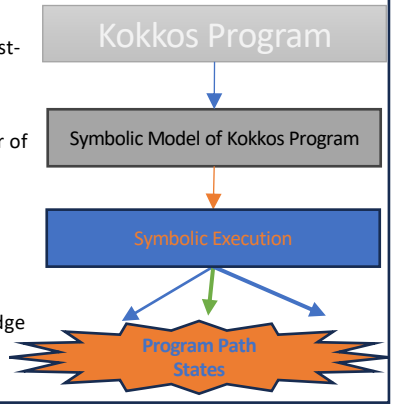
Ground Truth for Guided Symbolic Execution of Kokkos Parallel Programs

Vivek Kale[1], Keita Teranishi[2], Shyamali Mukherjee[1], Richard Rutledge[1], Sam Pollard[1], Jackson Mayo[1]

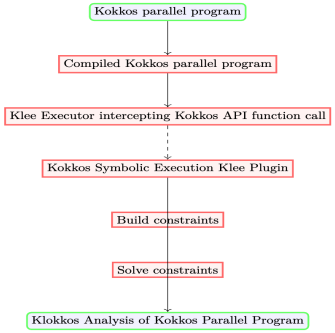
Sandia National Laboratories[1], Oak Ridge National Laboratory[2]

Testing a Kokkos Parallel Program is Hard

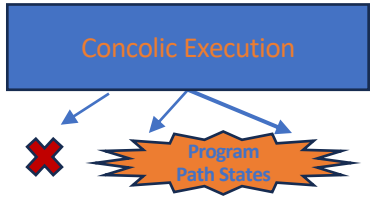
- **Background**: Kokkos is single model sitting on top of host-device backend, e.g., OpenMP-CUDA
- **Problem:** Once portability is introduced, more prone to bugs (number of bugs grows exponentially with number of combinations of backends) than bare OpenMP-CUDA
- **Fix:** Symbolic execution to aid programmer to identify bugs
- **Still a Problem**: program path state explosion
- **Research Question:** prune state explosion with knowledge of scientific and parallel computing?



LLVM Klee's Concolic Execution of Kokkos Program

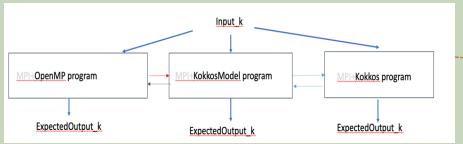


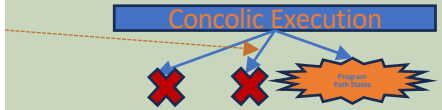
- Symbolic execution with common inputs for program --> LLVM Klee
- Challenge: Still very large number test cases in parallel program
- How do we further prune the number of cases?



Database of Parallel Programs → Prune Even More

1. Identify examples based on use cases, computational patterns, kind of bug
2. Classify or tag real applications associated with these
3. Guided symbolic analysis queries mistake examples during symbolic execution to hone in on error in Kokkos program in question
   - Via LLVM clang AST transform
4. Machine learning to develop rules for correctness for future analysis





Auto-Testing Science Simulations

1. Use case: From Kokkos LAMMPS → Concolic Analysis: Consider the input of 1 million particles are distributed across 8 threads, distance at most 1 nn
2. By providing heterogeneous bug in the stencil code, formal model can identify or specify rule for bug.
3. Further guiding by examples: (1) check equivalence with buggy proxy stencil (2) If not equivalent, generate paths of proxy stencil given input and check for data race.
4. Augment formal model to specify rule for bug.
5. Store rules in database for future analysis

Future work: Automatic synthesis of Kokkos mistake examples via LLVM AST translation

References:

[1] K. Teranishi, S. Mukherjee, S. Pollard, R. Rutlege, A. Orso, V. Sarkar. *Towards Automated Test Synthesis of Performance Portable Programs.* Klee 2022 workshop. London, United Kingdom. September 2022.

[2].*LLVM Project.* https://llvm.org

SAND2023-09183C