Debugging a Policy: A Framework for Automatic Action Policy Testing
Marcel Steinmetz, Timo P. Gros, Philippe Heim, Daniel Höller, Jörg Hoffmann
Saarland University, Saarland Informatics Campus
Saarbrücken, Germany
{steinmetz, timopgros, hoeller, hoffmann}@cs.uni-saarland.de, s8phheim@stud.uni-saarland.de

The Assumptions
- Deterministic action policies \( \pi \) mapping states \( s \) to actions \( \pi(s) \).
- Policy value function \( V^\pi \) mapping every state \( s \) to the objective value achieved by \( \pi \) on \( s \).
- For qualitative objectives:
  - \( V^\pi(s) = 0 \) if \( \pi \) is bug-free and \( \pi \) is optimal.
  - \( V^\pi(s) = 0.5 \) if \( \pi \) is bug-free but optimally suboptimal.
  - \( V^\pi(s) = 1 \) if \( \pi \) is suboptimal.
- Optimal value function \( V^* \) mapping every state \( s \) to best value any policy can achieve on \( s \).
- Generic better than notion: \( V(s) < V'(s) \) iff \( V'(s) \) objective is minimization
  \( V(s) > V'(s) \) objective is maximization

Policy Bugs: Definition
Definition (Bug)
A state \( s \) is a bug in \( \pi \) if
\[ \Delta := |V^\pi(s) - V^\pi(s)| > 0 \]
\( \Rightarrow \) \( s \) is bug-free if \( \pi \) is optimal
Note: \( \pi \) can have bugs despite each \( \pi(s) \) being optimal individually. Consider \( \pi(s_1) = \pi(s_2) = a \) with the (qualitative) objective to reach \( g \).

But, how can we actually
\( \Rightarrow \) confirm whether a given state \( s \) is a bug?
\( \Rightarrow \) generate suitable candidates for testing?

Bug Confirmation: The Easy Cases
- \( V^* \) can be computed for all states offline, before starting testing / deploying the policy.
- Domain-specific knowledge: user provides \( V^* \) for certain states as input, can use these states for testing.
- Specifically design test-case generation methods, generating only states \( s \) with a given \( V^*(s) \) value.
- Example: in qualitative setting, generate states via backward samples from the goal.
- \( V^*(s) \) may be efficiently computable for certain states \( s \), e.g., due to structural properties of the domain, or if a fixed-depth lookahead search is sufficient.

Bug Confirmation: The General Case
If \( V^* \) cannot be computed exactly, then approximate!
- Optimistic approximation of \( V^* \): \( h_\delta(s) \leq V^*(s) \)
- Pessimistic approximation of \( V^* \): \( H_\delta(s) \geq V^*(s) \)

Proposition 4 (Bug Confirmation):
If (i) \( h_\delta(s) \geq V^*(s) \) and (ii) \( H_\delta(s) \leq V^*(s) \), then
\[ |h_\delta(s) - H_\delta(s)| \leq |V^*(s) - V^*(s)|. \]

Bug Confirmation: Comparison & Takeaway
- All boil down to:
  - Approximate \( V^*(s) \) up to a high degree of precision.
  - Then try to find a better policy for \( s \).

Conclusion & Outlook
Bug confirmation:
- Many special cases where \( V^*(s) \) feasible to compute.
- If not, fall back to well-known policy-improvement algorithms.
Let’s debug your policies!
- Develop fuzzing methods (start states, fuzzing operators, biases, termination, ...)
- Adapt other techniques from software testing (e.g., metamorphic testing).
- See what it does in your favorite planning & learning scenarios.