POMDPStressTesting.jl: Adaptive stress testing for black-box systems

Robert J. Moss[1]

1 Stanford University

Summary

POMDPStressTesting.jl is a package that uses reinforcement learning and stochastic optimization to find likely failures in black-box systems through a technique called adaptive stress testing (Lee, Mengshoel, & Kochenderfer, 2019). Adaptive stress testing (AST) has been used to find failures in safety-critical systems such as aircraft collision avoidance (Lee, Kochenderfer, Mengshoel, Brat, & Owen, 2015), flight management systems (Moss et al., 2020), and autonomous vehicles (Koren, Alsaif, Lee, & Kochenderfer, 2018). The POMDPStressTesting.jl package is written in Julia (Bezanson, Edelman, Karpinski, & Shah, 2017) and is part of the wider POMDPs.jl ecosystem (Egorov et al., 2017). Fitting into the POMDPs.jl ecosystem, our package has access to simulation tools, policies, visualizations, and most importantly solvers. We provide several different solver variants including reinforcement learning solvers such as Monte Carlo tree search (Coulom, 2006) and deep reinforcement learning solvers such as trust region policy optimization (TRPO) (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015) and proximal policy optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). We also include stochastic optimization solvers such as the cross-entropy method (Rubinstein, 1999) and include random search as a baseline. Additional solvers can easily be added by adhering to the POMDPs.jl interface.

The AST formulation treats the falsification problem (i.e. finding failures) as a Markov decision process with a reward function that uses a notion of distance to a failure event to guide the search towards failure. The reward function also uses the state-transition probabilities to guide towards *likely* failures. Recall that reinforcement learning aims to maximize the discounted sum of expected rewards, therefore using the log-likelihood allows us to maximize the summations, which is equivalent to maximizing the product of the likelihoods. A gray-box simulation environment steps the simulation and outputs the state-transition probabilities, and the black-box system under test is evaluated in the simulator and outputs an event indication and the real-valued distance metric. To apply AST to a general black-box system, a user has to implement the following interface:

```julia
# GrayBox simulator and environment
abstract type GrayBox.Simulation end
function GrayBox.environment(sim::Simulation)::GrayBox.Environment end
function GrayBox.transition!(sim::Simulation)::Real end

# BlackBox.interface(input::InputType)::OutputType
function BlackBox.initialize!(sim::Simulation)::Nothing end
function BlackBox.evaluate!(sim::Simulation)::Tuple{Real, Real, Bool} end
    function BlackBox.distance!(sim::Simulation)::Real end
    function BlackBox.isevent!(sim::Simulation)::Bool end
function BlackBox.isterminal!(sim::Simulation)::Bool end
```

The simulator stores simulation-specific parameters and the environment stores a collection of probability distributions that define the state-transitions (e.g., Gaussian noise models, uniform control inputs, etc.). Two types of AST action modes are provided to the user: random seed actions or directly sampled actions. The seed-action approach is useful when the user does not have direct access to the environmental distributions or when the environment is complex. In this case, if the state-transition probability is inaccessible it may be set to 0, thus guiding the search solely by the distance metric. When using directly sampled actions, the TRANSITION and EVALUATE functions can take in an environment sample selected by the solvers and apply it directly as input to the black-box system, allowing for finer control over the search.

As an example, the functions in the above interface can either be implemented directly in Julia or can call out to C++, Python, MATLAB® or run a command line executable. We provide a benchmark example often used in the falsification literature (Hoxha, Abbas, & Fainekos, 2015) which uses a Simulink® automatic transmission model as the black-box system and selects throttle and brake control inputs as part of the environment. Typically, implementing the DISTANCE and ISEVENT functions rely solely on the output of the black-box system under test, thus keeping in accordance with the black-box formulation.

Our package builds off work originally done in the AdaptiveStressTesting.jl[1] package, but POMDPStressTesting.jl adheres to the interface defined by the POMDPs.jl package and provides different action modes and solver types. Related falsification tools (i.e. tools that do not include most-likely failure analysis) are S-TALIRO (Annapureddy, Liu, Fainekos, & Sankaranarayanan, 2011), Breach (Donzé, 2010), RRT-REX (Dreossi et al., 2015), and FALSTAR (Zhang, Ernst, Sedwards, Arcaini, & Hasuo, 2018). These packages use a combination of optimization, path planning, and reinforcement learning techniques to solve the falsification problem. The tool closely related to POMDPStressTesting.jl is the AST Toolbox in Python (Koren et al., 2018), which wraps around the gym reinforcement learning environment (Brockman et al., 2016). The author has contributed to the AST Toolbox and found the need to create a similar package in Julia for better performance and to interface with the POMDPs.jl ecosystem in pure Julia.

Research and Industrial Usage

POMDPStressTesting.jl has been used to find likely failures in aircraft trajectory prediction systems (Moss et al., 2020), which are flight-critical subsystems used to aid in-flight automation. A developmental commercial flight management system was stress tested so the system engineers could mitigate potential issues before system deployment. In addition to traditional requirements-based testing for avionics certification (RTCA, 2011), this work is being used to find potential problems during development. Other ongoing research is using POMDPStressTesting.jl for assessing the risk of autonomous vehicles.

Acknowledgments

---

[1] https://github.com/sisl/AdaptiveStressTesting.jl

References

Annapureddy, Y., Liu, C., Fainekos, G., & Sankaranarayanan, S. (2011). S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *TACAS* (pp. 254–257). Springer.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI gym. *arXiv:1606.01540.*

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games* (pp. 72–83). Springer.

Donzé, A. (2010). Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification* (pp. 167–170). Springer.

Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., & Deshmukh, J. V. (2015). Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods Symposium (NFM)* (pp. 127–142). Springer.

Egorov, M., Sunberg, Z. N., Balaban, E., Wheeler, T. A., Gupta, J. K., & Kochenderfer, M. J. (2017). POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty. *Journal of Machine Learning Research*, *18*(26), 1–5.

Hoxha, B., Abbas, H., & Fainekos, G. (2015). Benchmarks for Temporal Logic Requirements for Automotive Systems. In *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, EPiC Series in Computing (Vol. 34, pp. 25–30). EasyChair.

Koren, M., Alsaif, S., Lee, R., & Kochenderfer, M. J. (2018). Adaptive Stress Testing for Autonomous Vehicles. In *IEEE Intelligent Vehicles Symposium (IV)* (pp. 1–7).

Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., & Owen, M. P. (2015). Adaptive stress testing of airborne collision avoidance systems. In *IEEE/AIAA Digital Avionics Systems Conference (DASC)*.

Lee, R., Mengshoel, O. J., & Kochenderfer, M. J. (2019). Adaptive stress testing of safety-critical systems. In *Safe, Autonomous and Intelligent Vehicles* (pp. 77–95). Springer.

Moss, R. J., Lee, R., Visser, N., Hochwarth, J., Lopez, J. G., & Kochenderfer, M. J. (2020). Adaptive stress testing of trajectory predictions in flight management systems. In *IEEE/AIAA Digital Avionics Systems Conference (DASC)*.

RTCA. (2011, December). Software Considerations in Airborne Systems and Equipment Certification. DO-178C.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, *1*(2), 127–190.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning* (pp. 1889–1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv:1707.06347.*

Zhang, Z., Ernst, G., Sedwards, S., Arcaini, P., & Hasuo, I. (2018). Two-layered falsification of hybrid systems guided by Monte Carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *37*(11), 2894–2905.