



ExPert: Dynamic Analysis Based Fault Location via Execution Perturbations

Neelam Gupta Rajiv Gupta

T. Chen, D. Jeffrey, V. Nagarajan, S. Tallam, X. Zhang

The University of Arizona

Funded by *NSF* grant **CNS-0614707** from the *CSR* program.



Automated Debugging of Software

Software Characteristics

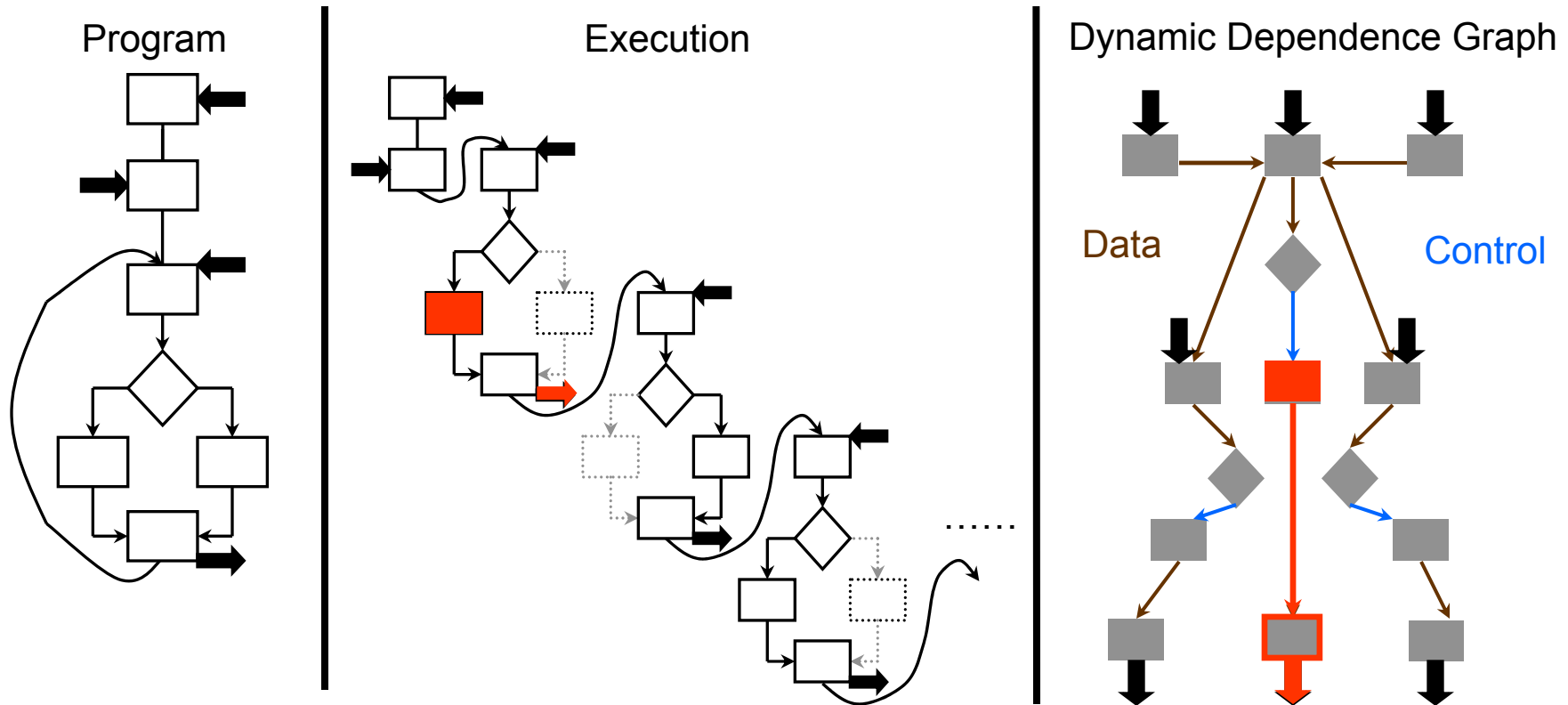
- ❑ *Long Running Programs*
- ❑ *Multithreaded Programs*

Goal: Assist the programmer in debugging by automatically narrowing the fault to a small section of the code.

- ❑ *Dynamic Information:* fine-grained execution traces
- ❑ *Execution Runs:* 1 failed execution & its perturbations
- ❑ *Scalable Analysis:* compressed traces & checkpointing/logging/replay.



Dynamic Information for Fault Location



Approach to Fault Location

Detect execution of statement s such that

- ❑ Faulty code *Affects* the value computed by s ; or
- ❑ Faulty code is *Affected-by* the value computed by s through a chain of dependences.

Estimate the set of potentially faulty statements from s :

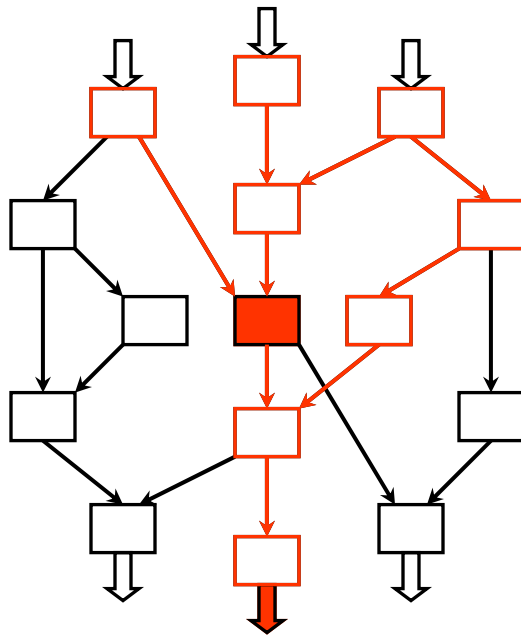
- ❑ *Affects*: statements from which s is reachable in the dynamic dependence graph. (Backward Slice)
- ❑ *Affected-by*: statements that are reachable from s in the dynamic dependence graph. (Forward Slice)

➔ *Intersect slices to obtain a smaller fault candidate set.*



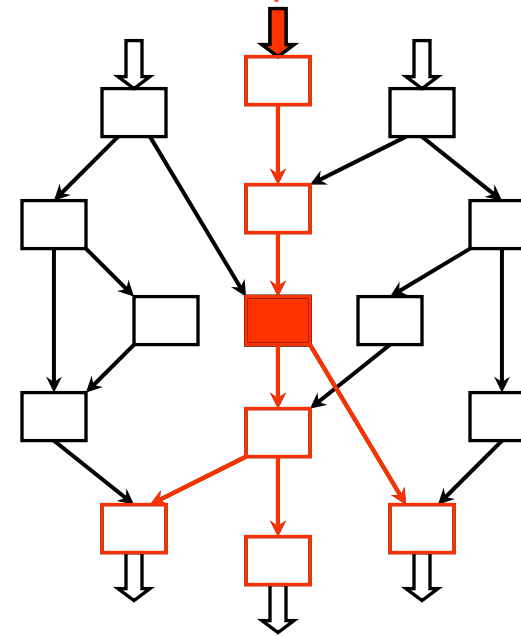
Backward and Forward Slices

Backward Slice



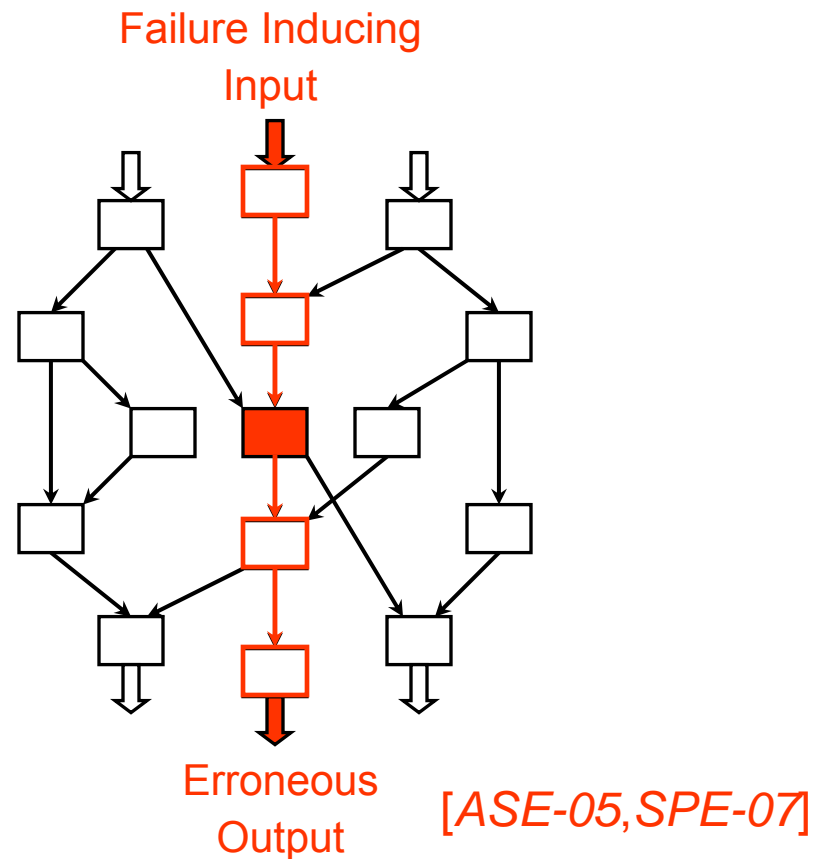
Erroneous Output [Korel&Laski, 1988]

Failure inducing Input



Forward Slice [ASE-05, SPE-07]

Backward and Forward Slices



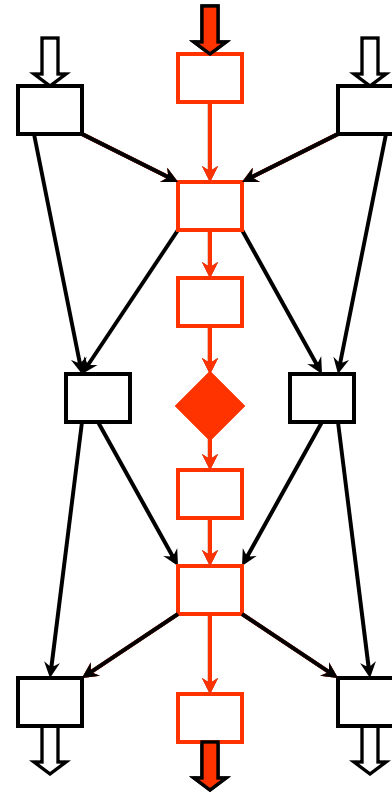
→ For memory bugs the number of statements is very small (< 5).



Bidirectional Slices of Critical Predicates

Critical Predicate:

An execution instance of a predicate such that changing its outcome “repairs” the program state.

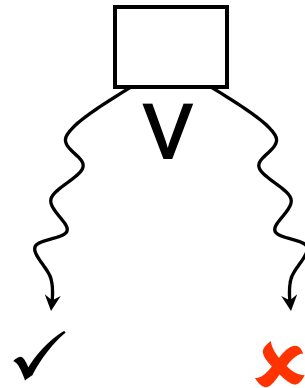
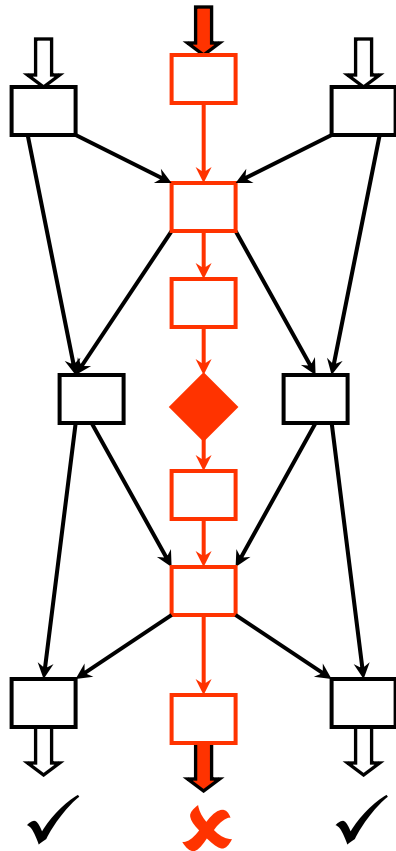


Combined
Slice
[ICSE-06,SPE-07]

- Found critical predicates in **12 out of 15** bugs
- Search for critical predicate:
Brute force: 32 predicates to 155K predicates;
After *Filtering* and *Ordering*: 1 predicate to 7K predicates.



Pruning Potentially Faulty Code

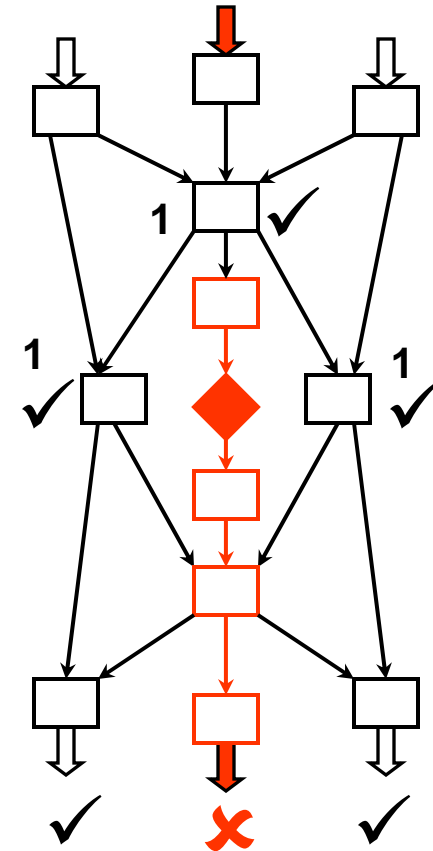


Confidence in V

$C(V): [0, 1]$

1 - any change in V will change ✓

0 - all values of V produce same ✓



[PLDI-06]

How? *Value profiles.*



Test Programs

Real Reported Bugs

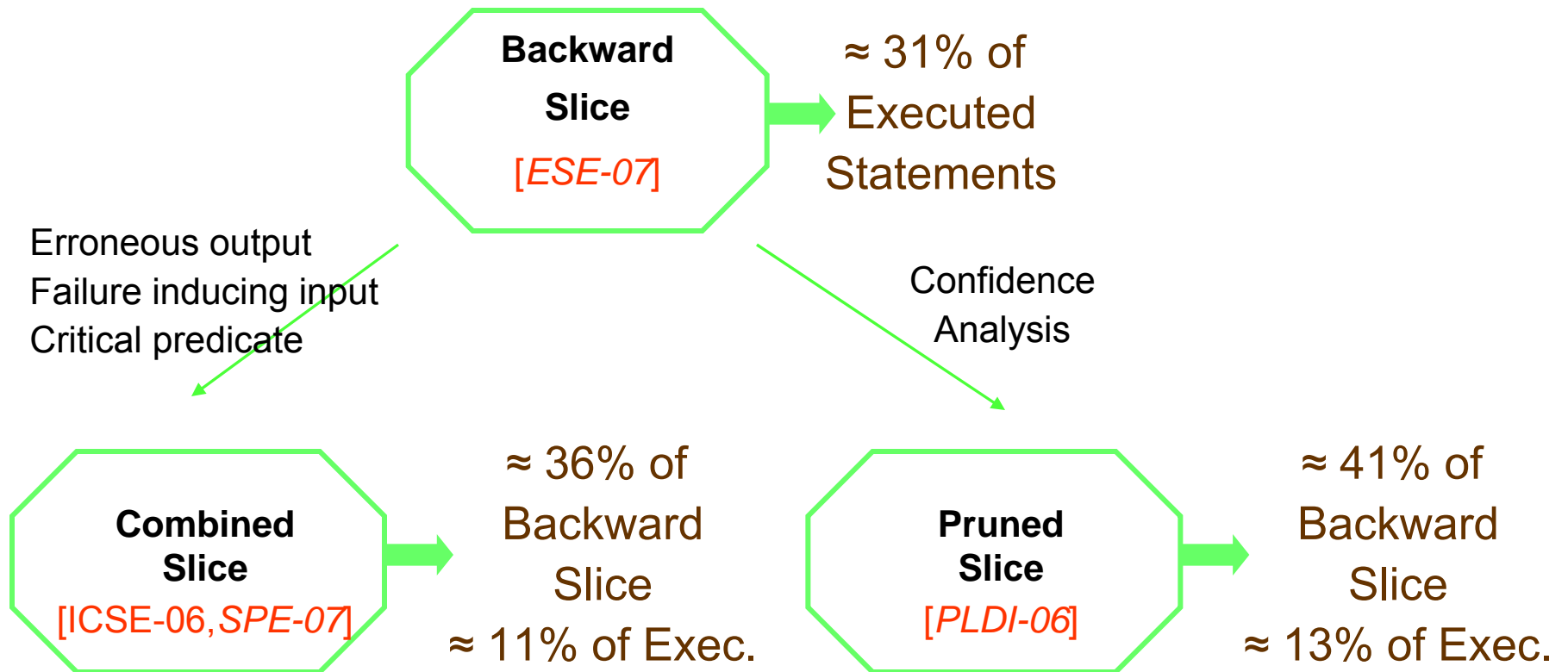
- ❑ Nine logical bugs (incorrect output)
 - **Unix utilities**
 - ❖ grep 2.5, grep 2.5.1, flex 2.5.31, make 3.80.
- ❑ Six memory bugs (program crashes)
 - **Unix utilities**
 - ❖ gzip, ncompress, polymorph, tar, bc, tidy.

Injected Bugs

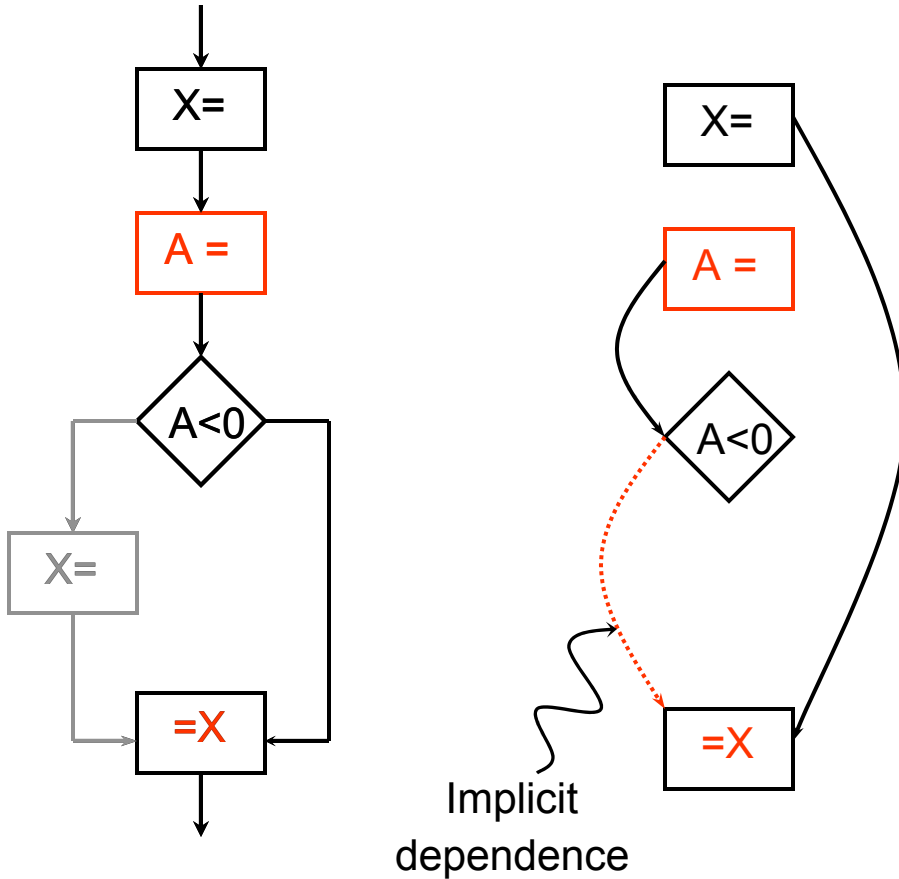
- ❑ Siemens Suite (numerous versions)
 - ❖ schedule, schedule2, replace, print_tokens..
- **Unix utilities**
 - ❖ gzip, flex



Effectiveness of Fault Location



Execution Omission Errors



- Inspect pruned slice.
- Dynamically detect an Implicit dependence.
- Incrementally expand the pruned slice.

[PLDI-07]



Trace Representation

Dynamic Information Needed

- Dynamic Dependences
 - ❖ for all slicing
- Values for Confidence Analysis
 - ❖ for pruning slices

➔ annotates the static program representation

Whole Execution Trace (WET) [MICRO-04, TACO-05]

□ Trace Size

- Before Compaction
 - ≈ 15 Bytes / Instruction
- After Compaction
 - ≈ 4 Bits / Instruction



Dependence Graph Generation Times

- ❖ *Offline* post-processing after collecting address and control flow traces
 - ➔ $\approx 35x$ slowdown
- ❖ *Online* techniques
 - ➔ *Information Flow*: 9x to 18x slowdown
 - ➔ *Basic block Opt.*: 6x to 10x slowdown
 - ➔ *Trace level Opt.*: 5.5x to 7.5x slowdown
 - ➔ *Dual Core*: $\approx 1.5x$ slowdown
- ❖ *Online* Filtering techniques
 - ➔ Forward slice of all inputs
 - ➔ User-guided bypassing of functions

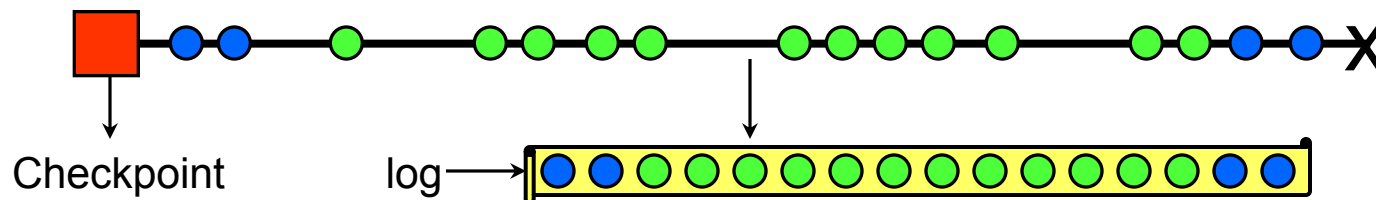


Beyond Tracing: Record (log) and Replay

- ❑ *Checkpoint*: capture memory image.

[FSE-06]

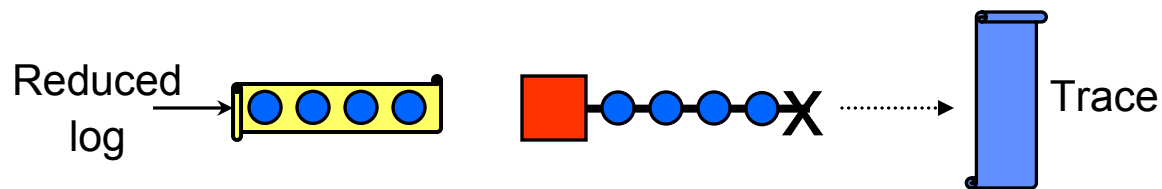
- ❑ Execute and *Record (log) Events*.



- ❑ *Upon Crash, Rollback* to checkpoint.

- ❑ *Reduce log* and *Replay* execution using reduced log.

- ❑ Turn on *tracing* during replay.



➔ *Applicable to Multithreaded Programs*

Debugging System

