Graph-Based Verification Patterns

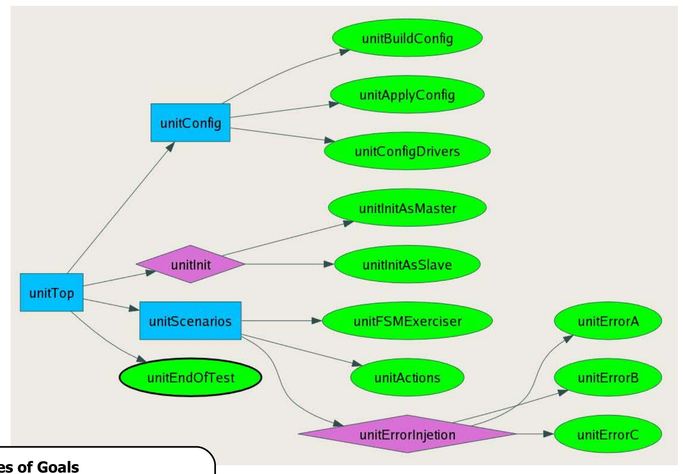
Jörg Behrend, Gero Dittmann, Klaus Keuerleber
IBM STG, Research and Development
Böblingen, Germany
jbehrend@de.ibm.com
ged@zurich.ibm.com
klaus_keuerleber@de.ibm.com

Jörg Grosse, Frederic Krampac
Breker Verification Systems
San Jose, USA
joerg@brekersystems.com
fred@brekersystems.com

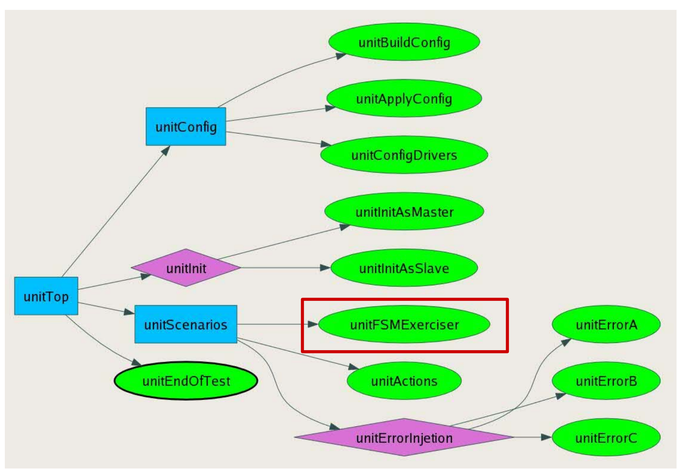
Abstract:

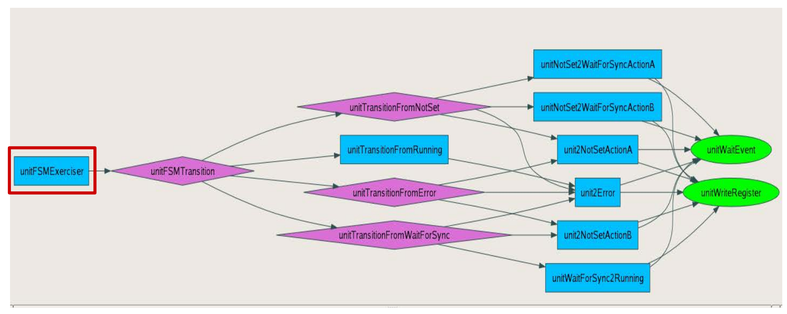
We provide an overview of graph-based verification patterns used in recent projects at IBM. We present a basic graph structure and show how to randomly traverse a finite state machine. We proceed to give an example of graph-based error injection. We extend this approach to end-to-end checking and error recovery. Finally, we show how to use reachability analysis to sanity-check testcases, improving the debuggability of complex scenarios

Basic Graph Structure

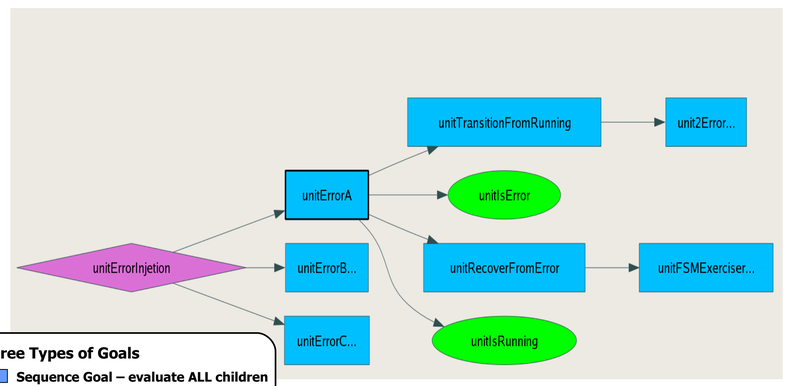


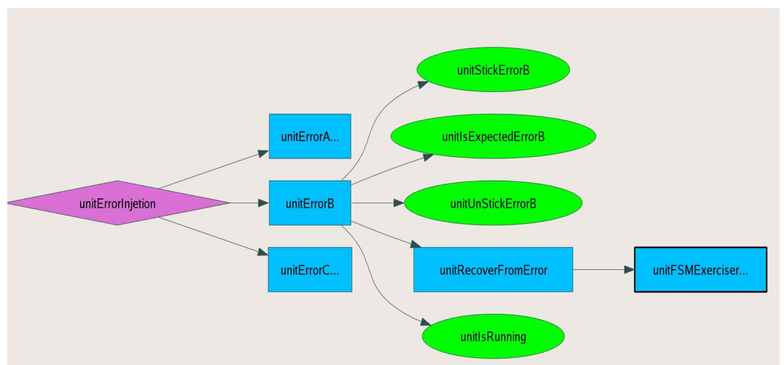
Exercising a Finite State Machine



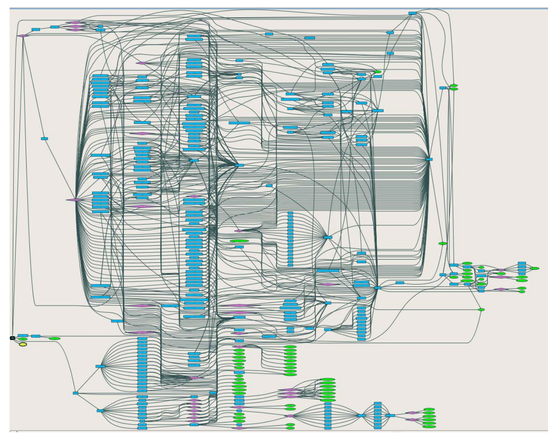


Error Injection, Recovery and End-to-End Checking

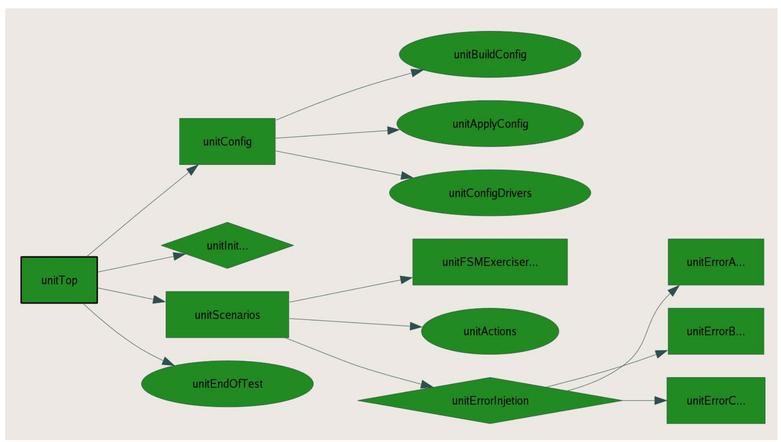


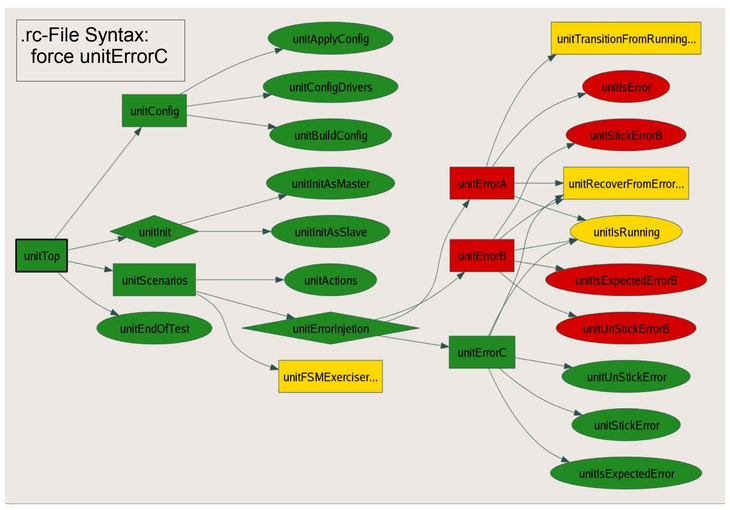


Example: Complete Unit Graph



Sanity Checking: Reachability View





- We have presented the following basic graph-based verification patterns:
  - How to structure a basic unit graph.
  - How to randomly exercise an FSM.
- We have also shown examples of error injection scenarios:
  - How to structure a scenario sub-graph.
  - How to recover from an error.
  - How to implement an end-to-end check using graph-based verification.
  - How to sanity-check testcases for complex environments using reachability analysis.
  - All examples run on the TrekSoC™ product from Breker Verification Systems.