# Insights into Learning Broadcast Protocols
## (Short Paper)

Dana Fisman[1], Noa Izsak[1,2(✉)], and Swen Jacobs[2]

[1] Ben Gurion University, Beer-Sheva, Israel
dana@bgu.ac.il, izsak@post.bgu.ac.il
[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
jacobs@cispa.de

**Abstract.** Broadcast protocols (BPs) are a formal model of distributed systems with an unbounded number of processes communicating through broadcasts. We study the problem of passively learning BPs from execution traces, focusing on the class of *fine BPs* which does not have hidden states and admits a cutoff. We present a passive learning algorithm with a constraint-based approach that guarantees consistency with the sample, and returns a minimal equivalent BP if the sample is sufficiently complete (i.e., subsumes a characteristic set). Furthermore, we describe *LeoParDS*, the first tool that implements these techniques, supporting the practical inference of fine BPs, as well as tasks that include sample generation and approximate equivalence checking.

This work was previously published at AAAI'24 [8] and later implemented at ATVA'24 [12]. We summarize its main results here to foster discussion within the cybersecurity and verification community. This short paper is intended as a concise overview for readers unfamiliar with both prior publications.

**Keywords:** Learning Theory · Broadcast Protocols · Multiagent Systems

## 1 Introduction

Learning computational models has long attracted interest in artificial intelligence and formal verification, e.g., [1,9,15]. In particular, concurrent computational models pose significant challenges for learning due to their succinctness and the absence of canonical minimal representations. While previous learning techniques have addressed models with a fixed number of processes (such as communicating automata [3], workflow Petri nets [7], and negotiation protocols [14]), they fall short for parameterized protocols, which are required to work correctly for *any* number of processes.

Broadcast protocols (BPs) are an expressive class of concurrent models with synchronous broadcast communication. They have previously been considered in

D. Fisman, N. Izsak and S. Jacobs—Contributed equally.

A. Akavia et al. (Eds.): CSCML 2025, LNCS 16244, pp. 306–313, 2026.
https://doi.org/10.1007/978-3-032-10759-6_20

the context of parameterized verification [4,6], where one seeks correctness guarantees for all system sizes. In parameterized verification, the notion of a cutoff provides a promising way to reduce the reasoning about infinitely many system instances to a finite-size representative. However, the application of cutoff concepts within a learning framework for BPs had not previously been explored.

Our work proposes a learning framework for BPs under two conditions: (1) the BP has no hidden states, and (2) there exists a cutoff (i.e., a number beyond which its language stabilizes). We call these *fine* BPs, and note that many broadcast protocols satisfy these constraints, making this a meaningful target class. Our main contributions are as follows.

- A constraint-based passive learning algorithm for fine BPs, using SMT solver to infer models from execution traces.
- Hardness results showing that consistency is an NP-hard problem, characteristic sets may be exponentially large, and fine BPs are not polynomially predictable under standard cryptographic assumptions.
- Implementation of these techniques in the tool *LeoParDS*, supporting sample generation, random BP synthesis, and approximate equivalence checking.

These results extend previous learning frameworks by not requiring a fixed system size or a known cutoff, thus introducing a new way of passive learning for parameterized concurrent models. For complete and detailed proofs, please refer to "Learning Broadcast Protocols" (AAAI 2024) publication [8], and for implementation details, see the "Learning Broadcast Protocols with LeoParDS" (ATVA 2024) paper [12].

## 2 Preliminaries

This section briefly recalls key definitions from [8,11,12] for self-containment.

*Broadcast Protocols (BPs).* Broadcast protocols (BPs) [5,6] are finite-state systems that use synchronous broadcast messages. Formally, a BP $\mathcal{B} = (S, s_0, L, R)$ consists of a finite state set $S$ with initial state $s_0 \in S$, a set of labels $L = \{a!!, a??|a \in A\}$ for a finite set of actions $A$, where $a!!$ is a *broadcast sending transition* and $a??$ is a *broadcast receiving transition* (or *response*), and a transition relation $R \subseteq S \times L \times S$. All processes execute the same protocol, $\mathcal{B}$, and the system with $n$ identical processes is denoted $\mathcal{B}^n$. A global step consists of *one* process broadcasting $a$ (taking an $a!!$ transition) while all others respond simultaneously (taking an $a??$ transition). Processes can always respond, yet at any step only a single action is broadcasted. States with no outgoing sending transitions are called *hidden*; in this work, we focus on BPs without hidden states, which is a mild restriction.

*Semantics and Cutoffs.* The semantics of $\mathcal{B}^n$ can be expressed by tracking how processes move between the states when a broadcast action occurs. Feasible words in $\mathcal{B}^n$ form the language $L(\mathcal{B}^n)$, and the language of the protocol $\mathcal{B}$ is: $L(\mathcal{B}) = \bigcup_{n\in\mathbb{N}} L(\mathcal{B}^n)$. A word $w \in A^*$ is feasible in $\mathcal{B}^n$ if there exists an execution trace of $\mathcal{B}^n$ based on the sequence of actions $w$.

A BP, $\mathcal{B}$, has a *cutoff* $c$ if its language stabilizes for all $n \geq c$, that is, $L(\mathcal{B}^n) = L(\mathcal{B}^c)$ for any $n \geq c$. If a BP has a cutoff and no hidden states, we call it a *fine BP*. Two BPs are equivalent if they accept the same language. We note that minimal fine BPs (and thus BPs in general) are not unique up to isomorphism (see Fig. 1 for an example).

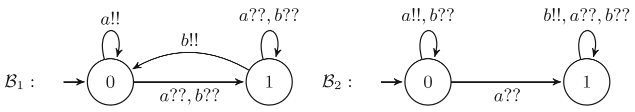

**Fig. 1.** Two non-isomorphic minimal fine BPs: $L(\mathcal{B}_1) = L(\mathcal{B}_2) = a(a \cup b)^*$.

*Learning Problems.* We consider the following learning problems for BPs. A *sample* is a set of labeled words that indicates their feasibility/infeasibility in $\mathcal{B}^n$. The key questions are:

- **Inference:** Given a consistent sample, infer a BP consistent with it.
- **Consistency:** Is there a BP with at most $k$ states consistent with a sample.
- **Polynomial Data:** Can characteristic sets be of polynomial size?
- **Polynomial Predictability:** Can a learner classify unknown words with high probability after polynomially many queries?

Further algebraic details of transition matrices and state vectors along with detailed definitions, including the formal language of membership and draw queries, appear in our full version [8].

## 3 Properties of Broadcast Protocols

This section summarizes key properties of broadcast protocols needed for our learning results. First, note that the language of any BP is prefix-closed, and adding processes can only increase feasible behaviors:

**Lemma 1 (Prefix-closedness and Monotonicity** [8]**).** *If $\mathcal{B}$ is a BP, then $L(\mathcal{B})$ is prefix-closed. Moreover, $L(\mathcal{B}^k) \subseteq L(\mathcal{B}^\ell)$ for all $\ell > k$.*

Second, BPs exhibit a type of progressive growth across process counts:

**Lemma 2 (Step-by-step Progress** [8]**).** *Let $w \in A^*$, $a \in A$, and $m < n$. If $w \in L(\mathcal{B}^m)$ and $wa \notin L(\mathcal{B}^m)$, yet $wa \in L(\mathcal{B}^n)$, then $wa \in L(\mathcal{B}^{m+1})$.*

Third, even though fine BPs do not have a unique canonical minimal representation (see Fig. 1), there is a consistent correspondence among their states:

**Lemma 3 (Relation Between Minimal Equivalent Fine BPs** [8]**).** *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be minimal fine BPs with $L(\mathcal{B}_1) = L(\mathcal{B}_2)$. Then for every $m \in \mathbb{N}$, it holds that $L(\mathcal{B}_1^m) = L(\mathcal{B}_2^m)$ and there is a bijection between their states that preserve the sets of enabled sending actions, and their reachable configurations are consistent under this bijection.*

These results lay the foundation for the inference algorithm described in Sect. 4.

## 4 Inferring a BP from a Sample

This section summarizes the inference algorithm developed in our previous work [8,12] for learning fine broadcast protocols from samples. Full technical details and proofs are given in that extended version.

Given a sample $\mathcal{S}$ of labeled traces, the algorithm $\mathfrak{I}$ constructs a BP $\mathcal{B}_{\mathcal{S}}$ consistent with $\mathcal{S}$. Its key idea is to encode constraints on partial functions:

- $f^{\mathsf{st}} : A \to S$, mapping each action to the state enabling its broadcast;
- $f^{!!} : A \to S$, giving the post-broadcast state;
- $f^{??}_a : S \to S$ for each $a \in A$, $s \in S$, specifying where receivers go on $a$.

These functions define the transition relation of $\mathcal{B}_{\mathcal{S}}$. The sample provides positive and negative examples of feasible words with various amounts of process counts, allowing constraints to rule out inconsistent states and to ensure specification satisfaction. In essence, the algorithm ensures:

- Consistency with positive example (i.e., with feasible words),
- Rejection of negative example,
- No hidden states,
- Partitioning of actions among states to match the observed behavior.

The resulting constraints are encoded in the theory of equality with uninterpreted functions (EUF) and can be solved with standard SMT solvers. A sample-consistent valuation defines a BP consistent with $\mathcal{S}$.

**Theorem 1 (Correctness [8]).** *Let $\mathcal{S}$ be a sample consistent with some fine BP. Then any $\mathcal{B}_{\mathcal{S}}$ satisfying the constraints is consistent with $\mathcal{S}$.*

The full constraint definitions, the induction-based correctness proof appear in [8] and the SMT encoding appear in [12] for completeness.

**Corollary 1 ([8]).** *Algorithm $\mathfrak{I}$ is a sound inference algorithm for fine BPs and can be implemented using existing SMT tools.*

## 5 Returning a Minimal BP

This section summarizes how a sufficiently complete sample allows returning a minimal equivalent BP, rather than just any consistent one.

We define a *characteristic set* (CS) for a fine BP $\mathcal{B}$ as a sample (a finite set) rich enough to capture the BP behavioral properties, that is, fully separate its states and transitions. We provide a constructive algorithm $\mathfrak{G}$ that explores a finite unfolding of feasible traces (via a prefix-tree-like exploration) to produce such a CS. Intuitively, $\mathfrak{G}$ explores all feasible words up to the cutoff (which is detected on the fly). It labels each node with reachable state-vectors, and terminates once the exploration stabilizes (i.e., the cutoff is detected).

**Theorem 2 (Minimal BP Recovery** [8]**).** *For any minimal fine BP $\mathcal{B}$, there exists a characteristic set $\mathcal{S}_B$ such that an inference algorithm $\mathfrak{A}$, given any sample subsuming $\mathcal{S}_B$, returns a minimal fine-BP equivalent to $\mathcal{B}$.*

Unfortunately, characteristic sets may be exponentially large:

**Theorem 3 (CS Lower Bound** [8]**).** *There exists a family of fine BPs with no characteristic set of polynomial size.*

A consequence of this construction is that fine BPs can be exponentially more succinct than the minimal DFA for the same language:

**Corollary 2 (**[8]**).** *There exists a family of fine BPs whose minimal equivalent DFA is exponentially larger.*

These results clarify the sample complexity limits to achieve minimal inference of fine BPs. The complete construction details and proofs are given in [8].

## 6 Consistency is NP-Hard for Fine BPs

We show that deciding BP consistency is NP-hard, even for fine-BPs. While DFA consistency is known to be NP-hard [10], a DFA is not a special case of a fine BP; however, a fine-BP can simulate any DFA with a modest overhead.

Figure 2 illustrates the key construction. It embeds the DFA states into a BP states and uses additional broadcast transitions to capture accepting/rejecting behavior while maintaining no hidden states. Extra symbols ($i$, \$, $\top$, $\bot$, $x$) coordinate the simulation, and projections recover the original DFA language.

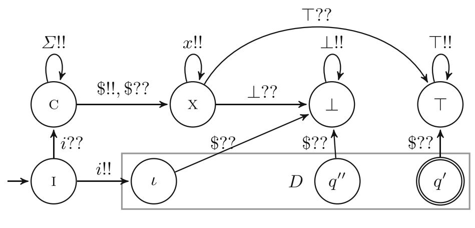

**Fig. 2.** Reduction from DFA-consistency to BP-consistency.

**Lemma 4 (BP Simulation of DFA** [8]**).** *Any nontrivial regular language $L$ over $\Sigma$ with DFA of $n$ states can be simulated by a fine BP with at most $n + 5$ states so that the feasible BP words project back to $L$.*

This simulation enables a reduction from DFA-consistency:

**Theorem 4** **([8]).** *BP consistency is NP-hard.*

The reduction is polynomial and it preserves the feasibility (or infeasibility) of words under the mentioned projection. See the extended version [8] for a full construction and an alternative reduction from all-eq-SAT [13].

Furthermore, since the feasibility of a word $w \in A^*$ in $\mathcal{B}^n$ can be checked in polynomial time, BP-consistency is NP-complete.

**Corollary 3** **([8]).** *BP consistency is NP-complete.*

## 7 Polynomial Predictability

Finally, we discuss the predictability of fine BPs under the polynomial-predictability learning paradigm which is an active learning problem. Roughly speaking, a class $\mathcal{C}$ is *polynomially predictable* if there is a learner that, given polynomially many membership or draw queries, can predict the membership of a random test word with high accuracy. See Angluin and Kharitonov [2] for formal background.

We show that under standard cryptographic assumptions, fine BPs are not polynomially predictable. The key idea is a reduction from the intersection of DFAs, known to be polynomially-unpredictable in this sense. Our construction uses a BP that simulates multiple DFAs concurrently, routing one process per DFA to its initial state, while controlling the alphabet via an additional broadcast controller, as shown in Fig. 3.

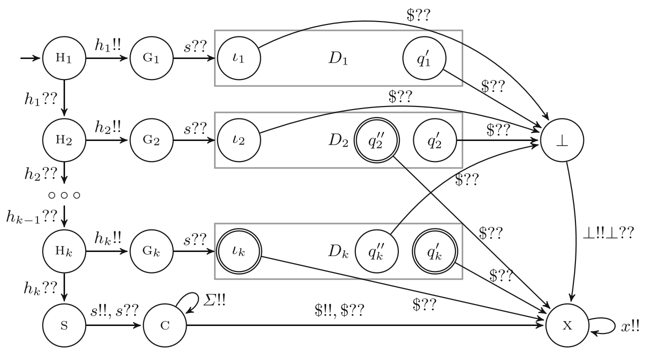

**Fig. 3.** A BP simulating intersection of $k$ DFAs.

**Theorem 5 (Predictability Lower Bound [8]).** *Assuming the hardness of quadratic residuosity, RSA inversion, or factoring Blum integers, BPs are not polynomially predictable with membership queries.*

The proof is for *fine*-BPs, but thus follows for broadcast protocols, in general. The reduction follows Angluin and Kharitonov's classical techniques [2], adapting them to broadcast-based models.

## 8 Conclusion

This paper summarized our investigation into the learnability of fine broadcast protocols, the first framework for learning concurrent models without assuming a fixed number of processes. On the positive side, we presented a passive inference algorithm capable of returning a consistent BP, and even a minimal equivalent BP given a sufficiently complete sample. On the negative side, we showed that consistency is NP-hard, characteristic sets can be exponentially large, and fine BPs are not polynomially predictable.

To bridge theory and practice, we implemented these methods in *LeoParDS*, the first tool for learning broadcast protocols in a parameterized setting. LeoParDS supports characteristic set generation, random sample and BP generation, equivalence checking, and demonstrates that these learning techniques can be applied in practice despite their worst-case theoretical complexity[1]. Full details and experimental results are provided in [12].

**Acknowledgment.** Noa Izsak carried out this work in part as a member of the Saarbrücken Graduate School of Computer Science.

## References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987)
2. Angluin, D., Kharitonov, M.: When won't membership queries help? J. Comput. Syst. Sci. **50**(2), 336–355 (1995)
3. Bollig, B., Katoen, J., Kern, C., Leucker, M.: Learning communicating automata from MSCs. IEEE Trans. Software Eng. **36**(3), 390–408 (2010)
4. Emerson, E.A., Namjoshi, K.S.: Automatic verification of parameterized synchronous systems. In: Alur, R., Henzinger, T.A. (eds.) Computer Aided Verification, pp. 87–98 (1996)
5. Emerson, E.A., Namjoshi, K.S.: On model checking for non-deterministic infinite-state systems. In: LICS, pp. 70–80. IEEE Computer Society (1998)
6. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: LICS, pp. 352–359 (1999)
7. Esparza, J., Leucker, M., Schlund, M.: Learning workflow petri nets. Fundam. Informaticae **113**(3–4), 205–228 (2011)
8. Fisman, D., Izsak, N., Jacobs, S.: Learning broadcast protocols. In: Proceedings of the 38th Annual AAAI Conference on Artificial Intelligence, Vancouver, Canada, vol. 11, pp. 12016–12023 (2024)
9. Gold, E.M.: Language identification in the limit. Inf. Control **10**(5), 447–474 (1967)
10. Gold, E.M.: Complexity of automaton identification from given data. Inf. Control **37**(3), 302–320 (1978)
11. Izsak, N.: Learning broadcast protocol. Master's thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel (2023). https://primo.bgu.ac.il/permalink/972BGU_INST/23v028/alma9927020902704361

[1] The tool is available online by GitHub or Zenodo (DOI 10.5281/zenodo.10968037).

12. Izsak, N., Fisman, D., Jacobs, S.: Learning broadcast protocols with leopards. In: Akshay, S., Niemetz, A., Sankaranarayanan, S. (eds.) Automated Technology for Verification and Analysis, pp. 220–234. Springer, Cham (2025)
13. Lingg, J., de Oliveira Oliveira, M., Wolf, P.: Learning from positive and negative examples: new proof for binary alphabets. Inf. Process. Lett. **183**, 106427 (2024)
14. Muscholl, A., Walukiewicz, I.: Active learning for sound negotiations. In: LICS 2022, pp. 21:1–21:12 (2022)
15. Vaandrager, F.W.: Model learning. Commun. ACM **60**(2), 86–95 (2017)