Efficient state mergin in symbolic execution

- Program analysis technique that can explore multiple path at the same time
- Using symbolic inputs
  A use case is a test input for a certain path for unit tests. You can achieve a higher code coverage then normal.

Introduction

- Take a compiled program
- Replace concrete inputs with symbolic values
- Execute the program, maintain symbolic state
- Build path conditions for each execution path
- Use constraint solver path feasibility
- External libraries an be called
  - Can build concrete arguments for calls
  Trade-offs
- scalability is a big issue.
- N of path is exponantial
  You can make it more efficient by merging if-statements. Unfortunately it increases symbolic expressions describing variables. The total cost of merging x depends on future branch conditions and array indices.
  Query count estimation
- Proprocess program with lightweight static analysis
- identify use-counts of variables from each location ℓ.
  - Mark variables as "hot" if likely to cause many queries if made symbolic
  QCE
- Statically approx future variable appearances after potential merge.
- Only merge if count below current threshold α.
- Use this heuristic to assess if merging would net a benefit
- impacts only completion time, not soundness or completeness
  Dynamic state merging
- Efficiently combine
  - State merging
  - Search strategies
- Dynammically identify merging opportunities
- Insentive to search strategy
  - without restrictions
  DSM
  When picking the next state in pick state, we check if some state α_1 is similar to a

predecessor a_2 of another state a_2 in the work list. Build a forwarding list. If similar fast-forward to a_1. Until no longer similar or caught up. It only prioritizes some states. If not similar, return control to search strategy.

Experimental evaluation

The experimental values are up to exponential speedup. They ran experiments on GNU Coreutils with programs for file processing and terminal control. He showed a graph of the results. The Path ratio decreases when the tool index becomes greater. When merging there is exponential improvement.

Conclusion

- Merging reduces amount of states
  - Increases burden
- QCE and DSM combined proven beneficial