Poster Abstract: PRoTECT: Parallel Construction of Barrier Certificates for Safety Verification of Polynomial Systems

Ben Wooding
School of Computing
Newcastle University, United Kingdom

Viacheslav Horbanov
School of Computing
Newcastle University, United Kingdom

Abolfazl Lavaei
School of Computing
Newcastle University, United Kingdom

ABSTRACT

We develop an open-source software tool, called PRoTECT, using *sum-of-squares (SOS) optimization* to search for barrier certificates (BCs) that verify safety properties over *four classes of polynomial systems*: (i) discrete-time *stochastic* systems, (ii) discrete-time *deterministic* systems, (iii) *continuous-time* stochastic systems, and (iv) *continuous-time* deterministic systems. PRoTECT is the first software tool that offers stochastic BCs while leveraging *parallelism* to efficiently search for a feasible BC for different polynomial degrees. PRoTECT is implemented in Python offering both an application programming interface (API) and a user-friendly graphic user interface (GUI).

1 INTRODUCTION

Formal verification aims to reliably assess whether a (stochastic) dynamical system is guaranteed (with high confidence) to fulfill a desired behavioral specification. This is particularly essential for safety-critical systems where failure can have catastrophic consequences. To ensure the safety of a system, barrier certificates (BCs) have been introduced in the literature [1] to provide assurances of safe behavior across diverse system classes. BCs can directly assess the safe behavior of systems across *continuous spaces* with uncountable states, contrasting with abstraction-based approaches that resort to discretization [2]. In particular, barrier certificates, akin to Lyapunov functions, are functions established over the system's state space,fulfilling specific inequalities concerning both the function itself and the one-step transition (or the flow) of the system. A suitable level set of a BC can segregate an unsafe region from all system trajectories originating from a specified set of initial conditions. Hence, the presence of such a function offers a formal (probabilistic) certification for system safety.

Related Tools. To the best of our knowledge, there exists only one other tool dedicated to the construction of barrier certificates, FOSSIL [3]. This tool aims at finding barrier certificates for discrete- and continuous-time *deterministic* systems using counter-example guided inductive synthesis (CEGIS), while facilitating verification and control synthesis for specifications including safety, reachability, and reach-while-avoid. However, FOSSIL lacks support for *stochastic* systems, whereas in PRoTECT, we provide support for both discrete- and continuous-time *stochastic* systems. Additionally, while FOSSIL can handle nonpolynomial BCs, it does not guarantee termination due to its use of CEGIS approach. In contrast, PRoTECT utilizes sum-of-squares optimization techniques, while enabling the exploitation of *parallelism* to concurrently search for multiple candidate BCs with different degrees, aiming to construct them effectively.

Original contributions. The primary contributions and noteworthy aspects of PRoTECT are as follows:

(i) PRoTECT verifies the safe behavior of *four classes of dynamical systems*: (i) discrete-time *stochastic* systems (dt-SS), (ii) discrete-time *deterministic* systems (dt-DS), (iii) *continuous-time* stochastic systems (ct-SS), and (iv) *continuous-time* deterministic systems (ct-DS). In particular, PRoTECT is the first software tool that designs *stochastic* barrier certificates.
(ii) PRoTECT is implemented in Python using SumOfSquares [4], and leverages *parallelization* to efficiently search for BCs of different degrees, aiming to satisfy the desired safety specifications.
(iii) PRoTECT supports *normal*, *uniform*, and *exponential* noise distributions for dt-SS, as well as *Brownian motion* and *Poisson processes* for ct-SS.
(iv) PRoTECT offers a GUI for all four classes of models, enhancing the tool's accessibility and user-friendliness.

The source code for PRoTECT, along with detailed guidelines on installation and usage, including tutorial videos, are available at:

https://github.com/Kiguli/PRoTECT

Details of the system descriptions, case studies, etc. can be found in the extended version [5].

Poster Description:
The poster includes the following elements:
- Motivation for the uses of PRoTECT;
- The BC conditions to solve for *dt-SS*, *dt-DS*, *ct-SS* and *ct-DS*;
- Tables displaying the results of PRoTECT, including the comparison against FOSSIL;
- Description and pseudo-code for the parallel algorithm;
- And a labeled GUI screenshot for operating PRoTECT.

2 OVERVIEW OF PROTECT

PRoTECT offers functionalities that automatically generate BCs and verify the safety property across *four distinct classes of systems*. The description of the system serves as an input to the tool, triggering the appropriate function. These functions are named as `dt-SS`, `dt-DS`, `ct-SS` and `ct-DS`. Additionally, the geometric characteristics for sets of interest, which define the safety specification, constitute another input to the tool. While a GUI is designed to enhance the user-friendliness of PRoTECT, the BCs may also be verified via configuration files executed through the command line. As the output, PRoTECT returns BC $\mathcal{B}(x)$, initial and unsafe level sets γ and λ, and, for stochastic systems, the confidence level ϕ and a positive value c used to compute the safety probability.

Using methodologies from the *sum-of-squares* (SOS) optimization, facilitated by the SumOfSquares Python toolbox [4], PRoTECT adopts polynomial structures for BCs expressed as $\mathcal{B}(x) = \sum_{j=1}^{z} q_j p_j(x)$, with basis functions $p_j(x)$ that are monomials over

ICCPS'25, May 6–9, 2025, Irvine, California, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

x, and unknown coefficients $q = [q_0, \ldots, q_z] \in \mathbb{R}^z$ that need to be designed. PRoTECT leverages parallelization techniques to facilitate the *simultaneous* verification of multiple BCs, differentiating them based on their polynomial degrees. In deterministic systems, upon finding a feasible BC, the parallel processing is terminated and the valid BC is returned to the user. Conversely, for *stochastic* systems, PRoTECT awaits until all potential solutions are fully processed, subsequently selecting and returning the BC that *offers the highest probabilistic confidence*. This process is detailed in the provided pseudo-code, illustrated in Algorithm 1.

Algorithm 1: *Parallel* Construction of BCs

Input: system Σ, maximum polynomial degree P, *required* parameters K_{req}, *optional* parameters K_{opt}

```
temp = [];
choose function func for Σ to identify the class of system;
forall p ∈ {2, 4, ..., P} in parallel do
    barrier = func(p, K_req, K_opt);
    if barrier is SOS then
        temp.append(barrier);
        if Σ is deterministic then
            // terminate all parallel processes
        end
    end
end
// return element with highest confidence in temp
B(x) = max(temp);
```

Output: barrier certificate $\mathcal{B}(x)$, level sets γ, λ; (confidence ϕ and constant c for dt-SS and ct-SS)

3 BENCHMARKING

All experiments were conducted on a desktop computer (Intel i9-12900). Table 1 compares the efficiency of PRoTECT and FOSSIL in finding BCs for deterministic systems, for both dt-DS and ct-DS. Here, PRoTECT was operated in a *"One-Shot" mode*, searching for a BC without parallelism. The majority of case studies are taken from FOSSIL's own benchmarks. All examples in this table, including the FOSSIL configuration files, can be located in the folder `/ex/benchmarks-deterministic`. Notably, PRoTECT demonstrates always better time efficiency for case studies up to four dimensions, while FOSSIL may (but not always) perform better in cases with six and eight dimensions. This can be attributed to the increase in Lagrangian multipliers in PRoTECT, which also necessitates designing more decision variables when searching for higher-degree BCs. However, as demonstrated in the higher-dimensional systems hi-ord$_6$-2 and hi-ord$_8$-2, FOSSIL performs significantly worse than PRoTECT, which exhibits superior time efficiency in solving these problems. This is primarily because FOSSIL relies on a neural network approach to solve the problem; the tool's runtime varies depending on the problem, as well as the verification process using satisfiability modulo theories (SMT) solvers. We re-highlight that FOSSIL does not guarantee convergence to a solution.

In *"Parallel" mode*, PRoTECT simultaneously searches for a valid BC of varying degrees, such as 2, 4, and 6. Once a valid BC is found, the other threads are terminated. The parallelism introduces minimal overhead when computing multiple results, and we found that running the degrees in parallel is 19-27% faster than computing the same degrees (2, 4, 6) sequentially. The cases in Table 1 all have valid degree 2 barrier certificates, and we omit the parallelism results due to space constraints.

Table 2 employs PRoTECT for *stochastic* benchmarks. The "One-Shot" mode is omitted for the sake of space constraint. PRoTECT optimizes the parameters during the SOS formulation, where either a level set or the overall confidence is fixed a priori. We highlight again PRoTECT is the first tool in the literature to offer *stochastic BCs*. A maximum degree of 6 is set and computations are run in parallel, returning the BC with the highest confidence. Unlike the deterministic setting, PRoTECT awaits barriers of all degrees to terminate and return the one with the *highest confidence*. Consequently, stochastic case studies typically require longer completion times compared to the "One-Shot" mode or "Parallel" mode for deterministic systems, but at the gain of offering the highest confidence. For instance, ex_nonlin$_1$ had a confidence improvement of 32% compared with the selecting degree 4 in "One-Shot" mode. All case study descriptions and figures can be found in [5].

Table 1: Efficiency comparison of BC construction for deterministic systems: PRoTECT vs. FOSSIL.

One Shot Deterministic Systems

			PRoTECT	FOSSIL
	n	system	(sec)	(sec)
1D system	1	dt-DS	0.09	0.20
DC Motor	2	dt-DS	0.20	0.22
barr$_2$room$_{DT}$	2	dt-DS	0.17	0.44
Jet Engine	2	ct-DS	0.18	0.28
hi-ord$_4$	4	ct-DS	1.30	27.5
hi-ord$_6$-1	6	ct-DS	17.0	11.6
hi-ord$_6$-2	6	ct-DS	18.9	652
hi-ord$_8$-1	8	ct-DS	172	21.3
hi-ord$_8$-2	8	ct-DS	175	569

Table 2: Efficiency evaluation in BC construction for *stochastic systems* via PRoTECT. We present case studies across three barrier degrees (2, 4, and 6), returning the barrier with the highest confidence ϕ.

Parallel Stochastic Systems

	n	system	ϕ	time (sec)
RoomTemp	1	ct-SS	0.99	0.33
RoomTemp	1	dt-SS	0.99	0.53
VDP	2	dt-SS	0.88	14.3
ex_lin$_1$	2	ct-SS	0.95	1.73
ex_nonlin$_1$	2	ct-SS	0.72	1.81
TwoTanks	2	dt-SS	0.99	5.14
RoomTemp	3	dt-SS	0.99	1501
hi-ord$_4$	4	ct-SS	0.99	1308

REFERENCES

[1] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[2] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani, "Automated verification and synthesis of stochastic hybrid systems: A survey," *Automatica*, vol. 146, 2022.

[3] A. Edwards, A. Peruffo, and A. Abate, "FOSSIL 2.0: Formal Certificate Synthesis for the Verification and Control of Dynamical Models," *arXiv:2311.09793*, 2023.

[4] C. Yuan, "SumOfSquares.py." [Online]. Available: https://github.com/yuanchenyang/SumOfSquares.py

[5] B. Wooding, V. Horbanov, and A. Lavaei, "PRoTECT: Parallelized Construction of Safety Barrier Certificates for Nonlinear Polynomial Systems," *arXiv*, pp. 1–27, Apr. 2024.