Attacking Symbolic Execution Issues

- Sergii Vozniuk

- Advanced Topics in Software Systems
- December 9, 2011

Motivation

- Scalability / Performance (4)
- Usability / Applicability (3)
- Functionality vs Correctness (2)
- The Oracle issue (2)
- Bugs classification

S^2E* Idea

All paths are equal,

but some path are more equal than others.

* V. Chipounov, V. Kuznetsov, G. Candea S2E: A Platform for In-Vivo Multi-Path Analysis of Software Systems, *ASPLOS 2011*

S^2E* Idea

All paths are equal,

but some path are more equal than others.

OR

Often only some families of paths are of interest.

* V. Chipounov, V. Kuznetsov, G. Candea S2E: A Platform for In-Vivo Multi-Path Analysis of Software Systems, *ASPLOS 2011*

S^2E* Idea

All paths are equal,

but some path are more equal than others.

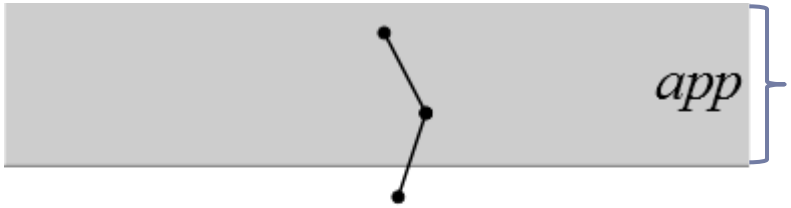
OR

Often only some families of paths are of interest.

Mainly attacks performance and applicability issues.

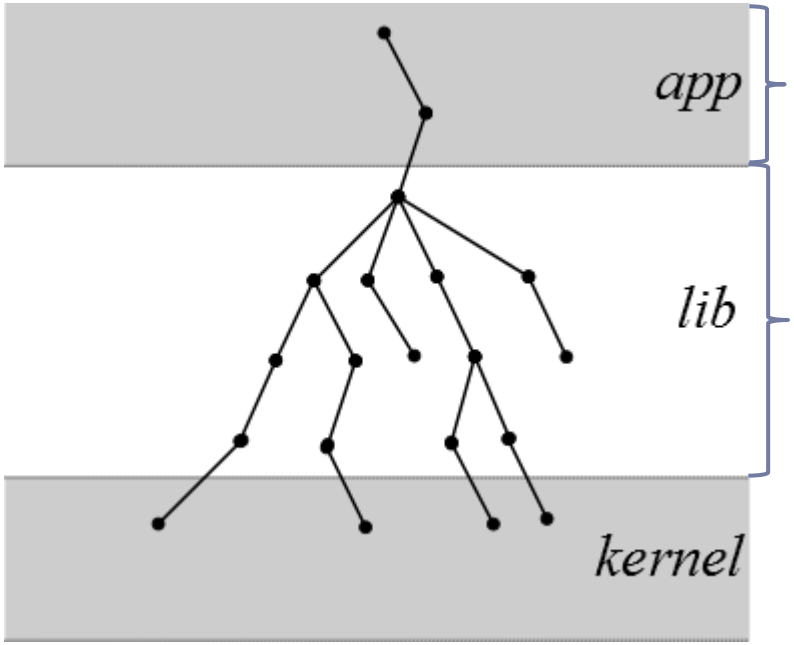
* V. Chipounov, V. Kuznetsov, G. Candea S2E: A Platform for In-Vivo Multi-Path Analysis of Software Systems, *ASPLOS 2011*

S²E multi-path & single path modes



Single-path (concrete)

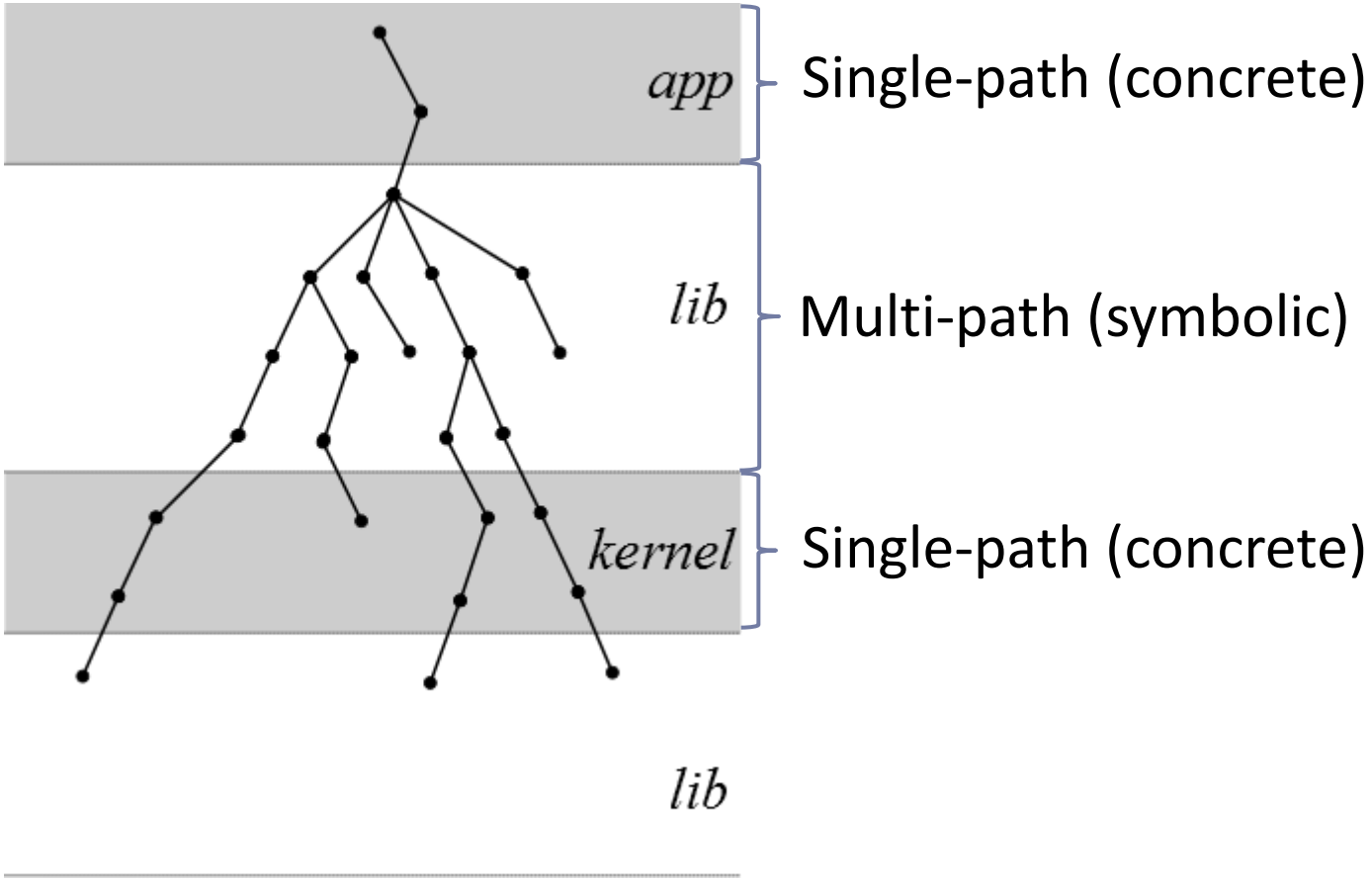
S^{2}E multi-path & single path modes



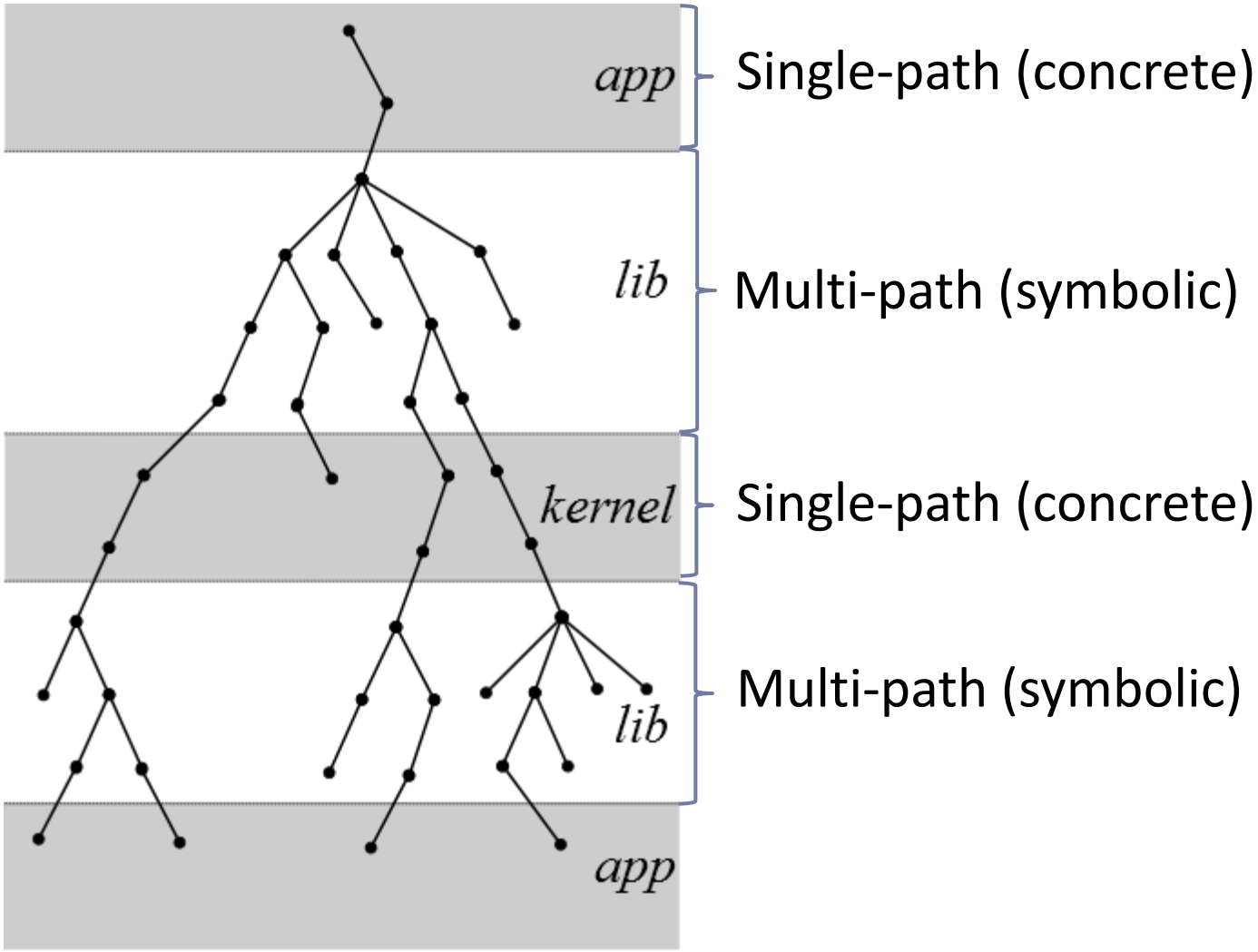
Single-path (concrete)

Multi-path (symbolic)

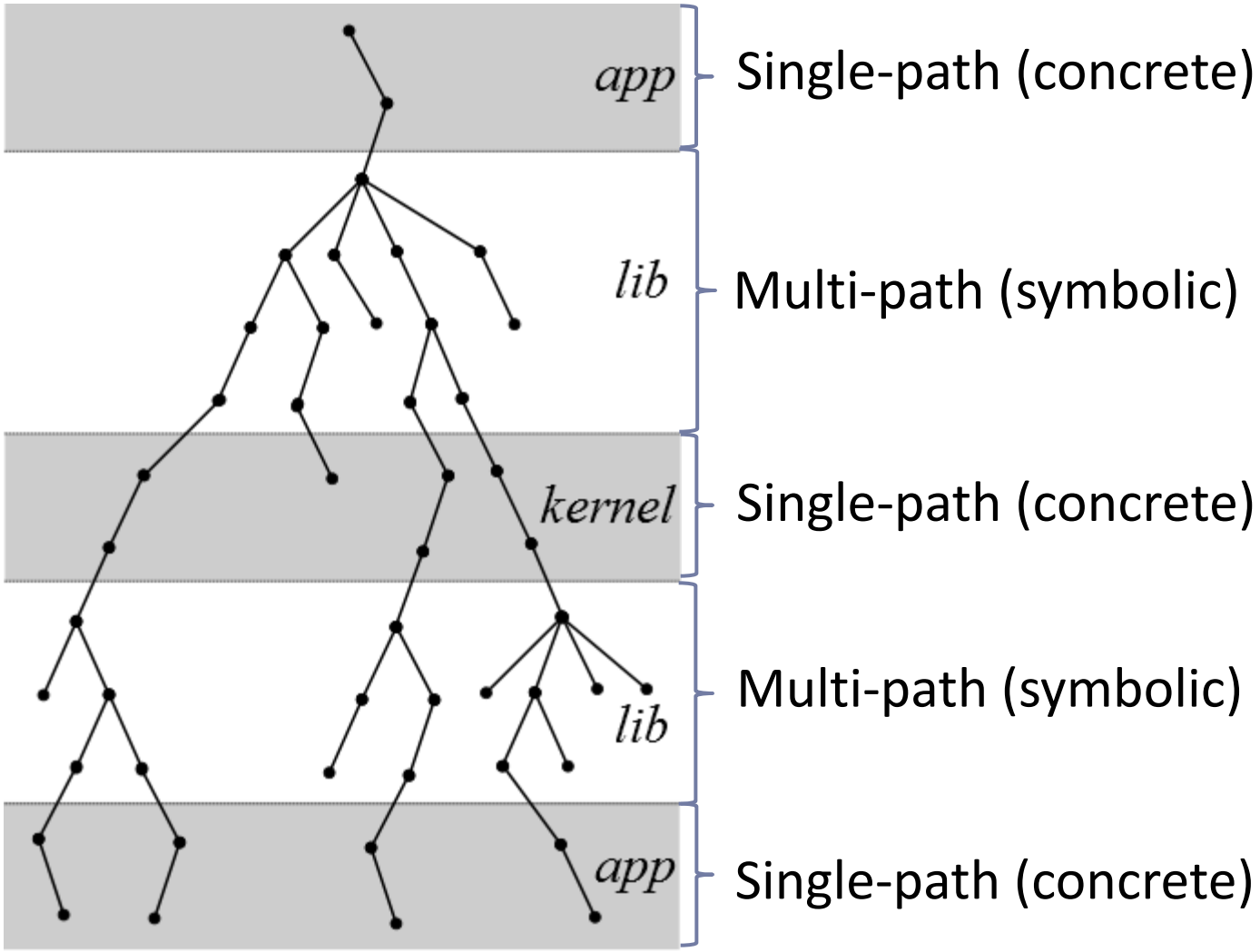
S²E multi-path & single path modes



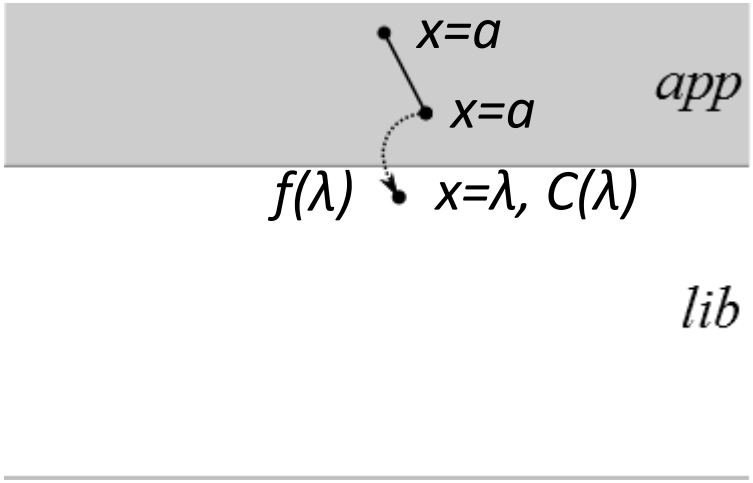
S²E multi-path & single path modes



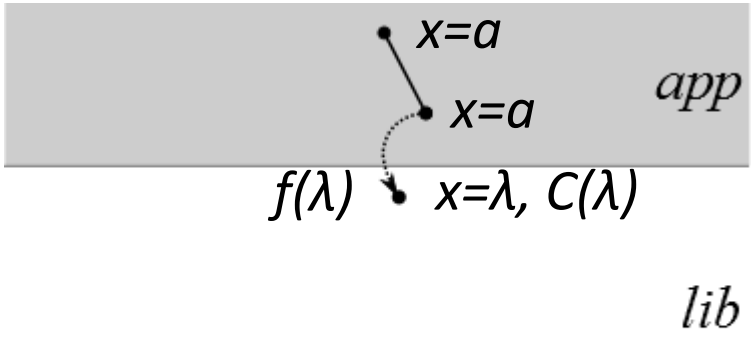
S²E multi-path & single path modes



Concrete ⟶ Symbolic Transition & Symbolic ⟶ Concrete Transition

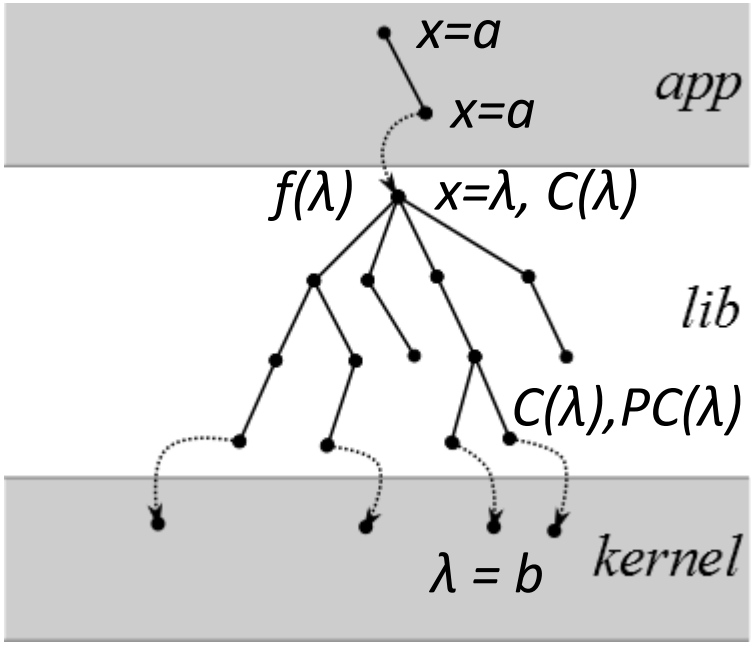


Concrete ⟶ Symbolic Transition & Symbolic ⟶ Concrete Transition



We may add additional constraints on *λ: C(λ)*

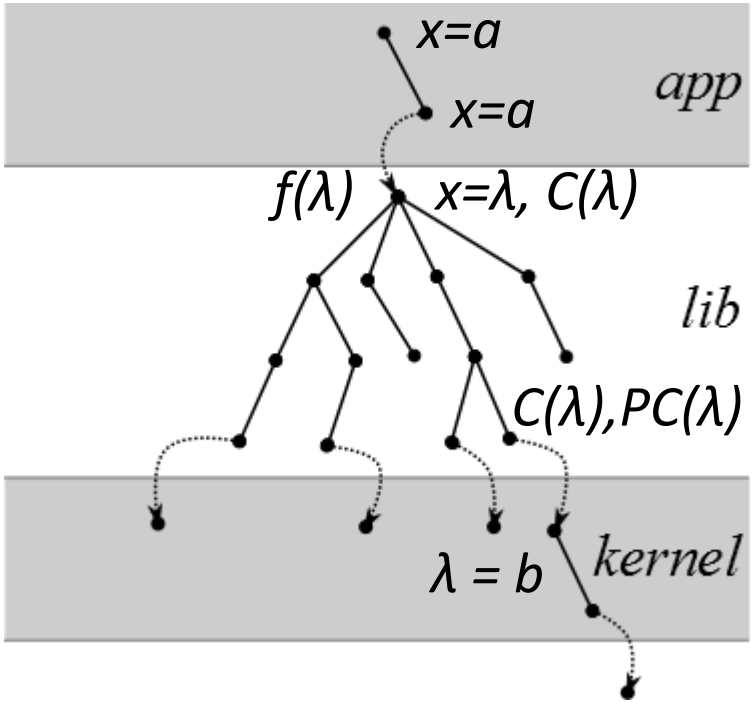
Concrete ⟶ Symbolic Transition & Symbolic ⟶ Concrete Transition



We may add additional constraints on *λ: C(λ)*

Gathered path constraints on *λ* for some path *PC(λ)*

Concrete ⟶ Symbolic Transition & Symbolic ⟶ Concrete Transition

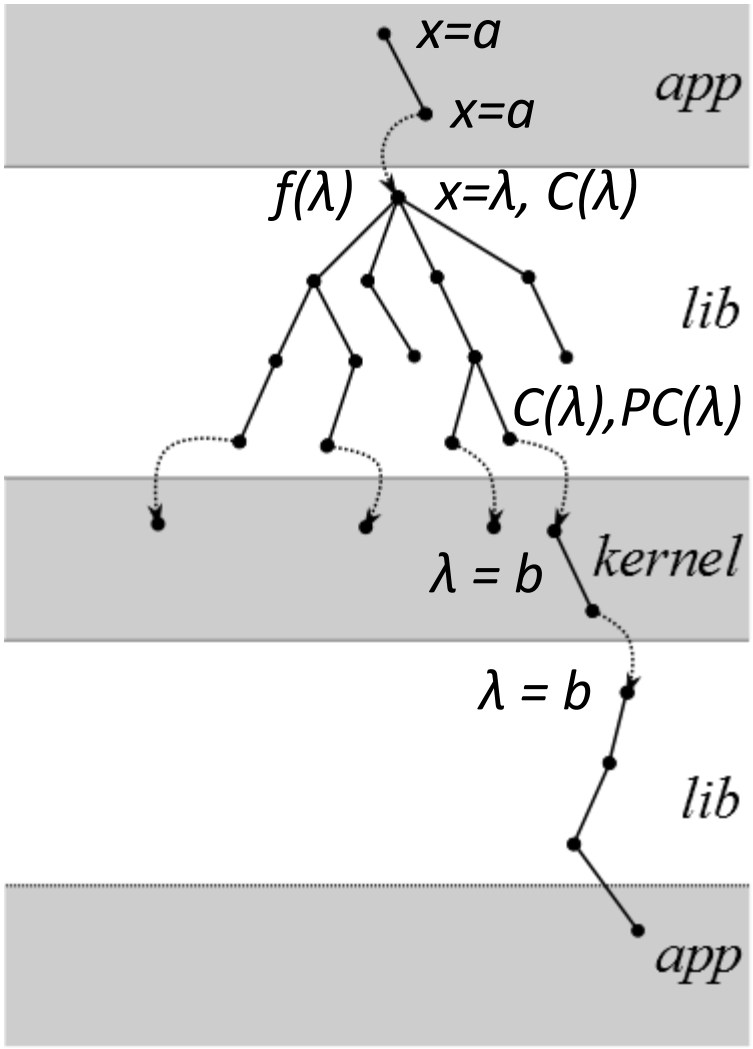


We may add additional constraints on *λ: C(λ)*

Gathered path constraints on *λ* for some path *PC(λ)*

Concretize *λ=b* such that *PC(b)=T and C(b)=T*

Concrete ⟶ Symbolic Transition & Symbolic ⟶ Concrete Transition



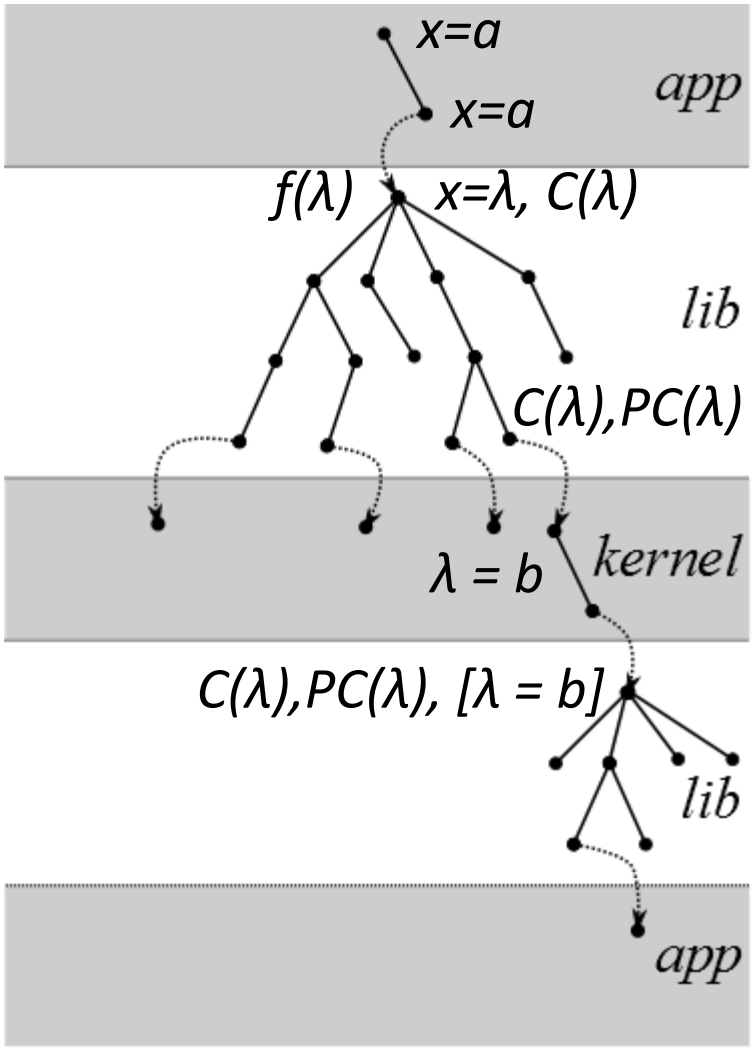
We may add additional constraints on *λ: C(λ)*

Gathered path constraints on *λ* for some path *PC(λ)*

Concretize *λ=b* such that *PC(b)=T and C(b)=T*

Return from concrete domain *λ = b.* Overconstraining. Equivalent to concrete execution from this point.

Concrete ⟶ Symbolic Transition & Symbolic ⟶ Concrete Transition



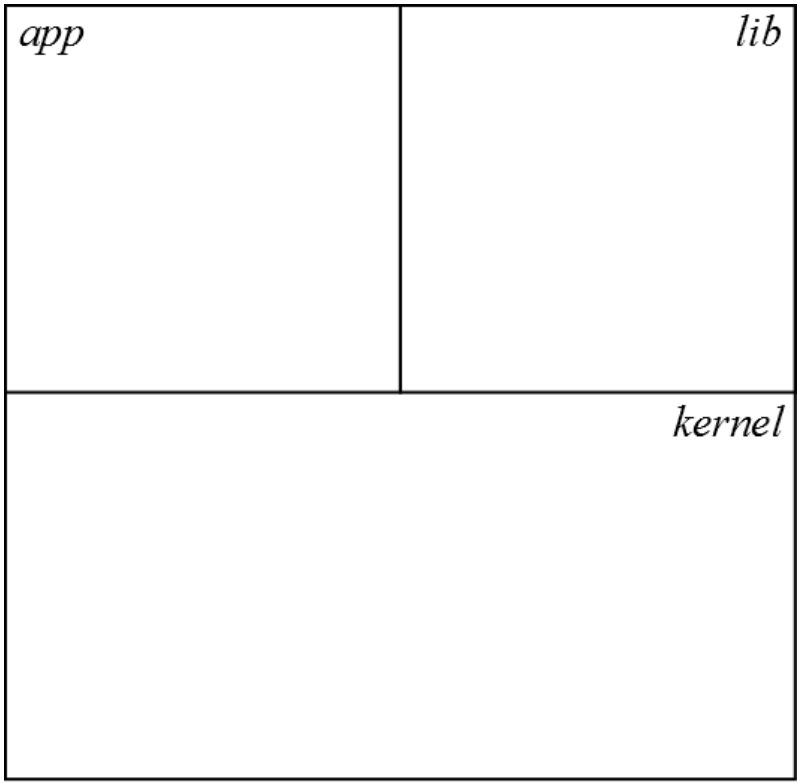
We may add additional constraints on *λ: C(λ)*

Gathered path constraints on *λ* for some path *PC(λ)*

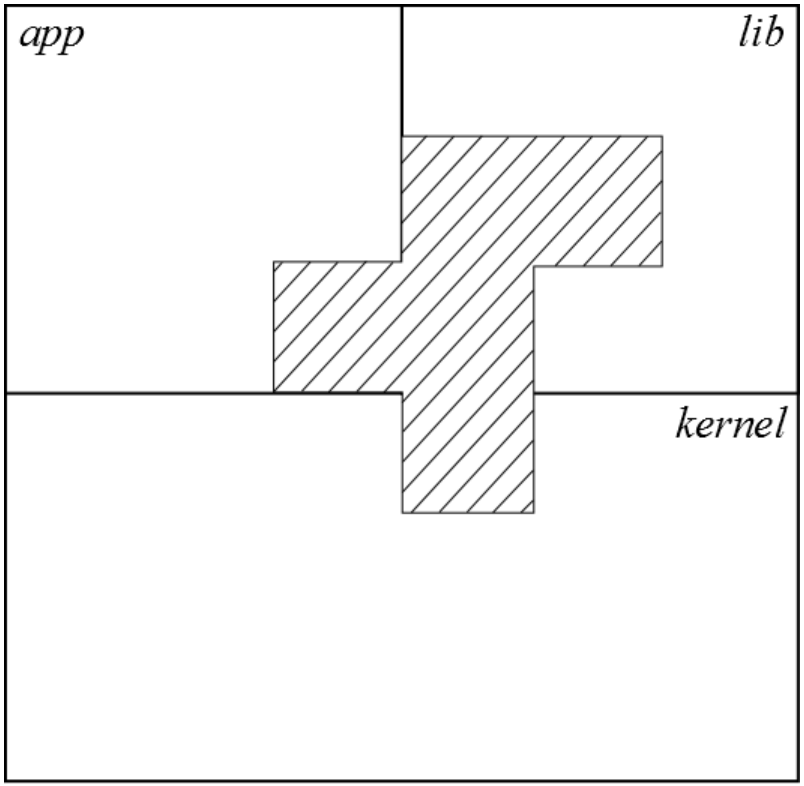
Concretize *λ=b* such that *PC(b)=T and C(b)=T*

Return from concrete domain *λ = b.* Soft constraint. Continue execution symbolically

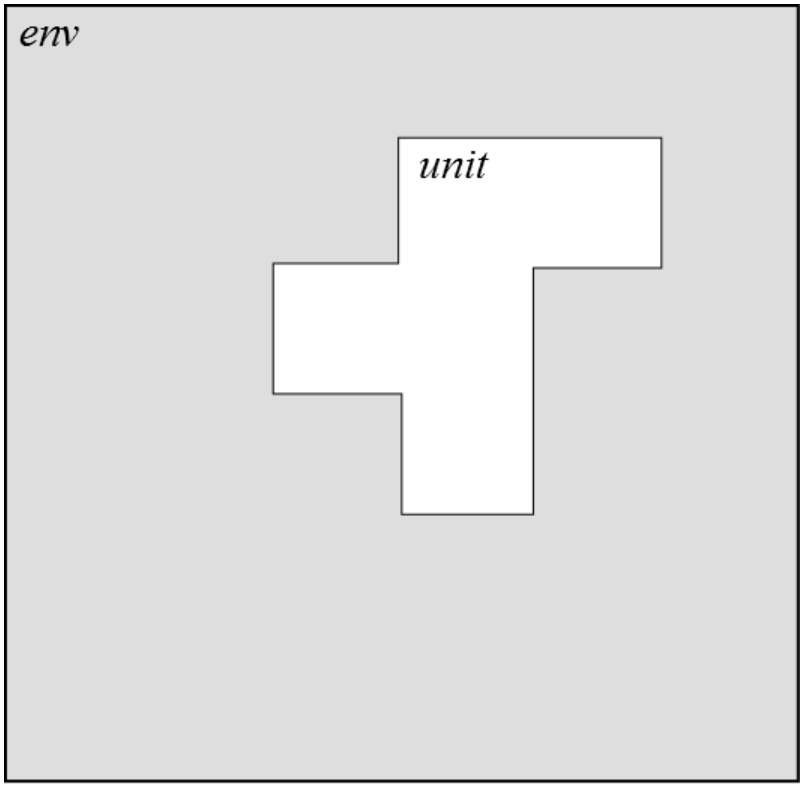
System, Unit, Environment



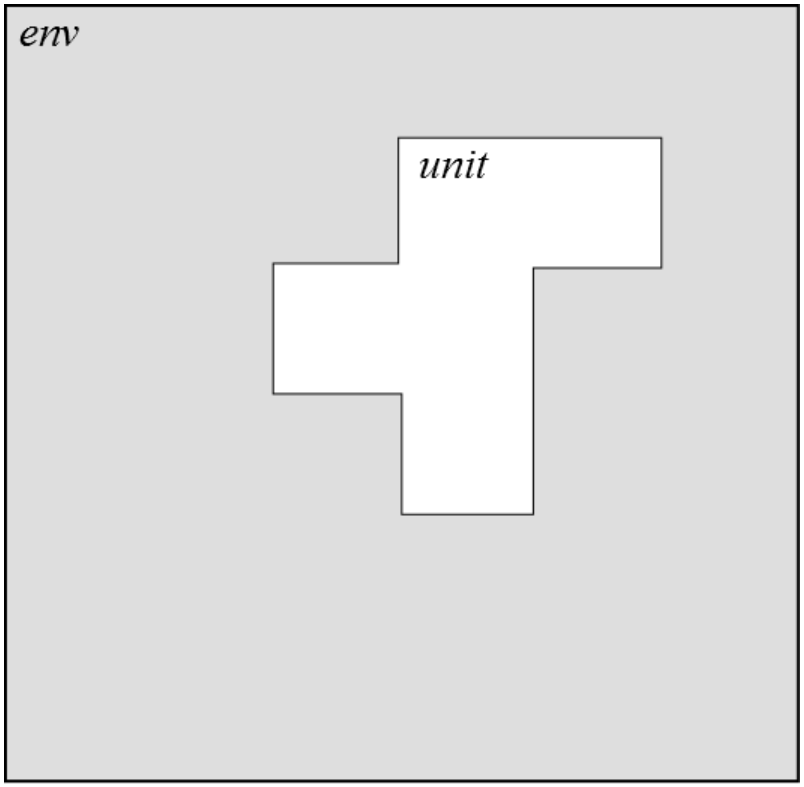
System, Unit, Environment

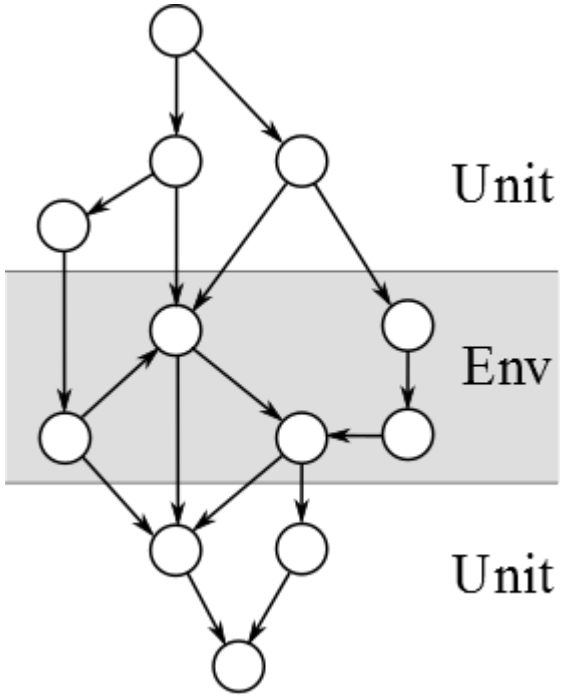


System, Unit, Environment

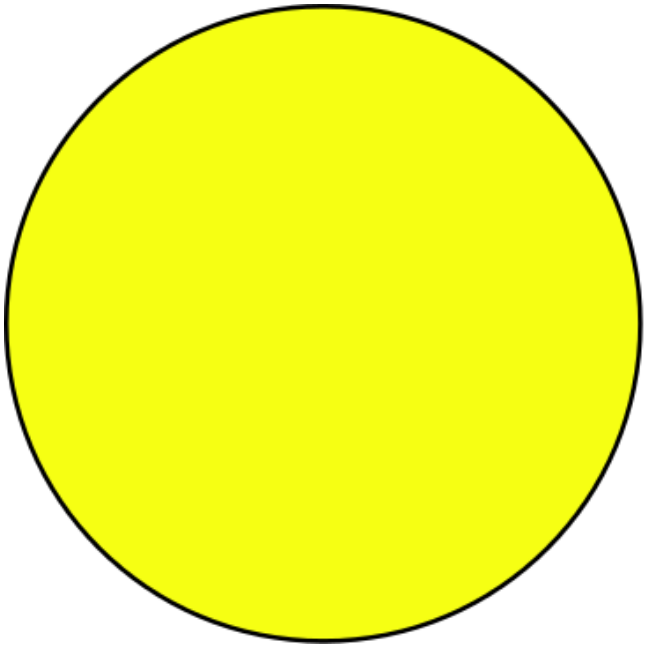


System, Unit, Environment

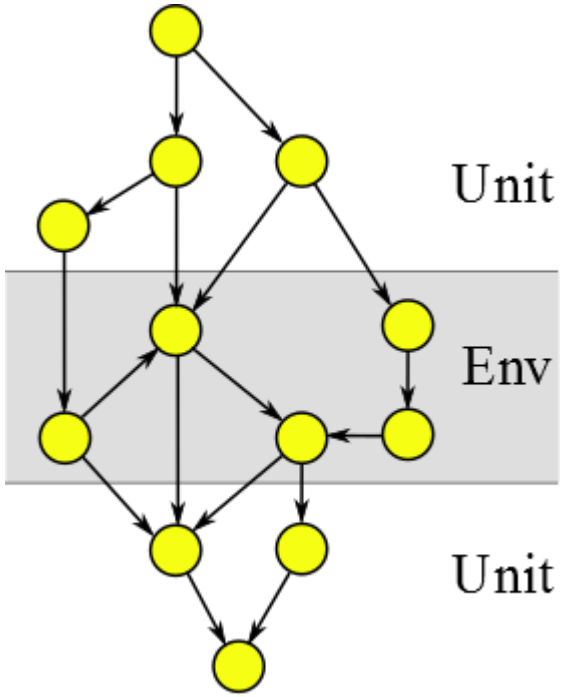




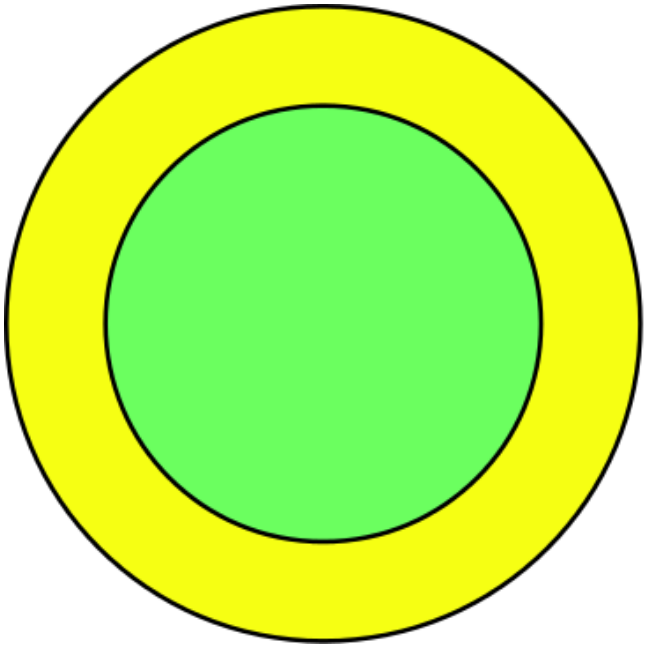
Paths feasibility



Statically feasible

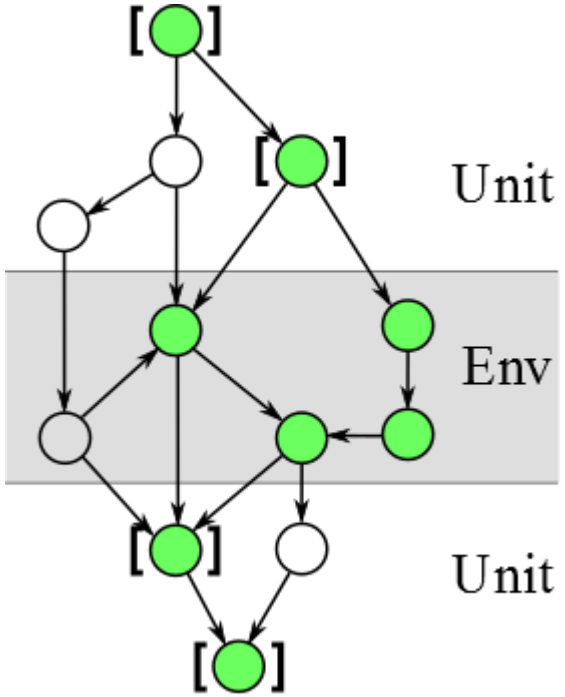


Paths feasibility

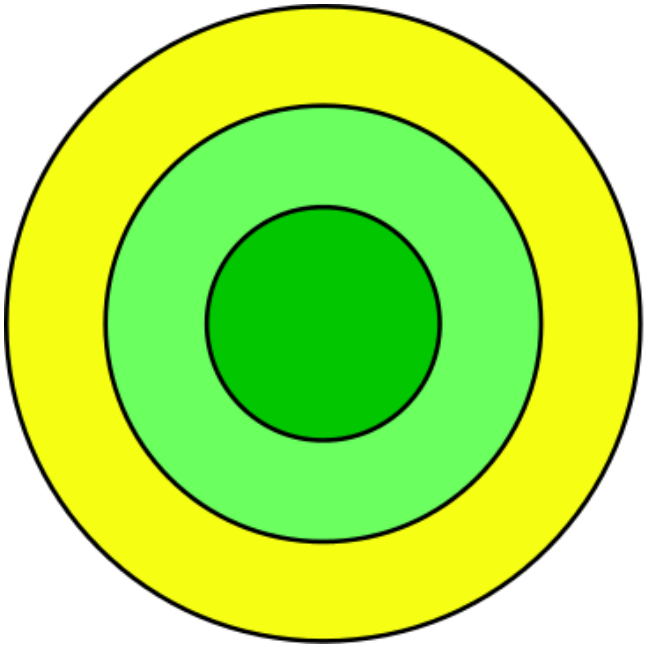


Statically feasible

Locally feasible



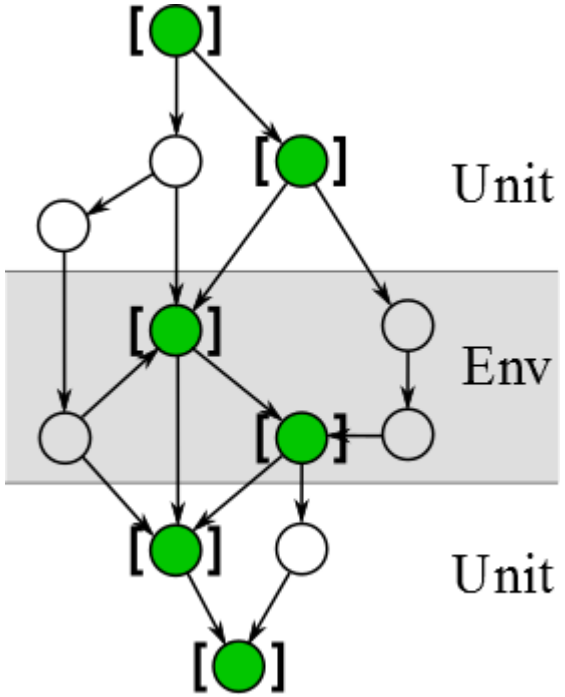
Paths feasibility



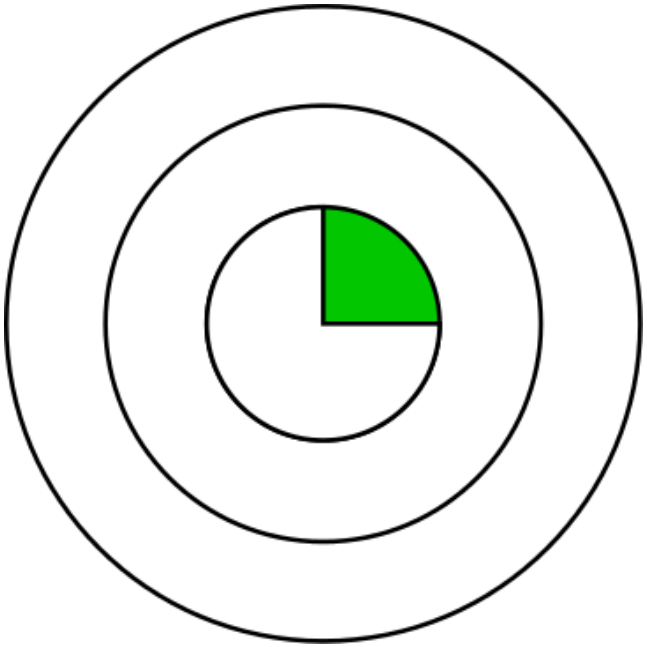
Statically feasible

Locally feasible

Globally feasible



Strictly Consistent Concrete Execution (SC-CE)



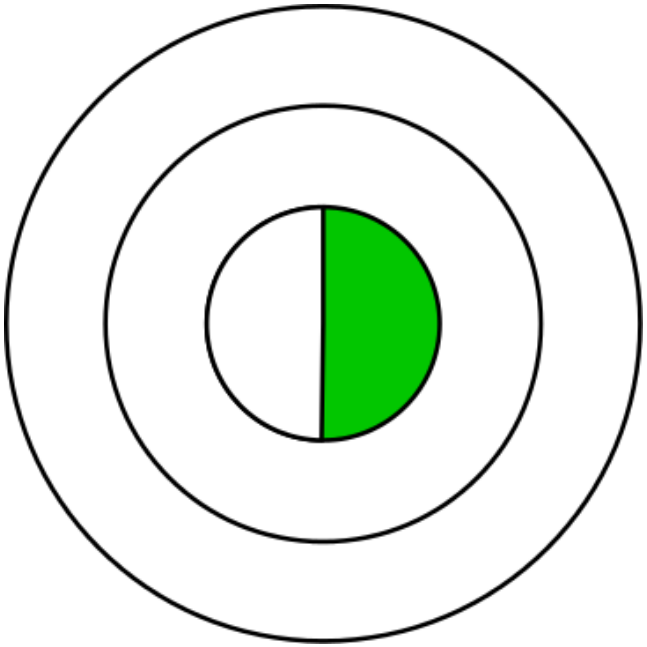
Properties:

- Only globally consistent paths
- Concrete execution

Example:

- Classic fuzzing

Strictly Consistent Unit-level Execution (SC-UE)



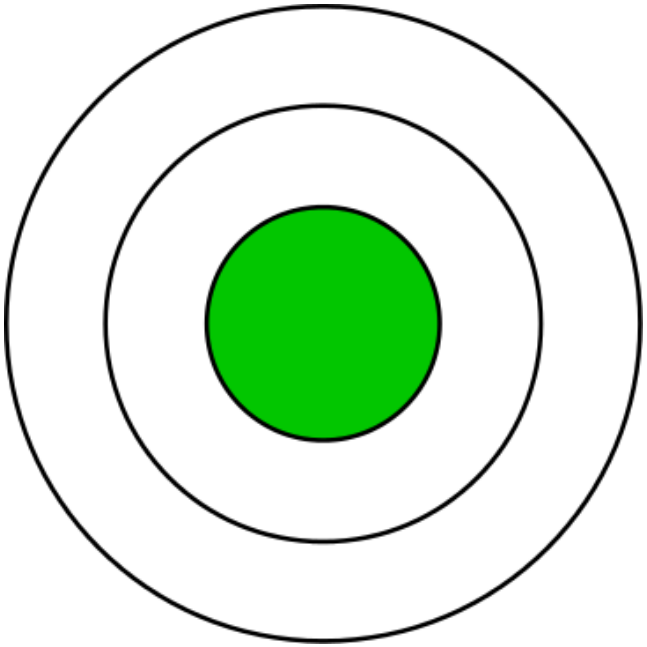
Properties:

- Only globally consistent paths
- **Path constraints for the unit**

Example:

- DART, KLEE

Strictly Consistent System-level Execution (SC-SE)



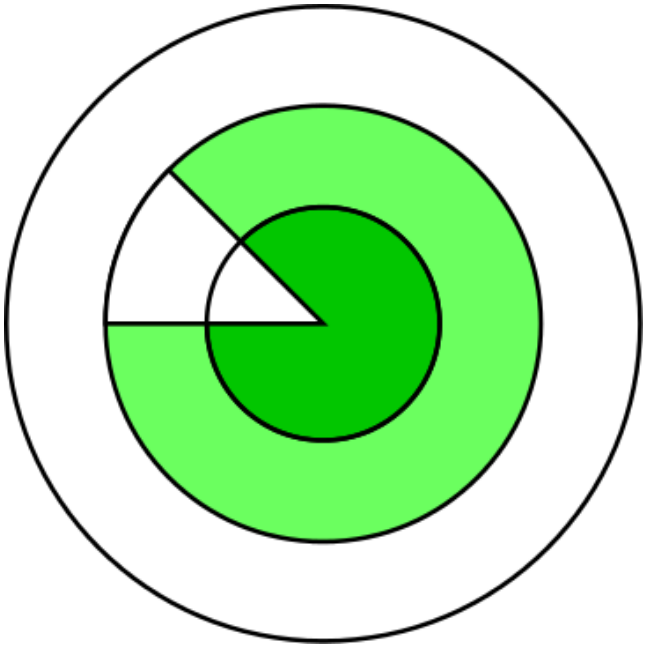
Properties:

- **Complete and consistent**
- **Path constraints for the env.**
- Path constraints for the unit

Example:

- KLEE (if env. is modeled)

Local Consistency (LC)



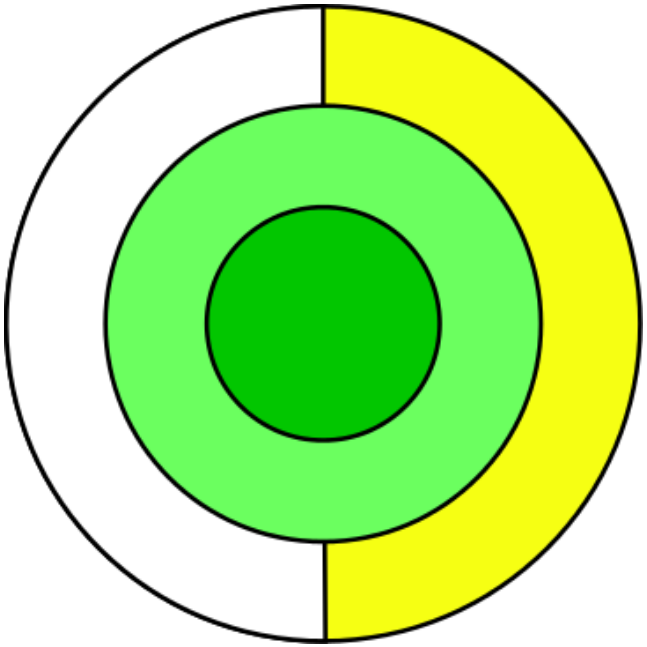
Properties:

- **Incomplete and inconsistent**
- **Locally consistent**
- **Follows environment contracts**

Example:

- S^2E (DDT^+)

Overapproximate Consistency(RC-OC)



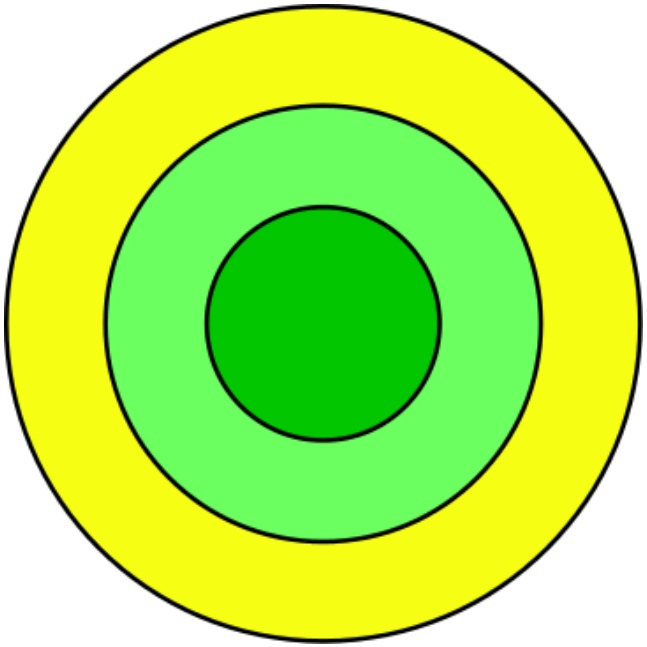
Properties:

- **Complete and inconsistent**
- **Prone to false-positives**
- **Disregard unit/env. interface contracts**

Example:

- Static analyzers
- Reverse engineering tools (REV^{+})

CFG Consistency(RC-CC)



Properties:

- **Complete and inconsistent**
- **Prone to false-positives**
- **May change any system state**

Example:

- Static analyzers
- Reverse engineering tools

User Interface

- Path selection
  - Data based selection
  - Code based selection
  - Priority based selection
- Path analysis
  - Data race detector
  - Memory checker
  - Execution tracer
  - Performance profile analyzer
  - WinDriveMon and WinBugCheck (Windows-specific)

S^2E Use Cases

- S^2E main goal is to enable rapid prototyping of useful, deep system analysis tools
- Tools developed
  - Testing of proprietary device drivers (DDT^+).
  - Reverse engineering of closed-source drivers (REV^+).
  - Multi-path in-vivo performance profiling ($PROF_S$).

Cloud9* Idea

Divide & Conquer

* S. Bucur, V. Ureche, C. Zamfir, G. Candea Parallel Symbolic Execution for Automated Real-World Software Testing, *EUROSYS 2011*

Cloud9* Idea

Divide & Conquer

Scalable parallelization of symbolic execution

* S. Bucur, V. Ureche, C. Zamfir, G. Candea Parallel Symbolic Execution for Automated Real-World Software Testing, *EUROSYS 2011*

Cloud9* Idea

Divide & Conquer

Scalable parallelization of symbolic execution

Attacks scalability, applicability and usability

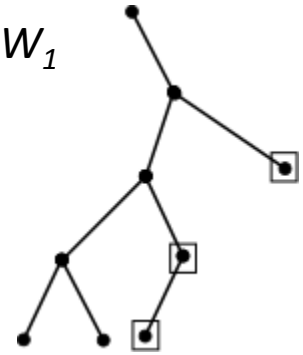
* S. Bucur, V. Ureche, C. Zamfir, G. Candea Parallel Symbolic Execution for Automated Real-World Software Testing, *EUROSYS 2011*

Parallel Symbolic Execution

W_1

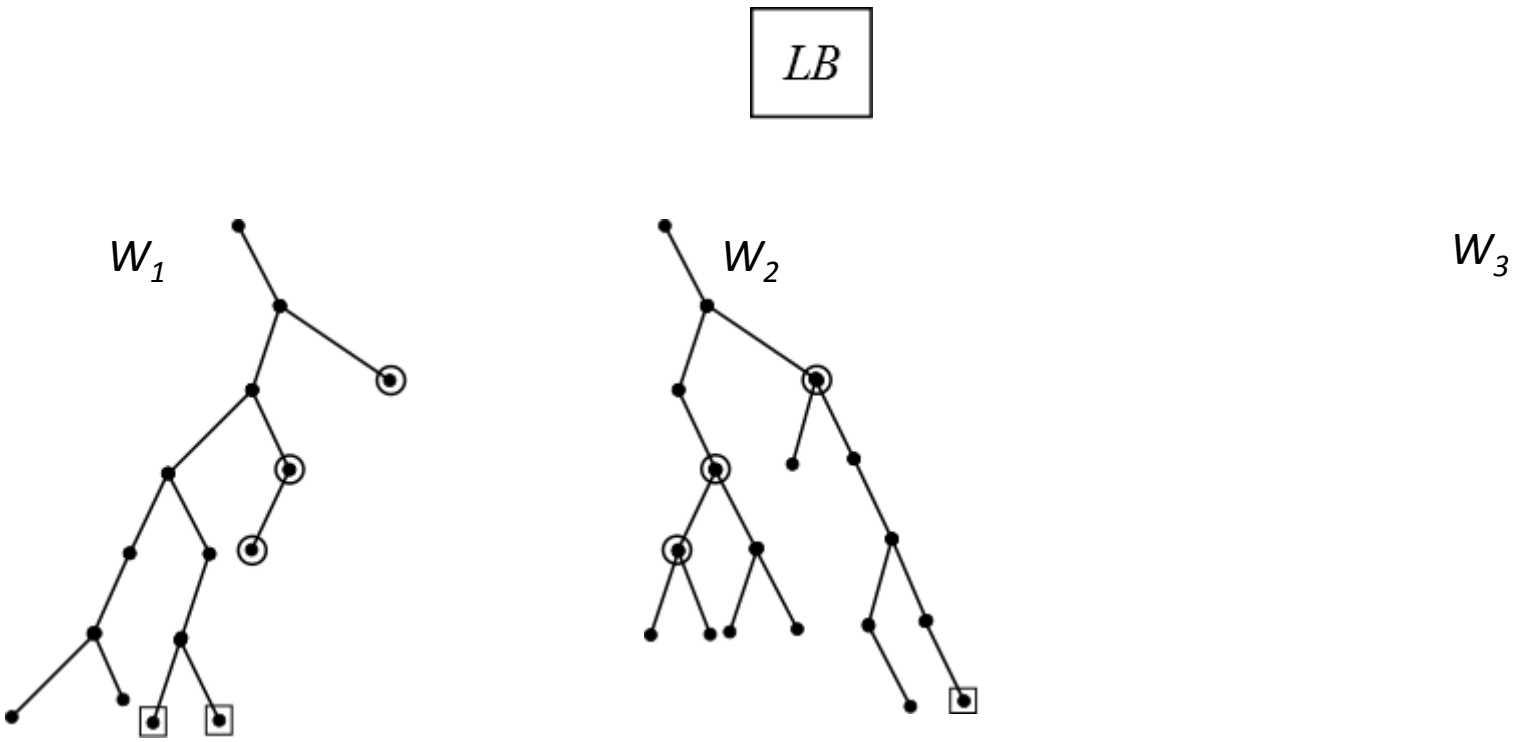
Parallel Symbolic Execution

W_1

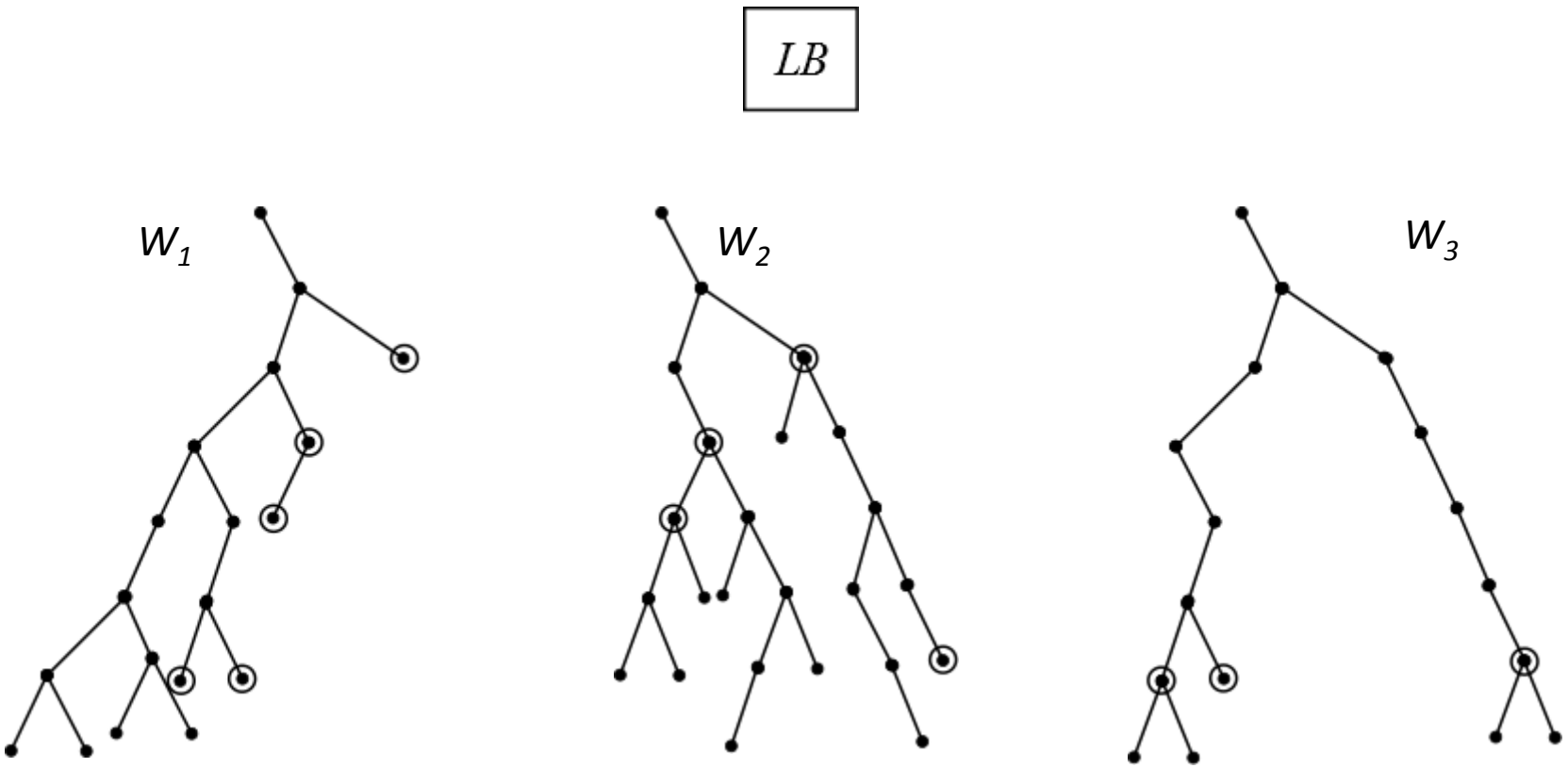


W_2

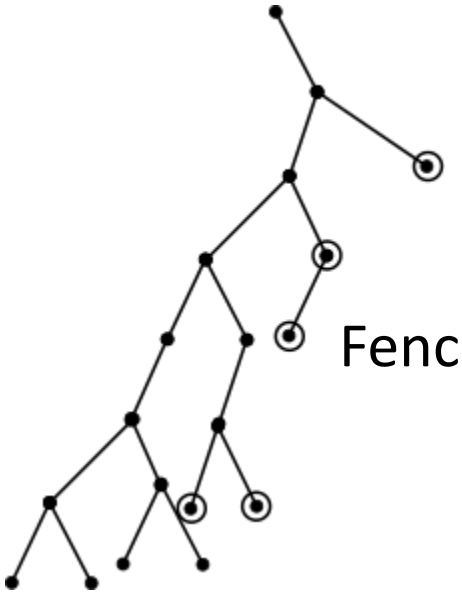
Parallel Symbolic Execution



Parallel Symbolic Execution



Worker-level Operation

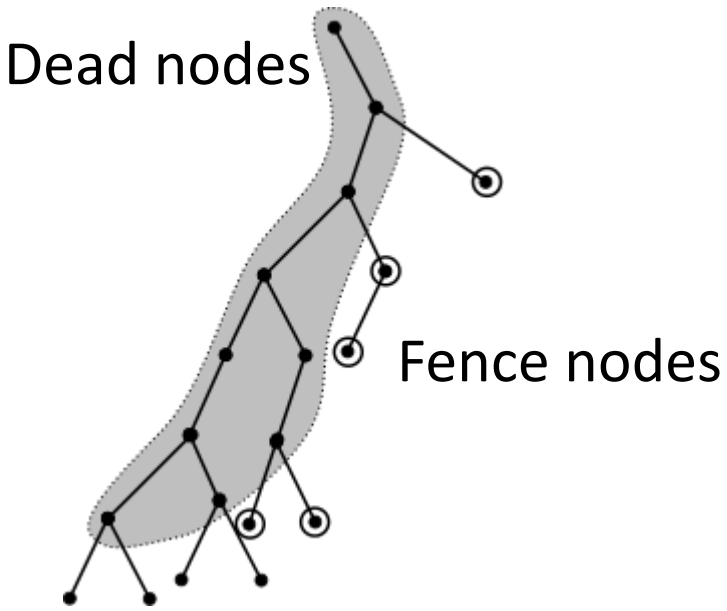


Fence nodes

Nodes:

- Fence – separating the domains of different workers

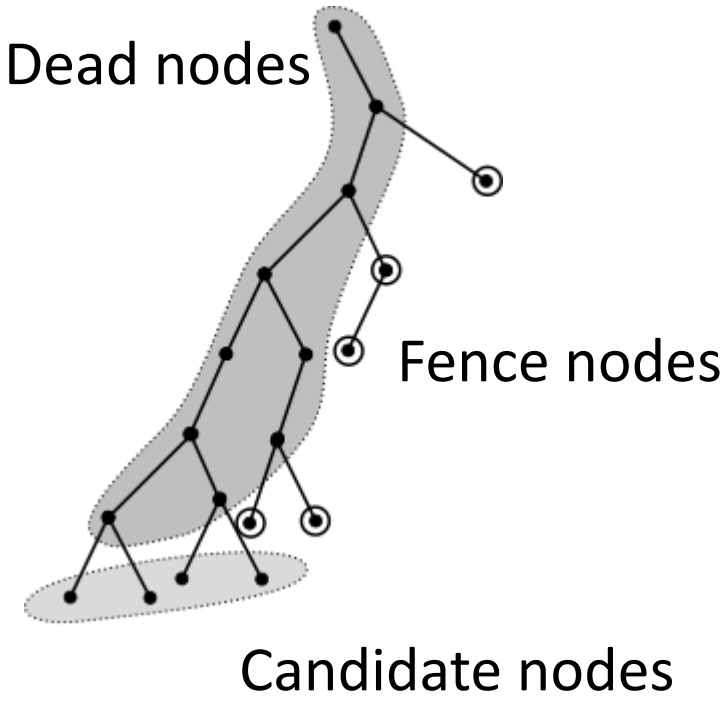
Worker-level Operation



Nodes:

- Fence – separating the domains of different workers
- Dead – have already been explored

Worker-level Operation



Nodes:

- Fence – separating the domains of different workers
- Dead – have already been explored
- Candidates – nodes ready to be explored. Candidates form the exploration frontier.

The POSIX Environment Model

- Goal: simulate behavior of a real execution environment
- Implemented model supports
  - Threads (!)
  - Process management
  - Sockets
  - Pipes
  - etc
- Only stateless or read-only system calls

Symbolic Test Suites

- Encompass many similar concrete test cases
- Programmatically control events in the environment
  - Symbolic data and streams
  - Network conditions
  - Fault injection
  - Symbolic scheduler

Cloud9 Use Cases

- Cloud9 goal is to bridge the gap between symbolic execution and real-world automated software testing
- Software systems analyzed
  - Coreutils
  - Curl
  - Memcached
  - Lighthttpd
  - Bandicot DBMS

Summary

S^2E

- Rapid prototyping of system analysis tools

Cloud9

- Bridge the gap between symbolic execution and real-world automated testing

Summary

S^2E

- Rapid prototyping of system analysis tools
- Performance, applicability

Cloud9

- Bridge the gap between symbolic execution and real-world automated testing
- Scalability, applicability

Summary

S^2E

- Rapid prototyping of system analysis tools
- Performance, applicability
- Selective symbolic execution

Cloud9

- Bridge the gap between symbolic execution and real-world automated testing
- Scalability, applicability
- Parallel symbolic execution

Summary

S^2E

- Rapid prototyping of system analysis tools
- Performance, applicability
- Selective symbolic execution
- Consistency models

Cloud9

- Bridge the gap between symbolic execution and real-world automated testing
- Scalability, applicability
- Parallel symbolic execution
- POSIX Environment model

Summary

S^2E

- Rapid prototyping of system analysis tools
- Performance, applicability
- Selective symbolic execution
- Consistency models
- Uses KLEE SEE

Cloud9

- Bridge the gap between symbolic execution and real-world automated testing
- Scalability, applicability
- Parallel symbolic execution
- POSIX Environment model
- Uses KLEE SEE

Questions

- Is it possible to combine two presented approaches? Is it a good idea? (7)
- Is Cloud9 really scalable? (7)
  - 1000 nodes?
- Why do we have superlinear speedup? (6)
- What metric to use to define the usability? (6)
- What are the core advantages of S2E? (5)
  - Actually useful in practice vs theory?
- Contrast the ways S2E and Cloud9 handle the env. (5)
- Can virtualization hide bugs? Can it cause problems? (5)
- Difference in input formats? How input format influence the performance, applicability, usability? (4)
- How to take advantage of having more hardware? (2)