Learning Formal Models of Parametrized Systems*

Dana Fisman[1,†], Noa Izsak[2,*,†] and Swen Jacobs[2,†]

[1]Ben-Gurion University of the Negev, Beer-Sheva, Israel
[2]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

Passive learning of formal models from observed system behavior has produced a rich body of results for sequential computational models. Extending this success to *concurrent* systems poses fundamental new challenges: concurrent models are more succinct than their sequential counterparts, they often lack a unique minimal representative, and their semantics depends on the interplay of multiple simultaneous processes. A common restriction in prior work on learning concurrent models is the assumption of a *fixed*, *known* number of processes—an assumption at odds with *parametrized* systems, which are designed to operate correctly for *any* number of processes. We summarize our recent work [1, 2] on passive learning of *broadcast protocols* (BPs), a well-studied model of parametrized concurrent systems. For the class of *fine BPs*—BPs with no hidden states and a finite cutoff—we establish a comprehensive learnability picture: a constraint-based inference algorithm that infers a BP consistent with a given sample, returning a minimal equivalent BP when the sample subsumes a characteristic set; hardness results showing that consistency is NP-complete, that characteristic sets may be exponentially large, and that fine BPs are not polynomially predictable; and LeoParDS, the first tool implementing these techniques in the parametrized setting.

Keywords

broadcast protocols, parametrized systems, automata theory, passive learning, distributed systems verification

1. Introduction

Passive learning of formal models—inferring a finite-state representation from labeled samples of system behavior—has a long history connecting the AI and formal methods communities [5, 6, 7]. From the perspective of formal methods, a learned model can serve directly as a specification for subsequent verification or synthesis, enabling a *black-box* analysis of systems whose internals are unavailable [8]. From the perspective of AI, formal models provide interpretable structured representations of system behavior amenable to reasoning and guarantee.

The majority of this work targets sequential systems. Extending it to concurrent systems is substantially harder for two reasons. First, concurrent models are typically more succinct than their sequential counterparts, admitting compact representations of behaviors that would require exponentially larger sequential models. Second, and more fundamentally, concurrent models often lack a unique canonical minimal representative—a cornerstone of classical automata learning that is absent here.

Existing results on learning concurrent models (e.g., communicating automata [9], workflow Petri nets [10], negotiation protocols [11]) share a common restriction: they assume the number of interacting processes is *fixed* and *known in advance*. While natural for closed systems, this assumption is incompatible with *parametrized distributed systems*, which must behave correctly for *any* number of processes; in particular, assuming a fixed number of processes is equivalent to assuming the cutoff is known, an assumption our framework does not make.

8th International Workshop on Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis. OVERLAY 2026 @ FLoC, July 18–19, Lisbon, Portugal

*This paper summarizes results previously published at AAAI'24 [1] and ATVA'24 [2]. The tool is available on Zenodo (10.5281/zenodo.10968037). For overviews aimed at the security and knowledge-representation communities, see [3, 4].

*Corresponding author.
†These authors contributed equally.
noa.izsak@cispa.de (N. Izsak)
0000-0002-6015-4170 (D. Fisman); 0009-0004-1333-2490 (N. Izsak); 0000-0002-9051-4050 (S. Jacobs)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-09

CEUR-WS.org/Vol-4324/paper8.pdf

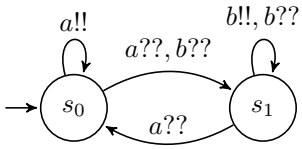


Figure 1: A fine BP B_{ex}. With one process: only a^* is feasible; $b!!$ requires s_1 to be occupied, reachable only when a second process responds to action a from s_0. With two processes: ab is feasible (yet $ab \notin L(B^1_{ex})$). Feasibility thus depends on the number of processes, a core challenge of parametrized learning.

Broadcast protocols [12] are a natural and well-studied model for this setting. In a broadcast protocol (BP), a single process synchronously communicates an action to an arbitrary number of peers. BPs are one of the most expressive concurrent models for which some parametrized verification problems remain decidable [13], and are strictly more expressive than other standard communication primitives such as pairwise rendezvous or disjunctive guards [14]—making them a meaningful target for learning: a BP inferred from a sample can serve as a formal specification for subsequent parametrized verification, leveraging the rich existing theory for this model.

We summarize the first passive learning framework for BPs in the fully parametrized setting [1, 2].

2. Broadcast protocols

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 label set $L = \{a!!, a?? \mid a \in A\}$ for a finite set of actions A, and a transition relation $R \subseteq S \times L \times S$. A transition labeled $a!!$ is a *broadcast send*; a transition labeled $a??$ is a *broadcast receive*. Every state has exactly one outgoing receive transition per action. Following [12], we assume that every action $a \in A$ has a unique origin state, i.e., $R \cap S \times \{a!!\} \times S$ is a singleton; we denote this state as $f^{\mathrm{st}}(a)$. For each state $s \in S$, let $f^{\mathrm{act}}(s) = \{a \in A \mid f^{\mathrm{st}}(a) = s\}$ denote the set of actions that are *enabled* in s.

The system B^n consists of $n \in \mathbb{N}$ identical processes executing B. A *configuration* of B^n is a vector $\mathbf{q} \in \mathbb{N}^{|S|}$ where $\mathbf{q}[s]$ counts the number of processes currently in state s; the initial configuration places all n processes in s_0. In each global step, one process broadcasts an action a (taking its $a!!$ transition), while all other processes in the system simultaneously take their $a??$ transition from their current state; this updates the configuration accordingly. A state s is *occupied* in configuration $\mathbf{q}$ if $\mathbf{q}[s] > 0$. A word $w \in A^*$ is *feasible* in B^n if B^n has an execution producing w, written $w \in L(B^n)$. The *language* of B is $L(B) = \bigcup_{n \in \mathbb{N}} L(B^n)$.

Figure 1 illustrates the central difficulty of the parametrized learning setting: the same word may be feasible with n processes but not with fewer. Two structural properties underpin our learning results.

Lemma 1 ([1]). *Let B be a BP: $L(B^n)$ is prefix-closed for every n, and $L(B^k) \subseteq L(B^\ell)$ for all $k \leq \ell$.*

A BP B has a *cutoff* c if $L(B^c) = L(B^n)$ for all $n \geq c$. A state $s \in S$ is *hidden* if $f^{\mathrm{act}}(s) = \emptyset$. A BP with no hidden states and a finite cutoff is called *fine*. Importantly, the cutoff is a property of the target protocol, and need not be known to the learner: the inference algorithm of Section 3 operates without any assumptions on its value. This distinguishes our setting from prior work on learning concurrent models [9, 10, 11], where assuming a fixed number of processes is equivalent to assuming a known cutoff. Note that many protocols from the parametrized verification literature are fine (e.g., the MESI cache coherence protocol [12]), and when a cutoff exists $L(B)$ is regular. Nevertheless, unlike DFAs, minimal fine BPs are not unique up to isomorphism: two non-isomorphic BPs may accept the same language. The following lemma shows that minimal equivalent fine BPs nonetheless share a tight structural correspondence and is the key enabler of the minimality guarantee in Theorem 1.

Lemma 2 ([1]). *Let B_1 and B_2 be minimal fine BPs with $L(B_1) = L(B_2)$, and state sets S_1 and S_2 resp. There exists a bijection $h : S_1 \to S_2$ such that $f^{\mathrm{act}}(s) = f^{\mathrm{act}}(h(s))$ for every $s \in S_1$, and for every word $w \in A^*$ and every $n \in \mathbb{N}$, a state $s \in S_1$ is occupied in the configuration reached by B_1^n after executing w iff $h(s) \in S_2$ is occupied in the configuration reached by B_2^n after executing w.*

3. Learning fine broadcast protocols

3.1. Setting

A *sample* $\mathcal{S}$ is a finite set of triples (w, n, b)—also referred to as *labeled words*—where $w \in A^*$, $n \in \mathbb{N}$ and $b \in \{\mathrm{T}, \mathrm{F}\}$ indicates whether w is feasible in B^n. A sample is *consistent* with B if all $(w, n, b) \in \mathcal{S}$ satisfy: $w \in L(B^n)$ if $b = \mathrm{T}$ and $w \notin L(B^n)$ if $b = \mathrm{F}$. Crucially, a single sample may contain labeled words from instances of *different* sizes, and the learner is given neither a bound on the number of processes nor the value of the cutoff.

We study four learnability problems for the class of fine BPs: (1) Inference – return a consistent fine BP from a sample $\mathcal{S}$; (2) Consistency – does a fine BP with $\leq k$ states consistent with $\mathcal{S}$ exist?; (3) Polynomial Data – do polynomial-size characteristic sets suffice?; and (4) Polynomial Predictability – can an active learner classify unseen words efficiently using polynomially many queries?

3.2. Inference algorithm

Our algorithm $\mathcal{I}$ [1] constructs a BP $B_{\mathcal{S}}$ consistent with $\mathcal{S}$ via an SMT encoding. Rather than constructing the BP explicitly, $\mathcal{I}$ represents it through three partial functions: $f^{\mathrm{st}} : A \to S$ (state enabling each action's send), $f^{!!} : A \to S$ (post-send state), and $f_a^{??} : S \to S$ (receiving transition for action $a \in A$). These functions, together with Lemma 1, yield a system of constraints in the theory of equality with uninterpreted functions (EUF): positive examples ($b = \mathrm{T}$) require the existence of a consistent execution, and negative examples ($b = \mathrm{F}$) require that no execution exists, with feasibility propagated across process counts via monotonicity (i.e., $L(B^k) \subseteq L(B^\ell)$ for all $k \leq \ell$). EUF is decidable and supported by standard SMT solvers such as Z3 [15].

Theorem 1 ([1]). *Let $\mathcal{S}$ be a sample consistent with some fine BP. Any BP $B_{\mathcal{S}}$ satisfying the constraints of $\mathcal{I}$ is consistent with $\mathcal{S}$. Furthermore, for every minimal fine BP B, there exists a characteristic set $\mathcal{S}_B$ such that if $\mathcal{S}$ subsumes $\mathcal{S}_B$, then $B_{\mathcal{S}}$ is minimal and $L(B_{\mathcal{S}}) = L(B)$.*

The characteristic set (CS) $\mathcal{S}_B$ is constructed by an algorithm G that explores the feasibility of words across increasing process counts, detecting the cutoff on the fly by observing when the language stabilizes. In effect, neither G nor $\mathcal{I}$ requires the cutoff to be known in advance: the cutoff is implicitly witnessed by the structure of $\mathcal{S}_B$. The minimality claim then follows from Lemma 2: a sample subsuming $\mathcal{S}_B$ forces the constraint system to recover exactly the f^{act}-partition of the target, up to bijection h.

3.3. Hardness results

Together, the three hardness results provide a broad picture regarding the learnability of fine BPs.

Consistency is NP-complete. Given a sample $\mathcal{S}$ and a bound $k \in \mathbb{N}$, deciding whether there exists a consistent fine BP with $\leq k$ states is NP-hard, by reduction from DFA consistency [16]. The reduction embeds any DFA into a fine BP with at most five additional states, encoding acceptance via the feasibility of suitably structured words. NP-membership follows since feasibility in B^n is checkable in polynomial time by tracking the configuration of B^n along w.

Exponential characteristic sets. There exists a family of fine BPs for which there is no CS of polynomial size [1]. As a corollary, fine BPs can be exponentially more succinct than the minimal DFA for the same language—a desirable property for representing distributed protocols compactly.

Fine BPs are not polynomially predictable. Under standard cryptographic assumptions (hardness of the quadratic residuosity problem, RSA inversion, or factoring Blum integers), fine BPs are not polynomially predictable with membership queries [17]. The proof reduces from the polynomial-unpredictability of DFA intersections: a fine BP can simulate k DFAs running concurrently.

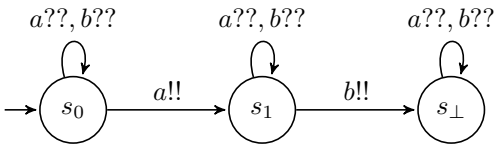


Figure 2: A non-fine BP B: state $s_\perp$ is hidden, and B does not have a cutoff. The language $L(B^n)$ consists of all prefixes of Dyck words with at most n opening brackets, identifying '(' with a and ')' with b.

4. LeoParDS

LeoParDS (*Learning of Parametrized Distributed Systems*) [2] is the first tool for passive learning of broadcast protocols in the parametrized setting. Built on the Z3 SMT solver, it implements the algorithm of Theorem 1 and supports four additional tasks: characteristic set generation, random sample generation, BP generation, and approximate equivalence checking between two BPs.

We evaluated LeoParDS on over 4,000 randomly generated BPs with up to 20 states and 8 actions. Despite the worst-case exponential complexity implied by the hardness results, more than 70% of the instances were solved within 5 minutes—the gap between worst-case and typical-case behavior is significant, and the inference algorithm of Theorem 1 is practically applicable.

5. Conclusion and open directions

We have presented the first passive learning framework for broadcast protocols in the parametrized setting, yielding a wide learnability picture: a sound inference algorithm with minimality guarantees, matching hardness results, and a practical tool.

Learning beyond fine BPs. The restriction to fine BPs is essential: the existence of a characteristic set is guaranteed precisely because fine BPs have a cutoff and hence a regular language. BPs that are not fine need not have a cutoff, and their language need not be regular (as illustrated by the BP B in Figure 2). Since $a^{n+1}b^{n+1} \in L(B^{n+1}) \setminus L(B^n)$ for every $n \in \mathbb{N}$, no cutoff exists and $L(B)$ is the language of all prefixes of Dyck words (not regular). While fine BPs already present substantial challenges—NP-hard consistency, exponentially large characteristic sets, and non-polynomial predictability—the class of non-fine BPs raises a qualitatively different set of questions, as neither a finite characteristic set nor a regular target can be assumed. Understanding what can and cannot be passively learned in this broader class is an open problem at the intersection of automata learning and formal verification.

Active learning for parametrized systems. The non-polynomial predictability result rules out efficient active learning under standard query models. An active learning approach in the style of Angluin's L^* [6] would issue membership and equivalence queries to an oracle. For sequential models, equivalence queries with counterexamples drive convergence; for parametrized concurrent systems, the structure of a suitable equivalence query is not immediate. Designing a suitable equivalence query that accounts for feasibility across all process counts simultaneously is non-trivial and remains open.

Acknowledgments

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

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic) in order to: grammar and spelling check, sentence rephrasing for conciseness. The authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. Fisman, N. Izsak, S. Jacobs, Learning broadcast protocols, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024) 12016–12023. doi:10.1609/aaai.v38i11.29089.

[2] N. Izsak, D. Fisman, S. Jacobs, Learning broadcast protocols with LeoParDS, in: International Symposium on Automated Technology for Verification and Analysis, Springer, 2024, pp. 220–234.

[3] D. Fisman, N. Izsak, S. Jacobs, Insights into learning broadcast protocols: (short paper), in: International Symposium on Cyber Security, Cryptology, and Machine Learning, Springer, 2025, pp. 306–313.

[4] N. Izsak, S. Jacobs, D. Fisman, Learning broadcast protocols (extended abstract), 2026. URL: https://publications.cispa.de/articles/presentation/Learning_Broadcast_Protocols_Extended_Abstract_/33060734. doi:10.60882/cispa.33060734.v1, presented at the 23rd International Conference on Principles of Knowledge Representation and Reasoning (KR 2026), Recently Published Research (RPR) Track.

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

[6] D. Angluin, Learning regular sets from queries and counterexamples, Inf. Comput. 75 (1987) 87–106.

[7] F. W. Vaandrager, Model learning, Commun. ACM 60 (2017) 86–95.

[8] D. A. Peled, M. Y. Vardi, M. Yannakakis, Black box checking, J. Autom. Lang. Comb. 7 (2002) 225–246.

[9] B. Bollig, J. Katoen, C. Kern, M. Leucker, Learning communicating automata from MSCs, IEEE Trans. Software Eng. 36 (2010) 390–408.

[10] J. Esparza, M. Leucker, M. Schlund, Learning workflow Petri Nets, Fundam. Informaticae 113 (2011) 205–228.

[11] A. Muscholl, I. 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, 2022, pp. 21:1–21:12.

[12] J. Esparza, A. Finkel, R. Mayr, On the verification of broadcast protocols, in: LICS, 1999, pp. 352–359.

[13] E. A. Emerson, K. S. Namjoshi, On model checking for non-deterministic infinite-state systems, in: LICS, IEEE Computer Society, 1998, pp. 70–80.

[14] E. A. Emerson, V. Kahlon, Model checking guarded protocols, in: 18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings., IEEE, 2003, pp. 361–370.

[15] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008. Proceedings, volume 4963 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 337–340. doi:10.1007/978-3-540-78800-3_24.

[16] E. M. Gold, Complexity of automaton identification from given data, Inf. Control. 37 (1978) 302–320.

[17] D. Angluin, M. Kharitonov, When won't membership queries help?, J. Comput. Syst. Sci. 50 (1995) 336–355. doi:10.1006/jcss.1995.1026.