^{!!}(a) = s_j$ and for every $(s_i, a??, s_j) \in R$ we have $f^{??}_a(s_i) = s_j$. 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 $\{w : (w, i, \mathrm{T}) \in \mathcal{S}\}$, and let $\mathcal{N}_i$ be $\{w : (w, i, \mathrm{F}) \in \mathcal{S}\}$. Note that by Lem.3.1 it follows that if $w \in \mathcal{P}_i$ then w is feasible in B^j for every $j \geq i$. Similarly, if $w \in \mathcal{N}_i$, then w is infeasible in B^j for every $j \leq i$. We define $\mathcal{N}$ (resp. $\mathcal{P}$) as the union of all $\mathcal{N}_i$'s (resp. $\mathcal{P}_i$'s).

We define a relation between actions $a, b \in A_{\mathcal{S}}$ as follows. 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 Lem.3.3, $a \#_{\mathcal{S}} b$ means that the sample $\mathcal{S}$ has information contradicting that a and b are enabled in 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 identify states by the set of actions they enable and we assume there are no hidden states, we define the set of states $S = \{f^{\mathsf{st}}(a) : a \in A\}$ as the set of terms $f^{\mathsf{st}}(a)$. This definition guarantees that no states are hidden. Moreover, we designate one of the states, that we term s_0, as the initial state: $\exists s \in S : s = s_0$.
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 $au \in \mathcal{P}_i$ for some i and some $u \in A^*$ 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 inductively for every word $w \in \mathcal{P}$ for the minimal $i \geq 1$ such that $w \in \mathcal{P}_i$ and for every $w \in \mathcal{N}$ for the maximal $i \geq 1$ such that $w \in \mathcal{N}_i$.

   In the base case $i{=}1$ we have $w \in \mathcal{P}_1 \cup \mathcal{N}_1$. 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 process reaches after the system reads $a_1 \ldots a_k$, and p_0 should be s_0.

   If $w \in \mathcal{P}_1$, we add the constraint $\psi_{w,1}$ defined as
   $$(p_0 = s_0) \wedge \bigwedge_{1 \leq \ell \leq m} \left(p_{\ell-1} = f^{\mathsf{st}}(a_\ell) \wedge p_\ell = f^{!!}(a_\ell)\right)$$
   This requires that the next letter a_ℓ is enabled in the state the process reached after $a_1 a_2 \ldots a_{\ell-1}$ was executed.

   If $w \in \mathcal{N}_1$ we add the following constraint
   $$\bigvee_{0 \leq \ell < m} \left(\psi_{w[..\ell],1} \wedge p_\ell \neq f^{\mathsf{st}}(a_{\ell+1})\right)$$
   where $w[..\ell]$ denotes the ℓ'th prefix of w, namely $a_1 a_2 \ldots a_\ell$, and we let $\psi_{\epsilon,1} = \mathrm{T}$. 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.

[4]Note that in §5 we show that it is enough to consider $A = A_{\mathcal{S}}$ if $\mathcal{S}$ is sufficiently complete.

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 $1 \le j \le i$. The state of the processes after reading the next letter, a_{l+1}, depends on their state after reading a_l.

   Let $w \in \mathcal{P}_i$, we add the constraint $\psi_{w,i}$ defined as follows.

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

   where

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

   Intuitively, $\psi_{w,i}$ requires that for every letter a_ℓ of w one of the processes, call it j, reached a state in which a_ℓ is enabled. The formula $\varphi_{j,\ell}$ states that the j-th process took the sending transition on a_ℓ and the rest of the processes took the respective receiving transition.

   Let $w \in \mathcal{N}_i$. We then add the following requirement

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

   where 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 4.1. *Let $\mathcal{S}$ be a sample that is consistent with some fine BP. Let $B_{\mathcal{S}}$ be a BP that satisfies the prescribed constraints $\Psi_{\mathcal{S}}$. Then $B_{\mathcal{S}}$ is a BP consistent with $\mathcal{S}$.*

Proof. We prove that if $w \in \mathcal{P}_i$ (resp. $w \in \mathcal{N}_i$) then w is feasible (resp. infeasible) in $B^i_{\mathcal{S}}$, 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 of the form w that are in $\mathcal{P}_i \cup \mathcal{N}_i$. If $w \in \mathcal{P}_i$ is already in $\mathcal{P}_j$ for $j < i$ then by the induction hypothesis it is already feasible for j processes in the constructed BP, and by Lem.3.1, it is also feasible with i processes. Otherwise, $w \in \mathcal{P}_i \setminus \bigcup_{j<i} \mathcal{P}_j$. In this case, constraint (5) makes sure that every prefix of w is feasible with i processes, 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 $1 \le \ell \le m$ such that while $w[..\ell{-}1]$ is feasible, a_ℓ is not enabled in 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, which is added for i or some $j > i$. In the latter case, Lem.3.1 implies that w is infeasible in $B^i_{\mathcal{S}}$. ☐

Finally, note that our constraints are in the theory of equality with uninterpreted functions (EUF), and are therefore decidable.[5] Thus, an algorithm for inferring fine BPs can be implemented using an SMT solver.

Corollary 4.2. *$\mathfrak{I}$ is an inference algorithm for fine BPs.*

5 Returning a Minimal BP

In this section we show that when the sample is sufficiently complete we can guarantee that we return a minimal equivalent BP, and not just a BP that agrees with the sample. We thus first show that every fine BP B 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. We refer to such a sample as a *characteristic set* (CS).

In the following, we first describe a procedure $\mathfrak{G}$ that generates a sample $\mathcal{S}_B$ from a fine BP B, and then we prove that an inference algorithm $\mathfrak{A}$ can correctly infer a minimal BP B' equivalent to B from any sample subsuming $\mathcal{S}_B$.

Generating a Characteristic Set

The CS 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 actions. 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 $B^i(w) = \mathbf{p}_{w,i}$, i.e. 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 $\mathcal{T}_{i+1} = \mathcal{T}_i$.[6] Note that given that 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 $n \in [i+1]$, let $\mathcal{P}_n = \{(u, n, \mathrm{T}) \mid n \text{ is the minimal for which } u \text{ is positive in } \mathcal{T}_n\}$, $\mathcal{N}_n = \{(u, n, \mathrm{F}) \mid n \text{ is the maximal for which } u \text{ is negative in } \mathcal{T}_n\}$. Then the sample is $\mathcal{S}_B = \bigcup_{n=1}^{i+1} (\mathcal{P}_n \cup \mathcal{N}_n)$.

[5]While constraint (2) depends on the unknown set A, we can bound the size of A, e.g., by the size of the prefix tree of $\mathcal{S}$.

[6]We say $\mathcal{T}_{i+1} = \mathcal{T}_i$ if they agree on the tree structure and the edge labels (regardless of the nodes' annotations).

Proving That $\mathfrak{G}$ Generates Characteristic Sets

We first note that for any reachable 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 s in the state-vector annotating the node is lit). The following lemma strengthens this statement further.

Lemma 5.1. *Let* $\boldsymbol{p}$ *be a state-vector that is reachable in* B^n*. Then for every shortest word* w *that reaches* $\boldsymbol{p}$ *in* B^n *there exists a node* w *in* $\mathcal{T}_n$ *such that* $\boldsymbol{p}_w = \boldsymbol{p}$*.*

When the sample $\mathcal{S}$ subsumes the set $\mathcal{S}_B$ then $\#_{\mathcal{S}}$ induces an equivalence relation between the actions:

Lemma 5.2. *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* $\mathcal{S}_B$ *then* $\sim_{\mathcal{S}}$ *is an equivalence relation.*

Theorem 5.3. *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* $\mathfrak{A}$ *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 §4 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. 4.1 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.5.2 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.) Hence $\mathfrak{I}'$ will not return that there is no satisfying assignment.

Next we note that by Lem.5.1 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 $w = a_1 a_2 \ldots a_n$ 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 $\mathbf{p}_2$.

Hence, 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.3.3. Thus, $L(B) = L(B')$. □

Regarding the problem of *polynomial data* we show that there exist fine BPs for which there is no CS of polynomial size. The proof constructs a family of BPs of size quadratic in n for which there exists an action $a_\top$ such that the length of the shortest word containing $a_\top$ is exponential in n. Thus, any CS has to include at least one such long word.

Theorem 5.4. *There exists a family of fine BPs with no characteristic set of polynomial size.*

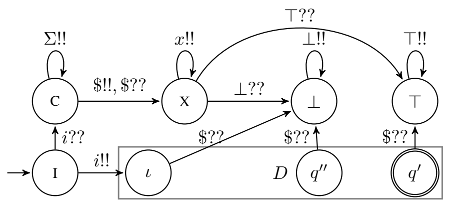


Figure 3: Reduction of DFA-consistency to BP-consistency.

The same family used in the proof of Thm.5.4 also shows that fine BPs can be exponentially smaller than the minimal DFA accepting the same language. This is since in a DFA for every state q the length of the shortest word reaching q is bounded by the number of states in the DFA.

Corollary 5.5. *There exists a family of fine BPs for which the corresponding minimal DFA is of exponential size.*

6 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 (Gold 1978), 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. However, a fine BP can simulate a DFA, in a manner prescribed in Lem.6.1.

Fig.3 provides a schematic illustration of the simulation. The states in the rectangle are the original states of the DFA, and ι is the initial state of the DFA. All responses that are not shown in the figure are self-loops, we omit them to avoid clutter. In addition, to make the BP fine, we need to allow each state q to enable some action, call it q. The simulation would like to ignore these actions, i.e. consider the projection of the words to words without these actions. Formally, 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', such that $A' \supseteq A$, 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. Lem.6.1 states in which manner the BP simulates the DFA.

Lemma 6.1. *Let* L *be a non-trivial regular language over* Σ*, and assume* n *is the number of states in the minimal DFA for* L*.*[7] *Let* $A = \Sigma \cup \{i, \$, \top, \bot, x\}$*.*

1. *There exists a fine BP* B *over actions* A' *such that* $A' \supseteq A$ *satisfying that* $\pi_A(L(B^1)) = \{i\}$ *and* $\pi_A(L(B^n)) \subseteq i(\Sigma)^* \$ x^* (\top^* \cup \bot^*)$ *for every* $n \geq 2$*.*
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 sketch. In order to keep the number of processes in the DFA part exactly one, the i action from the initial state of the BP (state I) sends one process to ι and the rest to state C. To allow every letter $\sigma \in \Sigma$ to be taken from every state

[7]A language is non-trivial if it is not the empty set or Σ^*.

of the DFA, the state C of the BP enables all letters in Σ, and the original (q, σ, q') transitions of the DFA are transformed into responses $(q, \sigma??, q')$. Next, to deal with the fact that the language of a DFA is defined by the set of words that reach an accepting state whereas the language of a BP is the set feasible words, we introduce the letters \$, $\top$ and $\bot$. Upon \$ the single process in one of the DFA states moves into either state $\bot$ or state $\top$ depending on whether it was in an accepting state or a rejecting state; and all processes in states C move to state X where they wait to follow the respective $\bot$ or $\top$. Now only $\{x, \top\}$ or $\{x, \bot\}$ are enabled. An x!! transition would not change this situation whereas a $\top$ (resp. $\bot$) transition will ensure only $\top$ (resp. $\bot$) can be taken henceforth. The complete proof (in the extended version) shows the three requirements of the lemma are satisfied. □

Theorem 6.2. *BP consistency is NP-hard.*

Proof Sketch. The proof is by reduction from DFA consistency which is NP-hard by (Gold 1978). Given an input to DFA-consistency, namely a sample $\mathcal{S}$ and $k \in \mathbb{N}$, we produce a sample $\mathcal{S}'$ and $k'=k+5$ as an input to BP-consistency so the relation between the minimal BP consistent with $\mathcal{S}'$ and the minimal DFA consistent with $\mathcal{S}$ is as stated in Lem.6.1.

The sample $\mathcal{S}'$ consists of various triples ensuring a BP consistent with $\mathcal{S}'$ has the structure given in Fig.3. For instance, $(i, 1, \text{T})$ and $(ii, 2, \text{F})$ enforce that i is enabled in the initial state but is not a self-loop. A pair (w, T) (resp. (w, F)) in $\mathcal{S}$ is altered to triple $(iw\$\top\top, 2, \text{T})$ (resp. $(iw\$\bot\bot, 2, \text{T})$ in $\mathcal{S}'$, making sure that words accepted (resp. rejected) by the DFA create respective feasible words ending with $\top$'s (resp. $\bot$'s). Each such pair carries some additional triples added to $\mathcal{S}'$ to continue enforcing the structure of Fig.3. □

See the extended version for an alternative proof via a direct reduction from all-eq-sat, inspired by Lingg et al (2024).

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.

7 BPs Are Not Polynomially Predictable

Here 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 *membership queries* (MQ) with regard to the target language $C \in \mathcal{C}$ or *draw queries* (DR) that can be implemented using MQ. 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 $\text{MQ}(w)$. 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 ε between 0 and 1, there exists a learner that will classify the word to predict correctly with probability at least $(1 - \varepsilon)$, 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 (thus BPs in general) are not polynomially predictable.

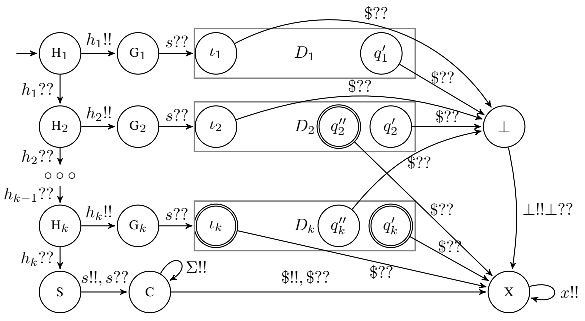


Figure 4: A BP simulating intersection of k DFAs.

Theorem 7.1. *Assuming the intractability of any of the following three problems: testing quadratic residues modulo a composite, inverting RSA encryption, or factoring Blum integers, fine BPs are not polynomially predictable with* MQ.

Proof Sketch. 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 (Angluin and Kharitonov 1995). 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. Given a set $D_1, D_2, \ldots, D_k$ of DFAs, we can construct a BP B as shown in Fig.4 such that B simulates the run of the k DFAs together. As in the proof of Lem.6.1 we can send one process to simulate any of the DFAs. Here we need k processes to send a process to the initial state of each of the DFAs, and an additional process to enable all letters in Σ. Thus, the cutoff is $k + 1$. The BP detects whether a given word w is accepted by all the DFAs by checking whether $uw\$\bot$ is infeasible in B where u is some initialization sequence that is required to send the processes to the initial states of the DFAs. □

8 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 showed a passive learning algorithm that can infer a BP consistent with a given sample, and even return a minimal equivalent BP if the sample is sufficiently complete. On the negative, we showed that the consistency problem for fine BPs is NP-hard; characteristic sets may be inevitably of exponential size; and the class is not polynomially predictable.

References

Angluin, D. 1987. Learning Regular Sets from Queries and Counterexamples. *Inf. Comput.*, 75(2): 87–106.

Angluin, D.; Eisenstat, S.; and Fisman, D. 2015. 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*, 3308–3314.

Angluin, D.; and Kharitonov, M. 1995. When Won't Membership Queries Help? *J. Comput. Syst. Sci.*, 50(2): 336–355.

Außerlechner, S.; Jacobs, S.; and Khalimov, A. 2016. Tight Cutoffs for Guarded Protocols with Fairness. In *VMCAI*, volume 9583 of *Lecture Notes in Computer Science*, 476–494. Springer.

Beimel, A.; Bergadano, F.; Bshouty, N. H.; Kushilevitz, E.; and Varricchio, S. 2000. Learning functions represented as multiplicity automata. *J. ACM*, 47(3): 506–530.

Bollig, B.; Habermehl, P.; Leucker, M.; and Monmege, B. 2013. A Fresh Approach to Learning Register Automata. In *Developments in Language Theory - 17th International Conference, DLT 2013, Marne-la-Vallée, France, June 18-21, 2013. Proceedings*, 118–130.

Bollig, B.; Katoen, J.; Kern, C.; and Leucker, M. 2010. Learning Communicating Automata from MSCs. *IEEE Trans. Software Eng.*, 36(3): 390–408.

de la Higuera, C. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Decker, N.; Habermehl, P.; Leucker, M.; and Thoma, D. 2014. 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*, 130–149.

Delzanno, G.; Esparza, J.; and Podelski, A. 1999. Constraint-Based Analysis of Broadcast Protocols. In *CSL*, volume 1683 of *Lecture Notes in Computer Science*, 50–66. Springer.

Emerson, E. A.; and Kahlon, V. 2000. Reducing Model Checking of the Many to the Few. In *CADE*, volume 1831 of *Lecture Notes in Computer Science*, 236–254. Springer.

Emerson, E. A.; and Kahlon, V. 2003. Model Checking Guarded Protocols. In *LICS*, 361–370. IEEE Computer Society.

Emerson, E. A.; and Namjoshi, K. S. 1998. On Model Checking for Non-Deterministic Infinite-State Systems. In *LICS*, 70–80. IEEE Computer Society.

Emerson, E. A.; and Namjoshi, K. S. 2003. On Reasoning About Rings. *Int. J. Found. Comput. Sci.*, 14(4): 527–550.

Esparza, J.; Finkel, A.; and Mayr, R. 1999. On the Verification of Broadcast Protocols. In *LICS*, 352–359.

Esparza, J.; Leucker, M.; and Schlund, M. 2011. Learning Workflow Petri Nets. *Fundam. Informaticae*, 113(3-4): 205–228.

Finkel, A.; and Schnoebelen, P. 2001. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2): 63–92.

Fisman, D.; Izsak, N.; and Jacobs, S. 2023. Learning Broadcast Protocols. *CoRR*, abs/2306.14284.

Fisman, D.; Nitay, D.; and Ziv-Ukelson, M. 2023. Learning of Structurally Unambiguous Probabilistic Grammars. *Log. Methods Comput. Sci.*, 19(1).

Geeraerts, G.; Raskin, J.; and Begin, L. V. 2007. Well-structured languages. *Acta Informatica*, 44(3-4): 249–288.

Gold, E. M. 1967. Language Identification in the Limit. *Inf. Control.*, 10(5): 447–474.

Gold, E. M. 1978. Complexity of Automaton Identification from Given Data. *Inf. Control.*, 37(3): 302–320.

Jacobs, S.; and Bloem, R. 2014. Parameterized Synthesis. *Log. Methods Comput. Sci.*, 10(1).

Lingg, J.; de Oliveira Oliveira, M.; and Wolf, P. 2024. Learning from positive and negative examples: New proof for binary alphabets. *Information Processing Letters*, 183: 106427.

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

Peled, D. A.; Vardi, M. Y.; and Yannakakis, M. 2002. Black Box Checking. *J. Autom. Lang. Comb.*, 7(2): 225–246.

Roy, R.; Gaglione, J.; Baharisangari, N.; Neider, D.; Xu, Z.; and Topcu, U. 2023. Learning Interpretable Temporal Properties from Positive Examples Only. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 6507–6515.

Vaandrager, F. W. 2017. Model learning. *Commun. ACM*, 60(2): 86–95.