TLA$^+$ for All: Model Checking in a Python Notebook

Proposal for a short presentation at the 2025 TLA$^+$ Community Event

Submission Categories: Innovative use of existing tools; use of TLA$^+$ in education

K. Läufer and G. K. Thiruvathukal, Loyola U. Chicago, {laufer,gkt}@cs.luc.edu

Motivation TLA$^+$ is widely recognized for its effectiveness in specifying and verifying concurrent and distributed systems. However, for educators and practitioners, barriers to adoption include installation complexity and tooling setup. In the proposed presentation, we demonstrate a lightweight, easily shareable, and fully reproducible approach to running TLA$^+$ in a *Python notebook hosted on Google Colab* without requiring new tools or custom Jupyter kernel development.[1] By creating an environment where attendees can *experiment with TLA$^+$ models instantly during the presentation*, we lower these barriers and demonstrate the suitability for education and outreach.

Background and Related Work A presentation at the 2021 TLA+ conference called for greater support for the interactive use of TLA$^+$ [1]. While focusing on building advanced tools for better tracing, visualization, and other features for understanding complex models, this presentation recognizes the importance of interactivity and user experience, including a REPL (read-eval-print loop) for evaluating portions of a model.

Furthermore, there have been specific efforts to integrate TLA$^+$ with interactive computing environments, particularly Jupyter notebooks, though recent development has been limited or possibly abandoned. [2], [3]. More recently, tla-web/TLAJS has resulted in an interactive TLA$^+$ interpreter running in a web browser [4]; the author has acknowledged the challenges of achieving feature parity with the TLC model checker.

Pedagogical Goals In our upper-division undergraduate/first-year graduate course[2] on *Formal Methods in Software Engineering*, we dedicate approximately *40% of the course* to TLA$^+$. We are offering this course for the third time this spring 2025 semester and have shared our preliminary experiences with the community [5], [6].

By using Python notebooks, we can lower the barrier to engaging and experimenting with TLA$^+$; specific goals include:

- *Easy to share and reproduce:* Students can run the notebook in a web browser without installing anything. This removes a common friction point and makes the examples accessible to a wide audience.
- *Encouraging interactive experimentation:* By hosting models on GitHub, students can modify and test different configurations quickly.
- *Supports teaching/training formal methods to a wider audience:* Easy sharing and zero installation makes it easier to teach formal methods examples to a wider audience.
- *Bridging formal methods with software engineering:* By integrating model checking in a widely used environment (Python notebooks), students see TLA$^+$ in a familiar setting.
- *Low cost and easy to maintain:* We can invoke the required TLA$^+$ command-line tools within a notebook, with minimal Python scripting for additional enhancements.

Method Our setup consists of these steps within a standard, publicly available Python notebook [7].

1. *Dynamically install* the TLA$^+$ Toolbox command-line tools and its dependencies, such as a Java SDK.
2. *Download the TLA$^+$ model to be analyzed* from a GitHub repository.
3. *Perform model checking* with tools such as *TLC* or alternative checkers like *Apalache.*
4. *Display results* within the notebook, leveraging standard command-line utilities (see Figure 1).
5. *Evaluate constant expressions* in the TLA$^+$

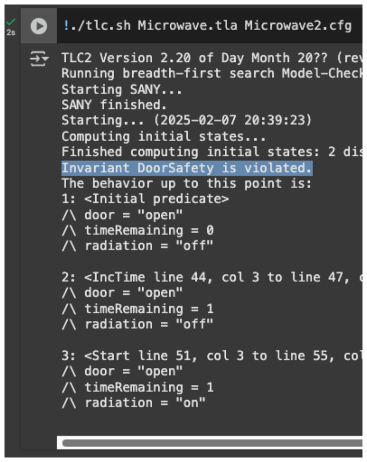
[1]Google Colab uses the same foundation as Jupyter notebooks and JupyterHub. Our demo will show that Colab is not required for those wanting to work on their own resources.

[2]In addition, we hope to introduce formal methods to first and second year students. Notebooks already have strong traction in introductory Python-based courses.

REPL (see Figure 2).

Unlike traditional cloud-based IDEs (e.g., *VS Code with the TLA$^+$ extension*), our approach offers immediate execution with no additional setup, making it ideal for *presentations, group activities, and self-paced learning.*

In addition, the notebook approach makes it very easy to gain a detailed understanding of the required steps, supports collaborative work, and allows for additional customization.



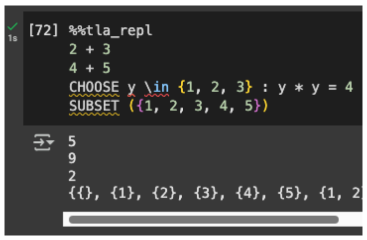
```
!./tlc.sh Microwave.tla Microwave2.cfg

TLC2 Version 2.20 of Day Month 20?? (rev
Running breadth-first search Model-Check
Starting SANY...
SANY finished.
Starting... (2025-02-07 20:39:23)
Computing initial states...
Finished computing initial states: 2 dis
Invariant DoorSafety is violated.
The behavior up to this point is:
1: <Initial predicate>
/\ door = "open"
/\ timeRemaining = 0
/\ radiation = "off"

2: <IncTime line 44, col 3 to line 47, c
/\ door = "open"
/\ timeRemaining = 1
/\ radiation = "off"

3: <Start line 51, col 3 to line 55, col
/\ door = "open"
/\ timeRemaining = 1
/\ radiation = "on"
```

Figure 1: TLC output showing a violation of a safety invariant. A simple filter makes the output concise and readable without requiring tool-specific plugins.



```
[72] %%tla_repl
     2 + 3
     4 + 5
     CHOOSE y \in {1, 2, 3} : y * y = 4
     SUBSET ({1, 2, 3, 4, 5})

     5
     9
     2
     {{}, {1}, {2}, {3}, {4}, {5}, {1, 2
```

Figure 2: By defining a custom directive (about 30 lines of Python), we can evaluate TLA$^+$ constant expressions in a Python notebook cell, again with no tool-specific plugins required.

Evaluation One might argue that notebooks are not ideal for large scale software development. However, when it comes to model checking, notebooks can be a useful first step toward developing a complex application that will later continue outside of a notebook environment. We believe that simpler, early-stage models can be ideal candidates for being developed within a notebook. We also show how to maintain source artifacts in an external repository, thereby supporting workflows based on full, non-notebook TLA$^+$ environments, such as the TLA$^+$ Toolbox or the VSCode extension.

In addition, we are planning to deploy this notebook-based approach in our *formal methods course* this semester and evaluate it with respect to these attributes: *a) Student engagement:* Students are more willing to explore TLA$^+$ by running models instantly. *b) Reproducibility:* Every student, regardless of local development setup, sees identical results. *c) Ease of maintenance:* We avoid the need for a custom Jupyter kernel, reducing long-term maintenance costs.

Future Work Several enhancements are planned for further improving this workflow: *a)* Support for pretty-printing of the TLA$^+$ code using `tlatex`; this works but currently requires a full LaTeX environment. *b) State graph visualization* using *Graphviz* to make TLA$^+$ execution results more interpretable. *c) Broader tool integration*, such as Alloy, to extend the notebook-based approach to other formal verification environments.

Conclusion By running *TLA$^+$ in a Python notebook*, we offer a practical, *zero-install, reproducible*, and *maintainable* solution for *teaching and experimentation.* This approach reduces barriers to entry, removes friction in learning TLA$^+$, and serves as a practical and maintainable tool for education and research. Our *workshop presentation could be delivered from the notebook itself*, reinforcing its effectiveness as a lightweight alternative to traditional IDEs and custom Jupyter kernels.

We look forward to discussing our approach with the audience and obtaining valuable feedback.

References

[1] A. J. J. Davis and S. Lanka, *Current and future tools for interactive TLA+*, Presented at the TLA+ Conf. 2021, Available at `https://conf.tlapl.us/2021/JesseSamy-talk.pdf`, 2021.

[2] *PlusPy*, Available at `https://github.com/tlaplus/PlusPy`.

[3] *Tlaplus_jupyter*, Available at `https://github.com/kelvich/tlaplus_jupypter`.

[4] W. Schultz, *Towards better interactive formal specifications*, Presented at the TLA+ Conf. 2024, Available at `https://conf.tlapl.us/2024/WillSchultz-TowardsBetterInteractiveFormalSpecifications.pdf`, 2024.

[5] K. Läufer, G. Mertin, and G. K. Thiruvathukal, *Wip: An engaging undergraduate intro to model checking in software engineering using TLA+*, Available at `https://arxiv.org/abs/2407.21152`, 2024.

[6] K. Läufer, G. Mertin, and G. K. Thiruvathukal, "Engaging more students in formal methods education: A practical approach using temporal logic of actions," *Computer*, vol. 57, no. 12, pp. 118–123, 2024, Available at `https://www.computer.org/csdl/magazine/co/2024/12/10754605`.

[7] K. Läufer, G. Mertin, and G. K. Thiruvathukal, *TLA+ Model Checking in Colab: Microwave Oven (WIP)*, Available at `https://figshare.com/articles/software/TLA_Model_Checking_in_Colab_Microwave_Oven_WIP_/27122916`, Sep. 2024.