ARTIFACT EVALUATION INSTRUCTIONS PROTECT: PARALLELIZED CONSTRUCTION OF SAFETY BARRIER CERTIFICATES FOR NONLINEAR POLYNOMIAL SYSTEMS

BEN WOODING, VIACHESLAV HORBANOV, AND ABOLFAZL LAVAEI[1]

[1]SCHOOL OF COMPUTING, NEWCASTLE UNIVERSITY, UNITED KINGDOM

{BEN.WOODING,V.HORBANOV2,ABOLFAZL.LAVAEI}@NEWCASTLE.AC.UK

Welcome to the PRoTECT artifact evaluation instructions. This document explains how to reproduce the results presented in the paper "**PRoTECT**: **P**arallelized Const**R**uction **o**f Safe**T**y Barri**E**r **C**ertificates for Nonlinear Polynomial Sys**T**ems". The GitHub repository with the latest updates can be found at:

https://github.com/Kiguli/PRoTECT

Every release of this repository is assigned a permanent DOI, the latest of which can be accessed here on Zenodo:

https://zenodo.org/doi/10.5281/zenodo.11085376.

As a form of documentation, we have provided detailed Youtube tutorial videos which explain installation, using the GUI for case studies of the four types of dynamical systems, and editing the example configuration python scripts for use of the tool as an application programming interface (API). Further documentation and details about the tool can be found in the GitHub repository and Section 5 of this document. *These artifact evaluation instructions will replicate the results of Table 1, Table 2, and Table 3 of the paper.* The figures are not replicated since they only visualize the results gathered in these tables.

We expect the artifact evaluation committee to read Sections $1 - 4$ of this guide covering the system requirements, installation, smoke test, and full evaluation of the results of the paper. Section 5 offers comprehensive guidance on applying PRoTECT to custom case studies and highlights additional tool features. Section 6 provides manual installation instructions for addressing any potential issues that may arise.

1. System Requirements for Artifact Evaluation

Virtual Machine with Network Access. Since our tool offers a GUI, we have tailored our artifact evaluation instructions for the Virtual Machine (VM), based on Ubuntu 22.04 LTS, provided by the Artifact Evaluation committee that can be downloaded here on Zenodo (username and password are `tacas23`). We ran all the examples on this VM using an Intel i9-12900 with 24 CPUs, 33GB memory and 8.6GB of swap memory.

Machines with fewer CPUs and less memory should still be able to run PRoTECT smoothly. Generally, each parallel run utilizes one CPU per degree value, while the memory requirements increase with the degree value and the number of dimensions. We estimate the largest example ($\mathtt{hi\_ord}_8$) requires up to 5GB RAM. Running all the results from all tables took about 63 minutes on a machine with Intel i9-129000.

The FOSSIL release 2.0 has been included in the repository in a folder named "fossil-main" which was used for comparison with our tool. We assume the VM will be setup with network access, this is necessary for the installation of FOSSIL for comparison with PRoTECT; however, PRoTECT does not include the FOSSIL dependencies. The network access can also useful for transferring the Mosek License to the VM if a shared folder is not used. Instructions for including network access and the shared folders to the VM in VirtualBox are provided.

Mosek License. All results were obtained using the MOSEK solver, which requires a MOSEK license. Academic users can obtain a free license, and a free trial is also available if needed. You can get a Mosek license at https://www.mosek.com/license/request/?i=acp. You will receive the license file via email, along with instructions on where to place it in your home directory. On the VM, this involves placing the license file, `mosek.lic`, into the `mosek` folder within the home directory (the folder is created during installation).

Tables 1-3 from the paper require the artifact evaluation. We provide shell scripts to run the installation and the artifact evaluation automatically.

2. Installation

2.1. **Setup VM.** It is straightforward to run the VM using VirtualBox installed on the host PC and double clicking the `TACAS23-AEC.ova` file once downloaded (https://zenodo.org/records/7113223 - username and password are `tacas23`) which auto-imports into VirtualBox. The user should enable network access (for FOSSIL dependencies). The use of the shared folder is optional if transferring files to the VM from a host computer. Using VirtualBox, network access can be added by going to `Settings > Network`, then putting a tick in the box `Enable Network Adapter`. For the shared folder go to `Settings > Shared Folders`, then click the blue folder icon with the green plus and choose a folder on the host PC to use as the shared folder. Selecting `Auto-mount` makes it easy to find the shared folder from inside the VM file manager.

2.2. **Install PRoTECT and FOSSIL.**

(1) Download the Zenodo zip file for PRoTECT v1.3 to the home directory, then unzip the folder and change the name of the new folder to PRoTECT (case sensitive); you can delete the zipped version of the folder. Alternatively clone the v1.3 branch of the PRoTECT GitHub page (which is directly

linked to this same Zenodo DOI). For the latter, go to the home directory and clone the repository in a terminal using these three commands sequentially:

```
cd ~
sudo apt-get install git
git clone https://github.com/Kiguli/PRoTECT --branch v1.3
```

(2) Navigate to the folder `bash-scripts` inside the PRoTECT folder and run the shell script that will install PRoTECT and FOSSIL onto the VM; other steps will be completed automatically:

```
cd ~/PRoTECT/bash-scripts
./install_ubuntu22_PRoTECT_and_FOSSIL.sh
```

It is important not to run the second command with sudo, the commands requiring sudo are defined inside this script and it will still request the sudo password from you running it this way. Terminating this command early may cause installation issues as folders are moved around and rerunning the script could throw errors.

(3) A new folder has been created in the home directory called `mosek`, copy your `mosek.lic` file into this folder (see previous section for how to acquire the Mosek license).

(4) Restart the VM (many different dependencies have just been installed; restarting allows the computer to determine all the new PATHs).

3. Smoke Test

Once all the previous installation instructions have been run correctly, and the VM has been restarted, it should be straightforward to run all the results for Tables 1-3. The results for these tables were acquired by running the python scripts in the folder `ex` and not using the GUI. *This is in part because the GUI provides an overhead that somewhat reduces the efficiency of the functions being called, and secondly it provides a fairer comparison against FOSSIL which does not use a GUI.*

To run the smoke test to check that PRoTECT and FOSSIL are both installed correctly, navigate first to the folder `benchmarks-deterministic`:

```
cd ~/PRoTECT/ex/benchmarks-deterministic
```

and run in the terminal:

```
./Table1.sh
```

You can cancel this command after a few seconds using (`Ctrl+C`) and you should see output similar to:

```
Filename:  ex1_dt_DS.py
Gamma:  3.8977793742053635,
```

```
Lambda:  4.039824500833843
Execution Time (PRoTECT): 0.0981757640838623 seconds
Execution Time (FOSSIL): 0.5015462219999982 seconds
-----------------------------
Filename:  ex2_DC_Motor_dt_DS.py
Gamma:  1.199460525159871,
Lambda:  1.2021164206743107
Execution Time (PRoTECT): 0.21004867553710938 seconds
Execution Time (FOSSIL): 0.25898631500000135 seconds
-----------------------------
```

If you receive this, you can consider the smoke test to be passed. If either/both FOSSIL or PRoTECT show `N/A` values, then one or both have not been installed correctly.

If you wish to test the GUI (not required in artifact evaluation), return to the PRoTECT folder and start the GUI via:

```
cd ∼/PRoTECT
python3 main.py
```

4. Full Evaluation

We now provide the details for the evaluation of Tables 1-3. All the results can be acquired in one go using:

```
cd ∼/PRoTECT/ex
./make_tables.sh
```

The `make_tables.sh` script runs three subscripts that each generate Table 1, Table 2, and Table 3. These tables are saved as `*.csv` files in the `ex` folder and can be compared with the results in the paper.

Note: *There may be some variations between results computed on different machines, unless N/A is presented a barrier certificate has been found even if the exact values of the level sets and computation times differ slightly. A-priori known values (such as degree, time horizon $\mathcal{T}$, etc.) are omitted from these tables.*

Running this script took about 63 minutes on a machine with Intel i9-129000. The terminal does not display any information until all three subscripts complete and then prints all the information to the terminal in one go. In particular, each table took respectively:

- Table 1 - 623 seconds,

- Table 2 - 100 seconds,
- Table 3 - 3029 seconds.

Alternatively, the tables can be computed individually, with all results printed to the terminal as they are processed (e.g., starting with Table 2 to estimate the computation time on your machine). The process can be carried out as follows:

Table 1.

```
cd ~/PRoTECT/ex/benchmarks-deterministic
./Table1.sh
```

Table 2.

```
cd ~/PRoTECT/ex/benchmarks-stochastic
./Table2.sh
```

Table 3.

```
cd ~/PRoTECT/ex/benchmarks-stochastic
./Table3.sh
```

The table results will be saved in a `*.csv` file in the directory from which the shell script was executed.

5. Running New Case Studies with PRoTECT

We have aimed to make using PRoTECT for your own examples as straightforward as possible. This can be done in two ways: either through the provided GUI or by using Python scripts that call PRoTECT functions as an API.

5.1. **Graphic User Interface (GUI).** To enhance accessibility and user-friendliness of the tool, PRoTECT offers the Model-View-Presenter architecture incorporating a GUI. Specifically, a GUI strengthens user-friendliness by abstracting away implementation details for the code, allowing for a push-button method to construct barrier certificates. In Fig. 1, color notation is utilized to represent labels by their corresponding color and number. While PRoTECT provides GUIs for all four classes of systems (see Fig. 1 (blue-1)), we only depict it for dt-SS here to avoid excessive information. Our tool offers two implementations, either serial or parallel (red-6). The tool processes the information entered into the GUI before executing the desired function upon pressing the *Find Barrier* button (blue-8). Outputs of barrier certificate $\mathcal{B}(x)$, confidence ϕ, level sets γ and λ, and constant c are displayed at (yellow-1), (yellow-2), and (yellow-3), respectively. Optionally, the GUI allows for the import and export of configuration parameters in JSON format using the *Import Config*

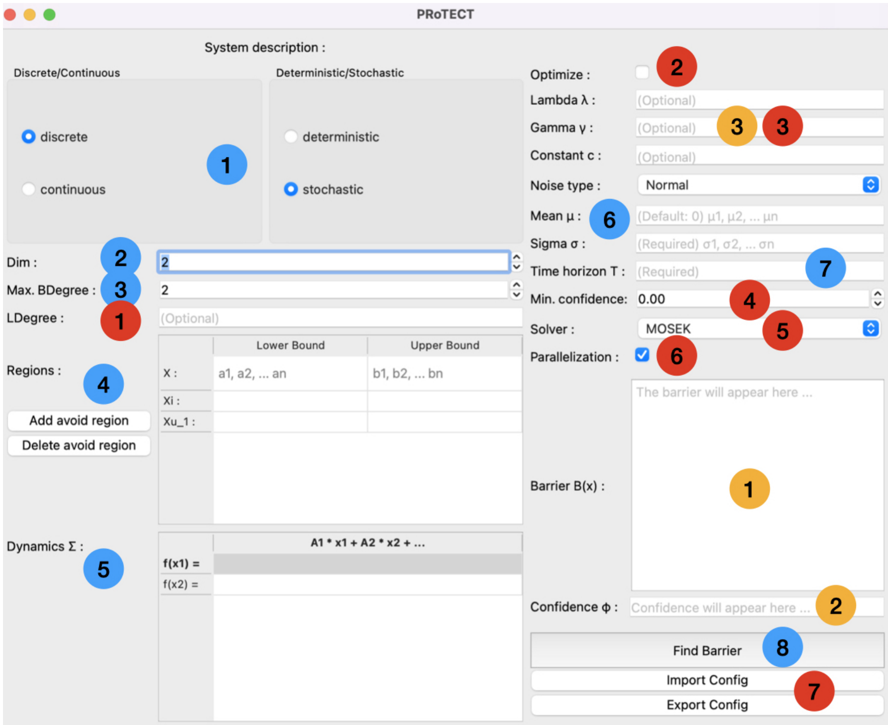

FIGURE 1. PRoTECT GUI for dt-SS, where required parameters, optional parameters, and outputs are marked with blue, red, and yellow circles, respectively.

and *Export Config* buttons (red-7), with all the examples from Table 1 and Table 2 of the paper available in the folder `/ex/GUI_config_files`.

By navigating to the `PRoTECT` folder in a terminal and running:

```
python3 main.py
```

you can open the PRoTECT GUI, to then run examples. We have provided extensive *Youtube tutorial videos* to help with this for the four classes of dynamical systems, as well as the importing and exporting of the config files: here.

5.2. **Treating PRoTECT as an API.** In addition to the GUI, as already seen in the artifact evaluation part of this document, we provide python scripts which can be used to call the PRoTECT functions as an API. We will now describe for each of the four dynamical systems how to use these functions as an API, in addition we have a *Youtube tutorial video* which explains the python scripts used by PRoTECTand explains how to edit them: here.

5.2.1. *Discrete-Time Stochastic Systems (dt-SS).* In general, the backend of PRoTECT behaves as an API, with functions that can be called and used in any python program. We provide some generic configuration files in /ex/benchmarks-stochastic and /ex/benchmarks-deterministic, which demonstrate how to use the functions in a standard python program. The user is expected to provide the following *required* parameters: dimension of the state set $X \subseteq \mathbb{R}^n$ (blue-2), indicated by dim, and the degree of the barrier certificate (blue-3), denoted by b_degree. The lower and upper bounds of the initial region $X_{\mathcal{I}}$, labeled as L_initial and U_initial; lower and upper bounds of the unsafe region $X_{\mathcal{U}}$, referred to as L_unsafe and U_unsafe; lower and upper bounds for the state set X, denoted as L_space and U_space; where the value of each dimension is separated with a comma (blue-4). Due to possible scenarios with multiple unsafe regions, the unsafe region is passed to the functions as a numpy array of numpy arrays describing each individual unsafe region. The transition map f, represented by f, written as a SymPy expression[1] for each dimension using states x1,x2,... and noise parameters varsigma1,varsigma2,... (blue-5). The time horizon $\mathcal{T}$, noted as t (blue-7). The distribution of the noise, NoiseType, can be specified as either "normal", "exponential", or "uniform" (blue-6).

Users may also specify *optional* parameters, with default values provided in Listing 1. These include the degree of the Lagrangian multipliers $l_i(x), l_u(x), l(x)$: l_degree (red-1), which, if not specified (i.e., set to None), will default to the same value as b_degree; the type of solver: solver (red-5), that can be either set to "mosek" or "cvxopt". The confidence level ϕ (in equation (5) of the paper) can be optimized using optimize (red-2), if set to True. In this case, due to having a bilinearity between γ and λ (in equation (5) of the paper), the user is required to provide one λ: lam, *e.g.,* select $\lambda = 1$ (red-3). The tool will then optimize for the other decision variables including γ and c to provide the highest confidence level ϕ. Alternatively, the user can select a minimum confidence level ϕ (red-4) using confidence they desire, so that PRoTECT attempts to search for a barrier certificate satisfying that confidence level. The parameters for the distributions should be specified as follows (blue-6): for normal distributions, the mean μ can be set using mean, and the diagonal covariance matrix σ can be provided using sigma. For exponential distributions, the rate parameter for each dimension can be set using rate. For uniform distributions, the boundaries for each dimension can be set using a and b. We provide two functions for dt-SS (red-6): the first dt_SS finds a barrier for a single degree, and the second parallel_dt_SS runs the first function in parallel for all barrier degrees up to the maximum barrier degree specified (also called b_degree).

```
dt_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
    varsigma, f, t, l_degree=None, NoiseType="normal", optimize=False, solver="mosek",
    confidence=None, gam=None, lam=None, c_val=None, mean=None, sigma=None, rate=None, a=None
    , b=None)
```

[1]https://docs.sympy.org/latest/tutorials/intro-tutorial/basic_operations.html

```
parallel_dt_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
     varsigma, f, t, l_degree=None, NoiseType="normal", optimize=False, solver="mosek",
    confidence=None, gam=None, lam=None, c_val=None, mean=None, sigma=None, rate=None, a=None
    , b=None)
```

Listing 1. dt-SS functions.

5.2.2. *Discrete-Time Deterministic System (dt-DS).* The user is required to input necessary (and optional) parameters as outlined in Subsection 5.2.1, excluding those parameters relevant to stochasticity (*e.g.,* time horizon, constant c, noise distribution, and confidence level). Optionally, the user can specify the level sets γ using `gam` or λ using `lam`. It is important to note that optimization for the level sets γ and λ is not performed, as any feasible solution with $\lambda > \gamma$ ensures a safety guarantee over an infinite time horizon. Similarly, we provide two functions `dt_DS` and `parallel_dt_DS` for the serial and parallel execution.

```
dt_DS(b_degree,dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x, f,
    l_degree=None, solver="mosek",gam=None,lam=None)
parallel_dt_DS(b_degree,dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
    f, l_degree=None, solver="mosek",gam=None,lam=None)
```

Listing 2. dt-DS functions.

5.2.3. *Continuous-Time Stochastic System (ct-SS).* The user is asked to enter necessary (and optional) parameters as detailed in Subsection 5.2.1. Additionally, via the corresponding GUI, users must provide the diffusion term δ using `delta` for Brownian motion, the reset term ρ using `rho` for Poisson process, and the Poisson process rate ω using `p_rate`. For cases lacking either Brownian motion or Poisson processes, the corresponding parameter should be set to zero. The confidence level ϕ can also be optimized if `optimize` is set to `True`. We provide functions `ct_SS` and `parallel_ct_SS` for the serial and parallel execution.

```
ct_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x, f, t,
    l_degree=None, delta=None, rho=None, p_rate=None, optimize=False, solver="mosek",
    confidence=None, gam=None, lam=None, c_val=None)
parallel_ct_SS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
     f, t, l_degree=None, delta=None, rho=None, p_rate=None, optimize=False, solver="mosek",
    confidence=None, gam=None, lam=None, c_val=None)
```

Listing 3. ct-SS functions.

5.2.4. *Continuous-Time Deterministic System (ct-DS).* The user is expected to input necessary (and optional) parameters as detailed in Subsection 5.2.1, omitting parameters pertinent to stochasticity (*e.g.,* time horizon, constant c, and confidence level). Optionally, users can define the level sets γ using `gam` or λ using `lam`. Optimization for level sets γ and λ is not conducted, *i.e.,* any feasible solution where $\lambda > \gamma$ ensures a safety guarantee over an infinite time horizon. Functions `ct_DS` and `parallel_ct_DS` are employed for the serial and parallel execution.

```
ct_DS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x, f,
   l_degree=None, solver="mosek", gam=None, lam=None)
parallel_ct_DS(b_degree, dim, L_initial, U_initial, L_unsafe, U_unsafe, L_space, U_space, x,
    f, l_degree=None, solver="mosek", gam=None, lam=None)
```

Listing 4. ct-DS functions.

6. Manual Installation

We have made every effort to ensure the following instructions are comprehensive for the VM under consideration. Additionally, a YouTube video is available here, which provides a general walk-through of the installation process. As we provide installation details for FOSSIL in this artifact evaluation, along with a few specific adjustments for this VM, the YouTube tutorial should be considered a supplement to the following instructions, not a direct substitute.

The username of the VM is `artifact` and the password (required for sudo commands) is also `artifact`.

6.1. **Install PRoTECT.** You download the repository from Zenodo and save it in the home directory; after unzipping the directory change the folder name to PRoTECT to help with the later steps. Alternatively, you can clone Github Release v1.3 which is connected to this Zenodo DOI via the following steps. The first step is to navigate to your home directory (the location that contains the folder `Documents`) and open a terminal (right click then `Open in Terminal`), or open a terminal and run the command

`cd ~`

to navigate there.

Install git with the following:

`sudo apt-get install git`

Clone the repository for the tool PRoTECT, then go into the folder and install the required dependencies via:

```
git clone https://github.com/Kiguli/PRoTECT --branch v1.3
cd PRoTECT
pip install -r requirements.txt
```

For the GUI to run correctly, this specific VM seems to also require the installation of `libxcb-cursor0` (often it is already installed, such as in the VM used in the Youtube tutorial video):

```
sudo apt-get install libxcb-cursor0
```

You can test this part of the installation from the `PRoTECT` directory; the GUI should load with no issues by running:

```
python3 main.py
```

6.2. **Get Mosek license.** For the results in Tables 1 and 2, the MOSEK solver was used. Therefore, we will now install the MOSEK license (a free trial is available, and it is free for academic users).

Fill in your details to get a Mosek license at https://www.mosek.com/license/request/?i=acp. The license file will be emailed to you with instructions of where to place the file in your home directory. On the VM, this involves creating a `mosek` folder in the home directory and placing the license file, `mosek.lic`, into this folder.

6.3. **Installing FOSSIL.** To install FOSSIL, navigate to the PRoTECT folder and move the folder "fossil-main" to the home directory, using:

```
cd ~/PRoTECT
mv fossil-main ..
```

Or alternatively navigate the terminal back to the home directory with `cd ~`. Then clone the FOSSIL repository (specifically release 2.0) from Github using the following commands:

```
git clone https://github.com/oxford-oxcav/fossil --branch 2.0
sudo apt-get install -y python3 python3-pip curl
```

After using either of the two methods above, install the dependencies by running:

```
curl -fsSL https://raw.githubusercontent.com/dreal/dreal4/master/setup/ubuntu/20.04/install.sh
| sudo bash
cd fossil
pip3 install .
```

Note: *There is a single command starting with* `curl -fsSL` *and ending with* `bash`*, which should remain as one continuous command and not be split across multiple lines, as this document has auto-formatted it. Additionally, the dot in the final command is crucial, as it indicates the current directory.*

6.4. **Copying PRoTECT models into FOSSIL.** To run the provided FOSSIL versions of our examples from Table 1, we need to copy the used models from PRoTECT into FOSSIL. You can append (concatenate) this file directly to the end of the other file using the following command (all one line):

```
cat /PRoTECT/ex/benchmarks-deterministic/FOSSIL-versions/models.py >>
/FOSSIL/experiments/benchmarks/models.py
```

You can also do it *without* the terminal. First navigate inside of the `PRoTECT` folder to the path:

`~/PRoTECT/ex/benchmarks-deterministic/FOSSIL-versions/`

Open the file `models.py` and copy all of the content from this file. Then navigate to the FOSSIL tool (the `fossil` folder) to the path:

`~/fossil/experiments/benchmarks/`

open the file `models.py` and paste the copied code at the end of the file, then save and exit.

6.5. **Setup PYTHONPATH.** The final task in the setup and installation of this artifact evaluation is to add the paths of both PRoTECT and FOSSIL to the PYTHONPATH so that the python scripts we will call can find the functions they require from the PYTHONPATH. Again this section can be completed without needing the terminal.

Go to the home directory and open the hidden file `.profile`. Assuming you are using the VM mentioned, and that both the folders `PRoTECT` and `fossil` are in the home directory, add the following line to the end of the `.profile` file:

```
export PYTHONPATH=$PYTHONPATH:/home/artifact/PRoTECT:/home/artifact/fossil
```

This adds the location of both repositories we have downloaded to the PYTHONPATH. Once this is done, save and exit the file and then restart the VM to enable the PYTHONPATH to be permanently updated on the PC.

If not already visible, you can see hidden files in the file manager by going to the menu and ticking the box "Show Hidden Files".