WALKING THE GRAPH

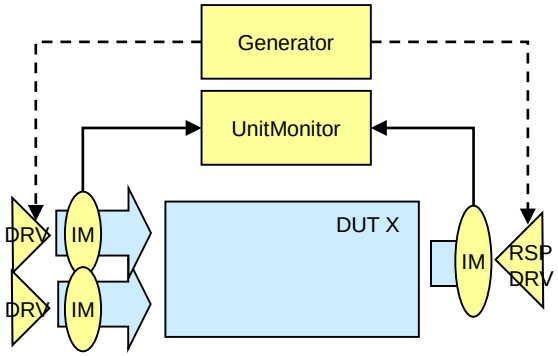
A holistic approach to graph-based verification for logic with sparse state space

Sandeep Korrapati	IBM – sakorrap@in.ibm.com
Holger Horbach	IBM - holle@de.ibm.com
Klaus Keuerleber	IBM - keuerleb@de.ibm.com
Alexander Jung	IBM - ajung@de.ibm.com

IBM

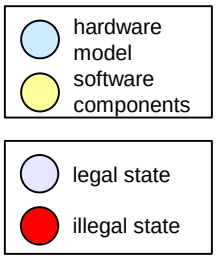
the limits of constrained random verification

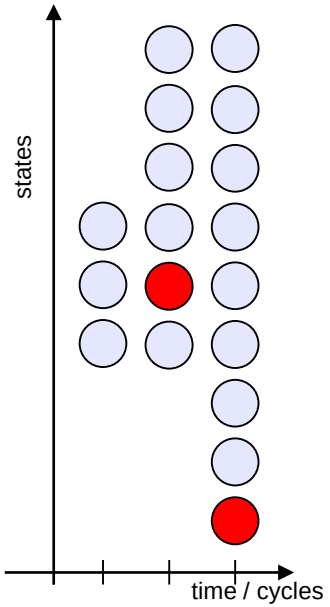
dealing with sparse state space

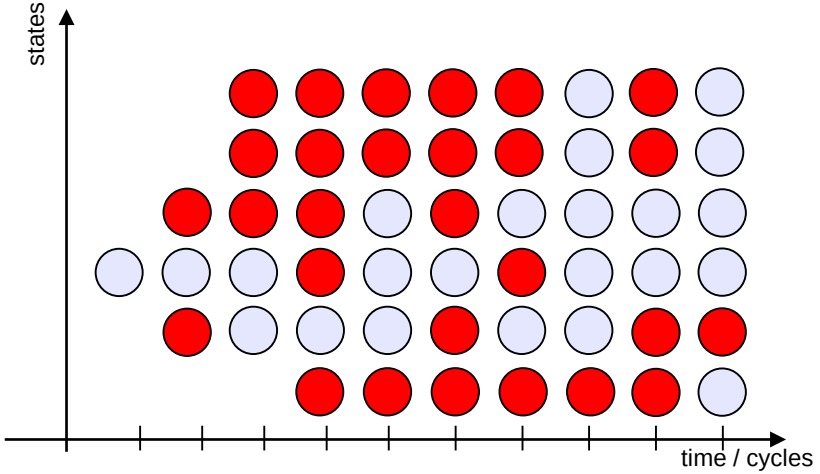


constrained random environments are well-suited for logic with broad state space with low sequential depth *"shotgun principle"*

… but they are weak in describing sequences (with random decisions) for logic with sparse state space and high sequential depth

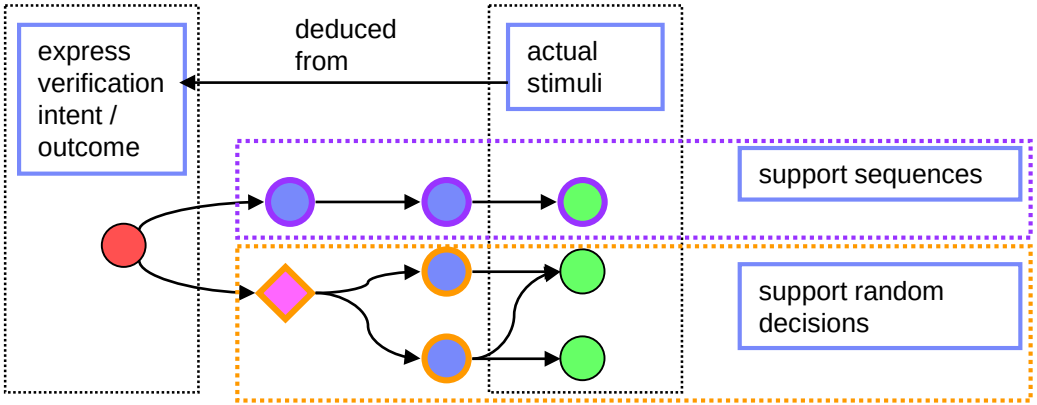




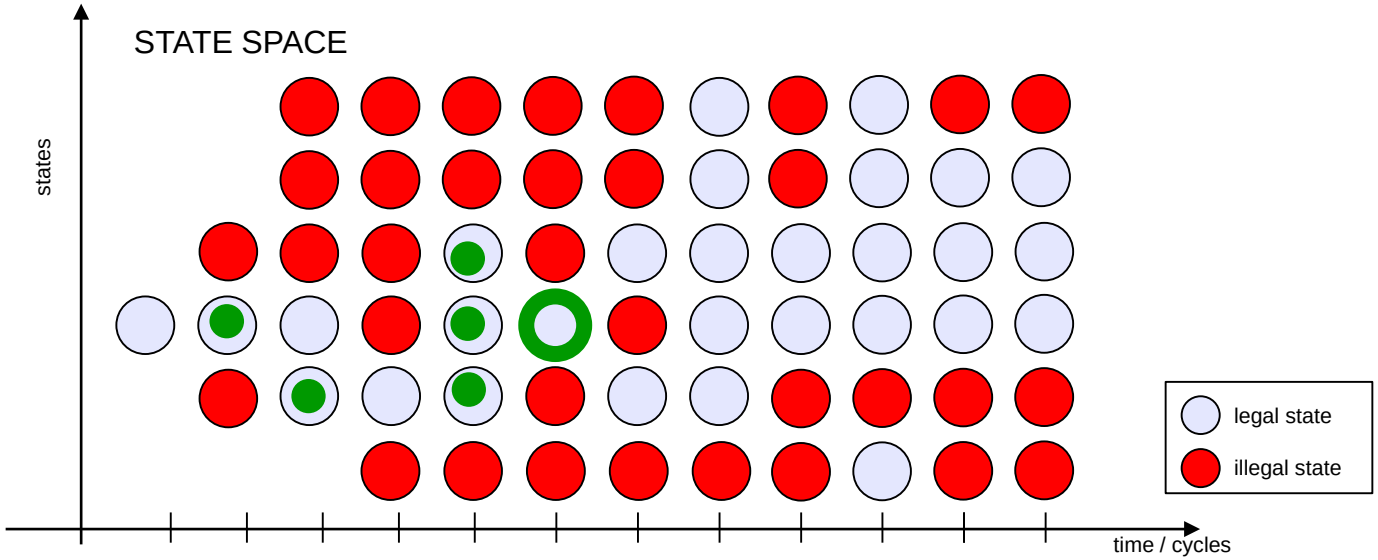


the power of graph-based verification

generating stimuli from a graph



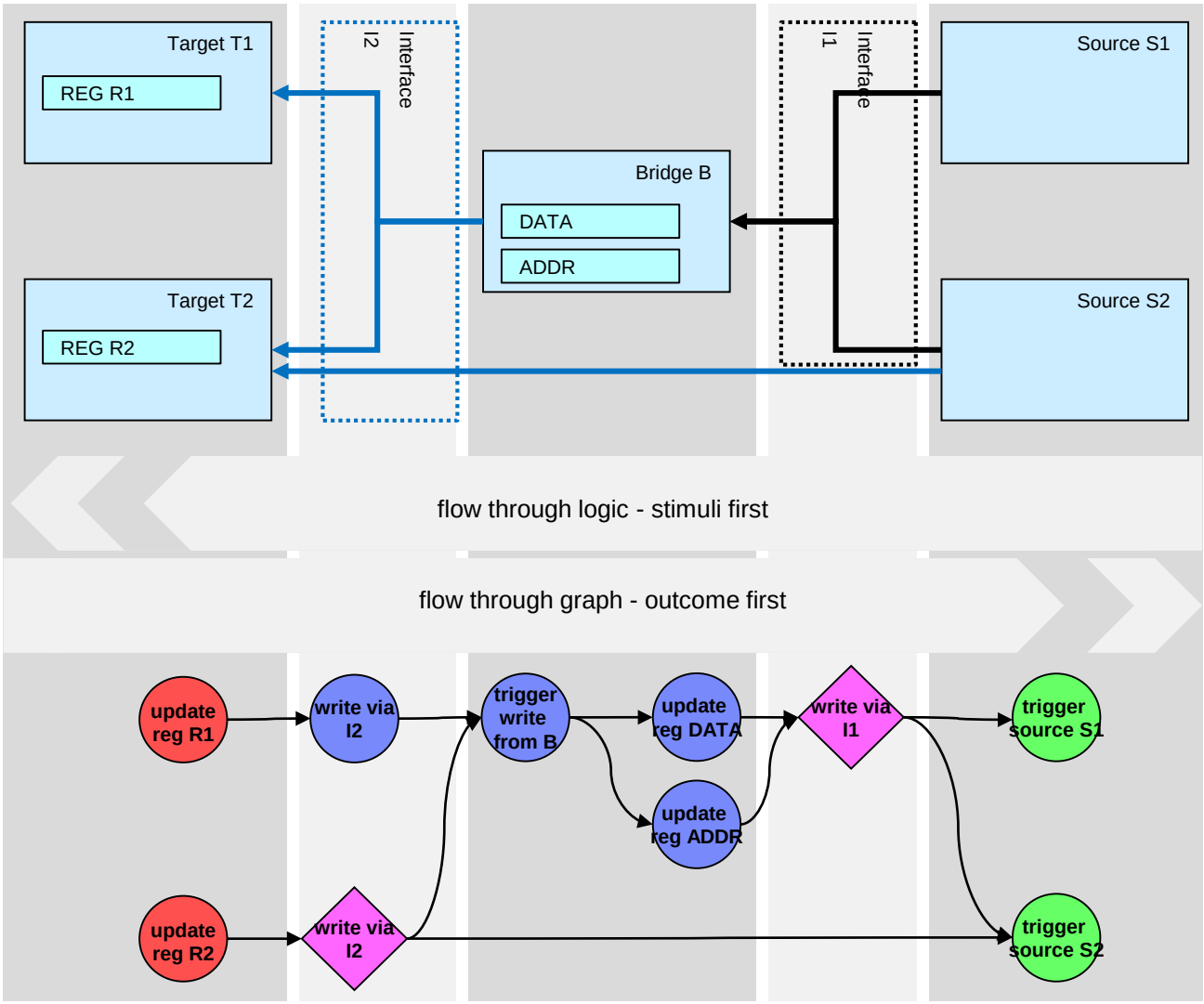
graphs are a great vehicle to describe the legal set of states through a combination of directed sequences (where state set is sparse) and random decisions (where state space opens up)



extending graph-based verification

describing verification intent

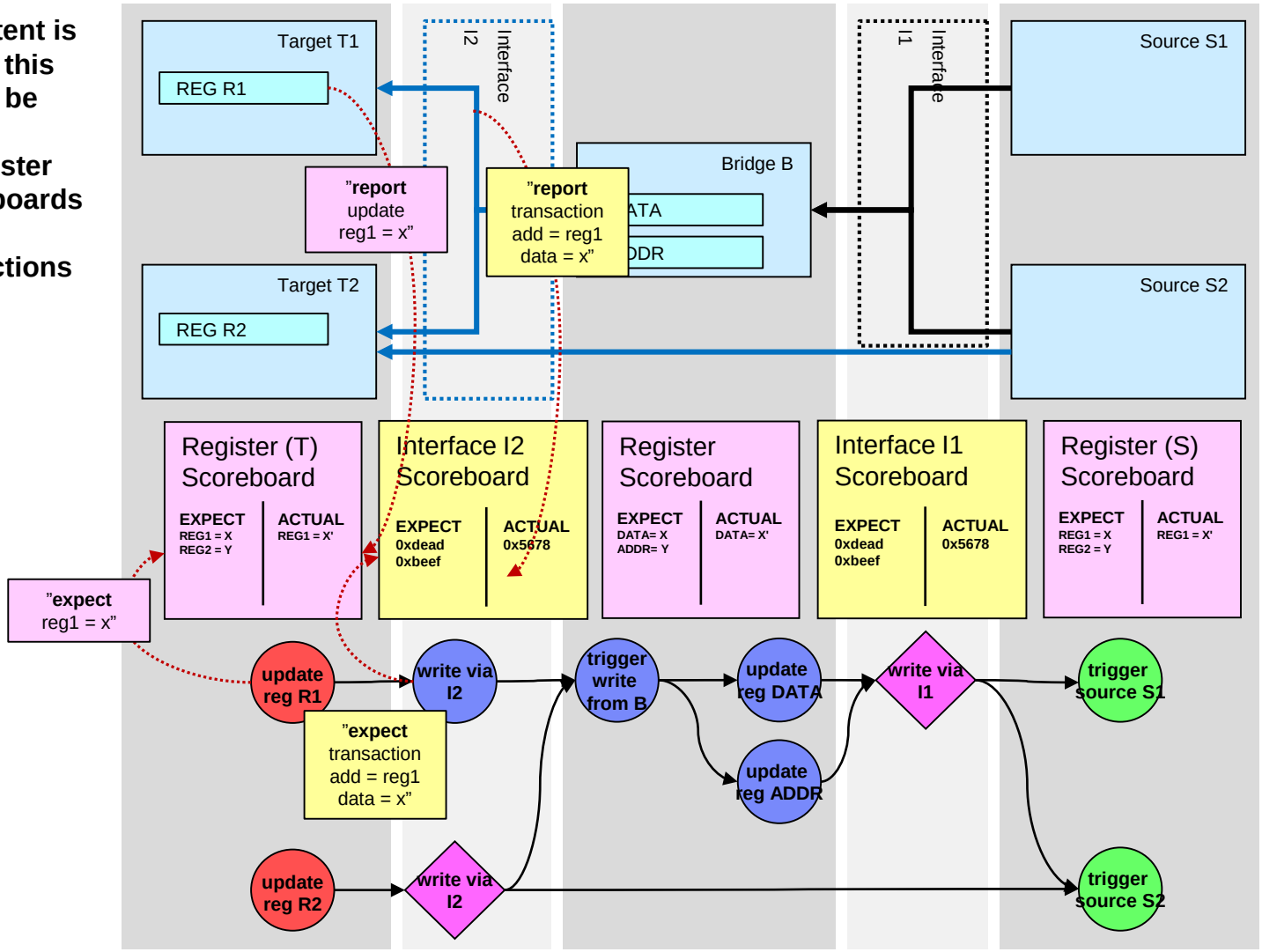
in a first step the graph is used to describe the verification intent (“outcome first”) and determining the input stimuli by walking the scenario graph



extending graph-based verification

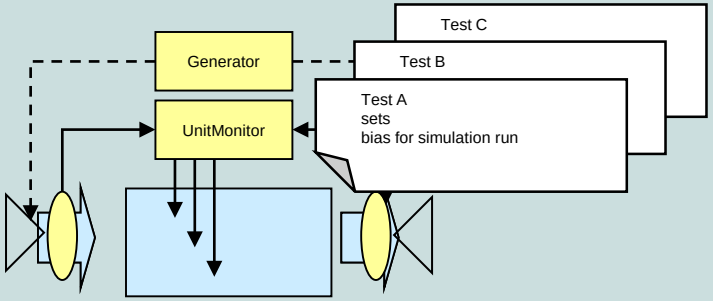
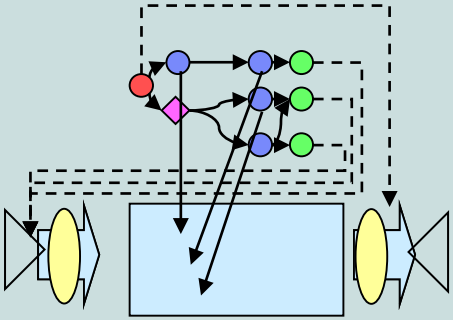
implementing checkers from the graph

since verification intent is expressed in graph, this knowledge can also be used for checking purposes using register and interface scoreboards to validate register updates and transactions



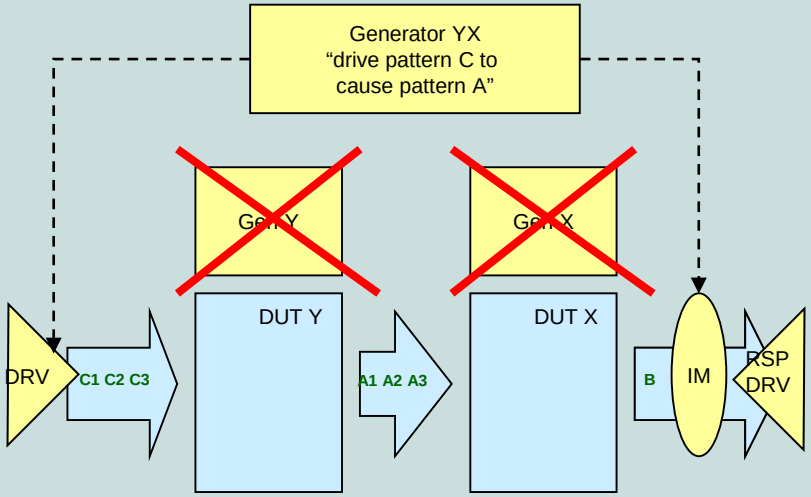
advantages of graph-based verification

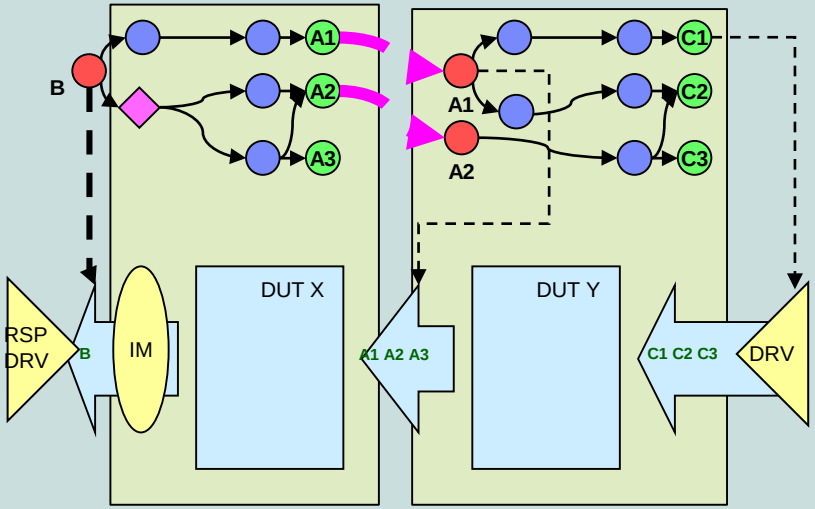
how the pervasive verification team benefits from the paradigm shift

from constrained random…	to graph-based environment
sequences implemented using directed tests or complex generators with constraints supplied as parms files 	scenario implemented within graph using a combination of sequence goals and select goals; each path through graph represents a possible scenario → *considerable increase in number of scenarios* 
strong unit focus with little emphasis on integration checking: “make sure the unit can cover every thinkable input pattern and react accordingly”	less unit focus but more emphasis on system scenarios: “make sure all units in system correctly interact and correctly support all use cases defined for the system”
testplan spread across large set of directed tests and parms files with constraints	testplan is part of graph that can be visualized and “walked” interactively
impossible to express interdependencies between separate dedicated testcases	graph is single source of tests and allows expressing interdependencies between concurrent scenarios → *significantly increasing subsystem coverage*

advantages of graph-based verification - con't

from constrained random…	to graph-based environment
set of heterogeneous unit environments with little to no reuse of generators at sub system level	set of unit graphs and test scenarios can be **chained** (since graphs are outcome-focused) and interconnected at sub system level → *very high level of vertical reuse / focus is on integration, but outcome-based thinking may feel unusual for experienced verif engineers*
simple cover points in simulation environment explicitly inserted by environment owners	built-in goal coverage as well as path-based coverage to describe sequences

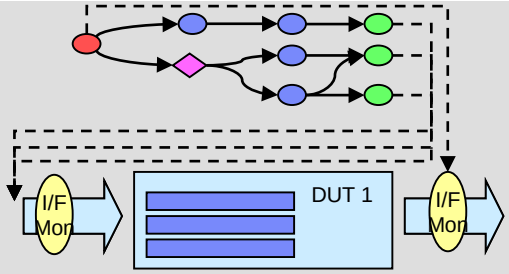




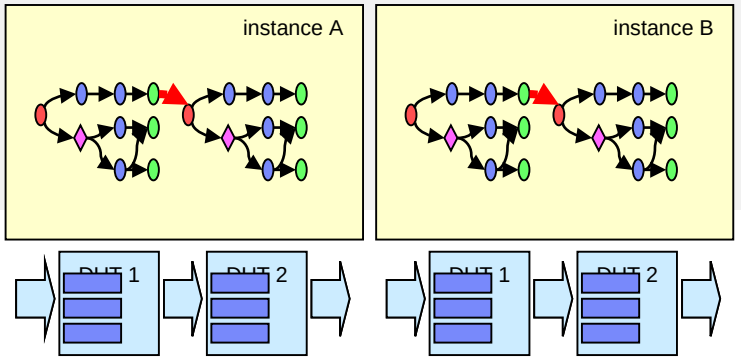
applying graph-based verification in real life

Graph-based verification environment (using Breker™ Trek™ tool) covering stimulus and register / interface checking successfully implemented for pervasive sub system of next-gen IBM POWER® processor:

- **30+ distinct graph modules implement functions offered by corresponding logic component:**
  - describing input stimulus at component boundaries
  - predicting expected changes for hardware registers
  - predicting expected transactions across interconnects between components



- **all modules integrated into singular sub system-level graph combining functions from modules to higher-level test scenarios such as**
  - power management scenarios
  - power-on-reset scenarios



- **robust set of basecode to support scoreboard checking**
- **team of 10+ verification engineers contributing to environment**
- **> 10000 nodes implemented (with replication)**

Legal statements

IBM and POWER are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

Trek is a trademark of Breker Verification Systems, Inc.
http://www.brekersystems.com/products/trek/