Network Verification and Synthesis: Lessons from Hardware (and Software) Verification and Synthesis

Sharad Malik

Cornell-Princeton Center for Network Programming

6/20/2016

Need Strong Practical Motivation

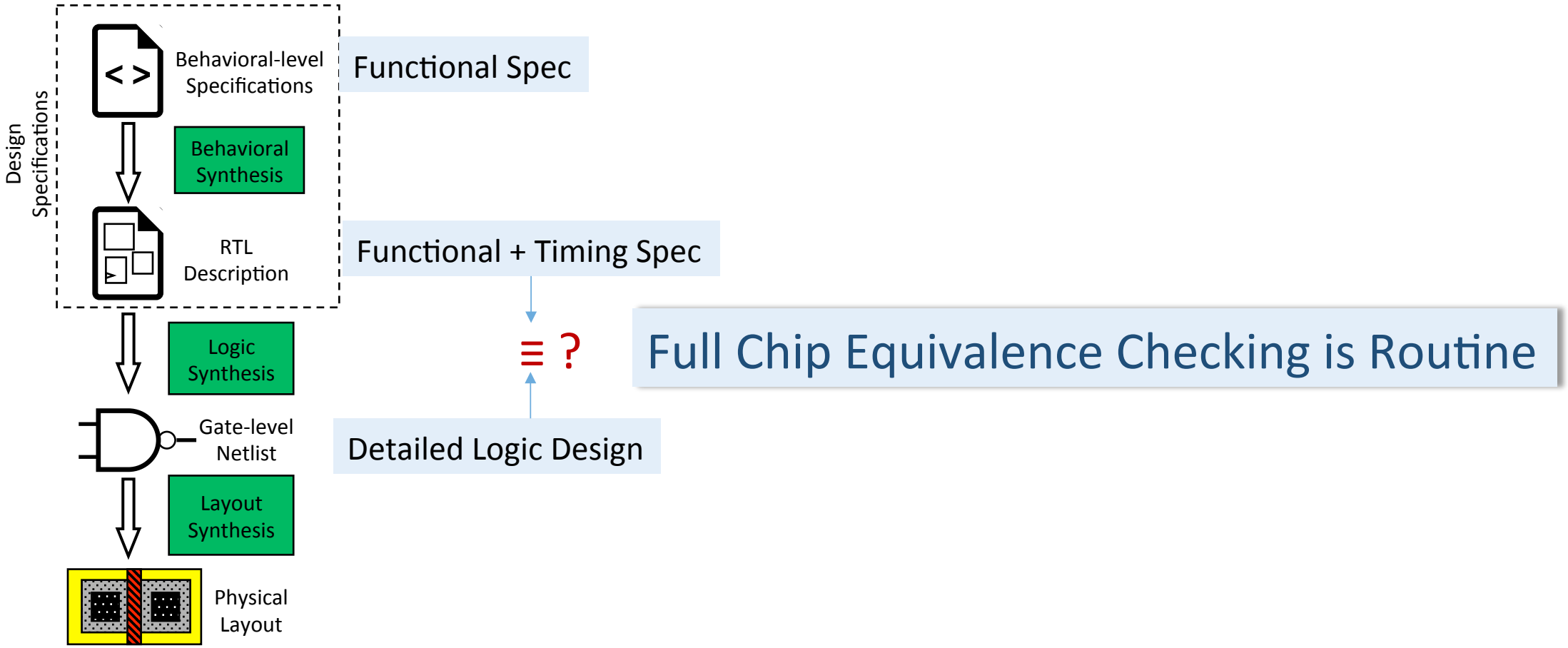
High cost of failure

- Need for first silicon success
  - High mask costs
- Product recalls
  - Intel Pentium FDIV Bug 1994
  - Total cost: $475 million

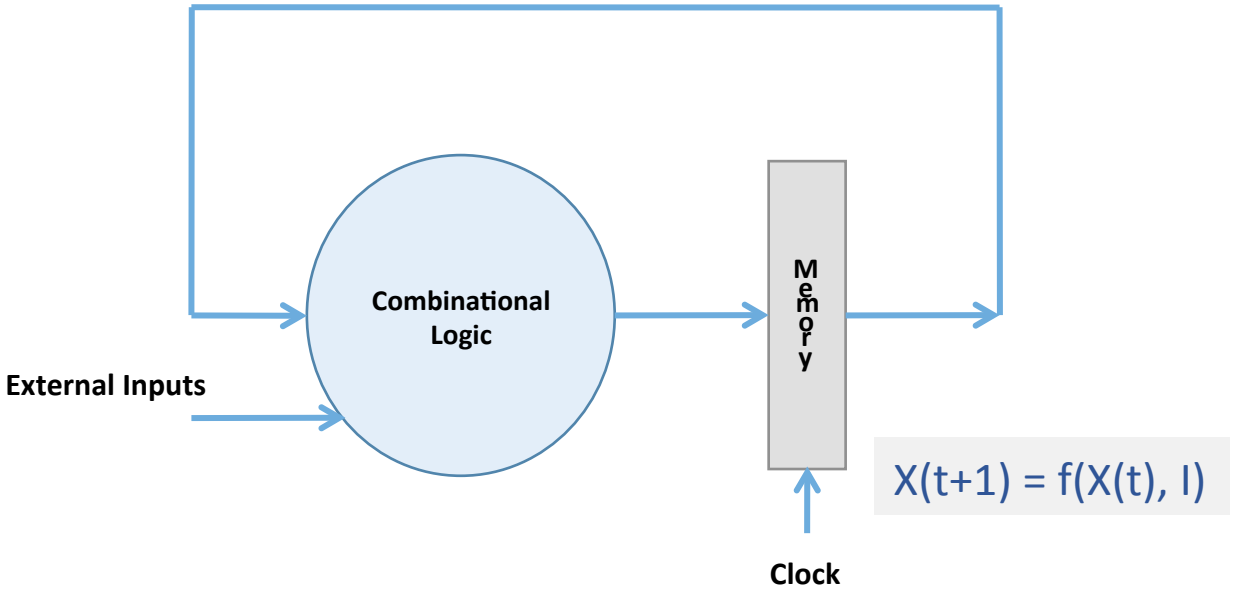
$$\frac{4195835}{3145727} = 1.333820449136241002$$

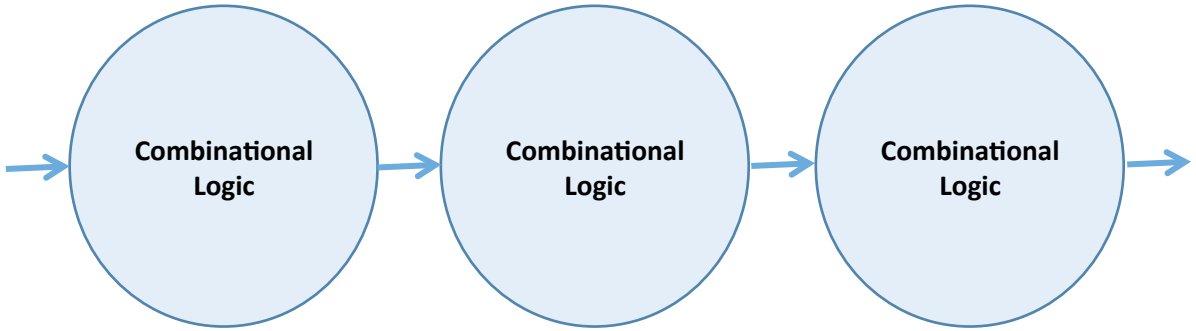
Down time and security breach costs compelling for Network Verification

Scalability is Key



Watch the complexity barrier: PSPACE-complete vs. NP-complete





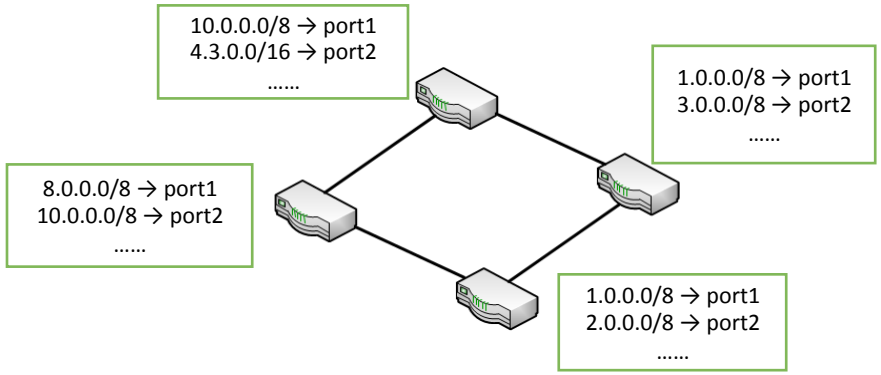
k-cycle verification

Model Checking
State space exploration:
Need to store sets of states

Bounded Model Checking
Propositional Logic, SAT based analysis:
Search, but no state storage

Snapshot Verification

- Verify the static network state
  - A snapshot of a dynamic system
  - A single SDN rule configuration
  - No performance verification

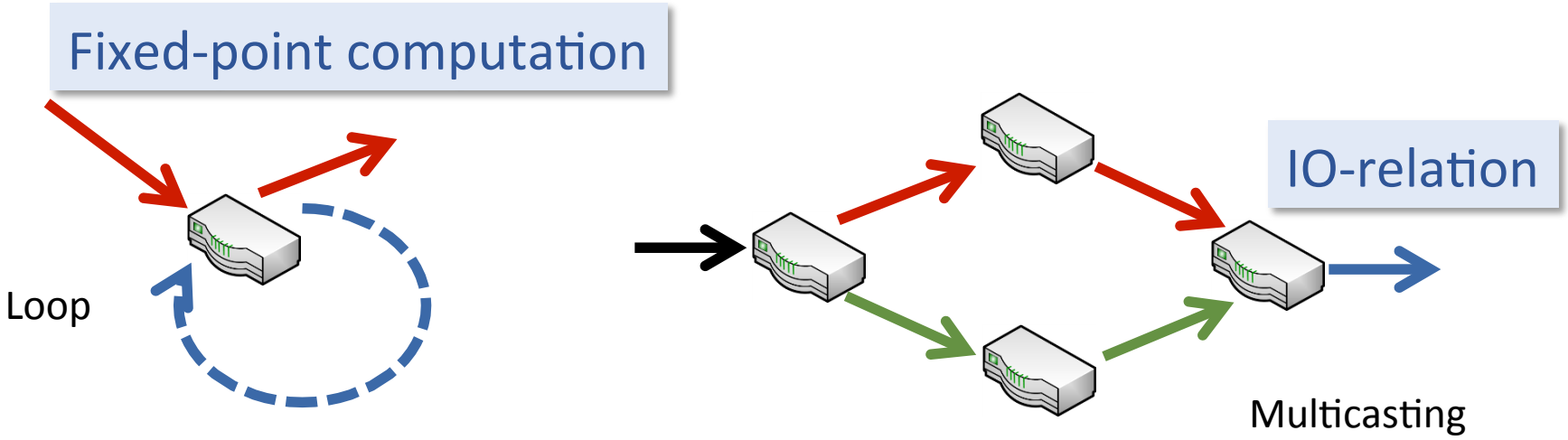


- Network state change (rule deletion/addition/change at a switch)[1]
  - Tens of events per second
- Packet arrival rate
  - Millions of arrivals per second

[1] Gude, N., Koponen, T., Pettit, J., Pfa, B., Casado, M., McKeown, N., Shenker, S.: “Nox: towards an operating system for networks,” SIGCOMM 2008

Modeling/Analysis Challenge

- Even for a single packet entering a network, a link may see multiple packets

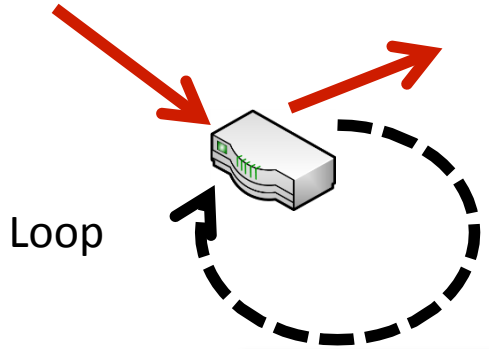


- Switch output not a combinational function of its inputs

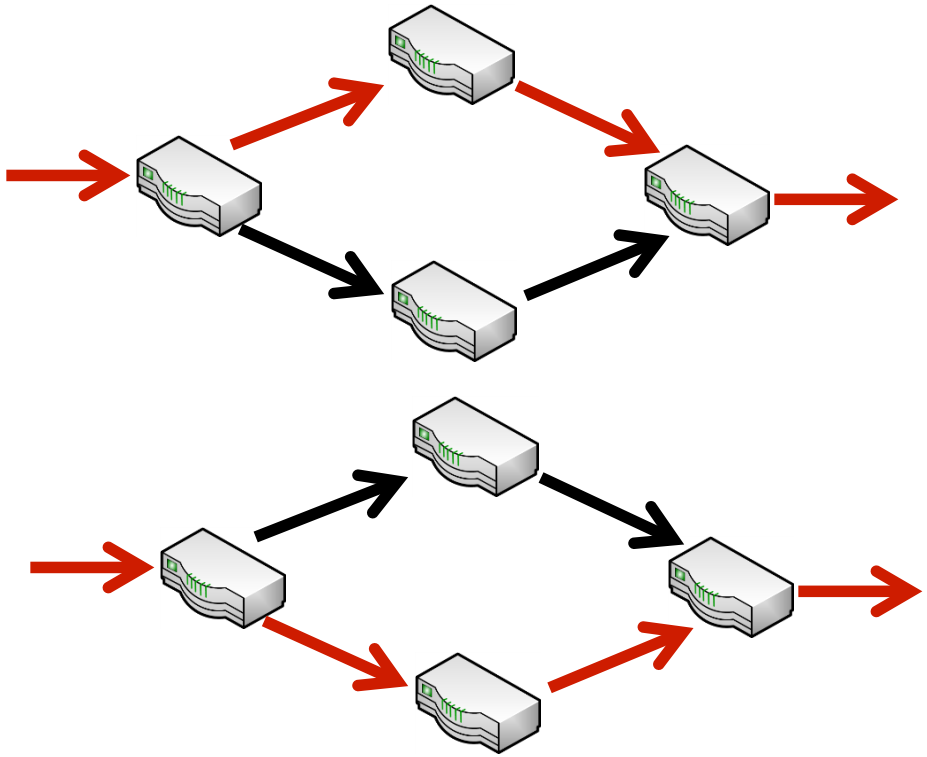
Need to store sets of values

Adapting Modeling/Analysis

- Limit packet flow to a *single path for a single packet* through the network



Loops implicitly blocked



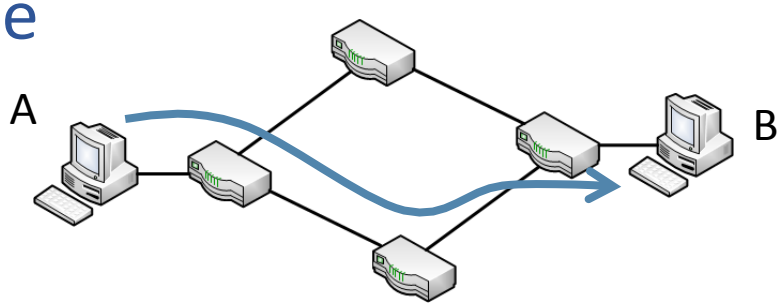
- Captures only part of the network behavior
- What good is this?

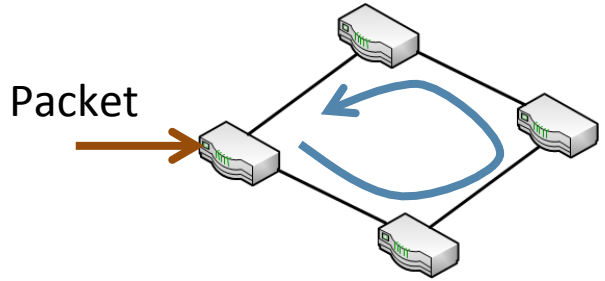
Goal: Counterexamples for Property Failures

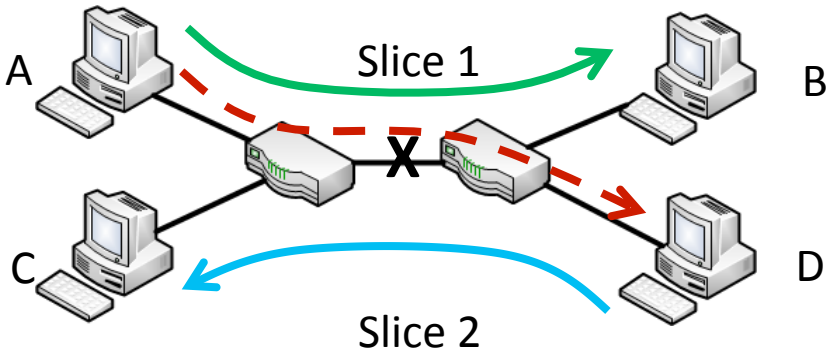
Single Path Single Packet Counterexample

Suffices for

- Functional Properties:
  - Reachability checking
    - Waypointing
    - Blacklisting
- Functional/Performance Properties:
  - Forwarding loop
- Security Properties:
  - Slice isolation
    - virtualization context







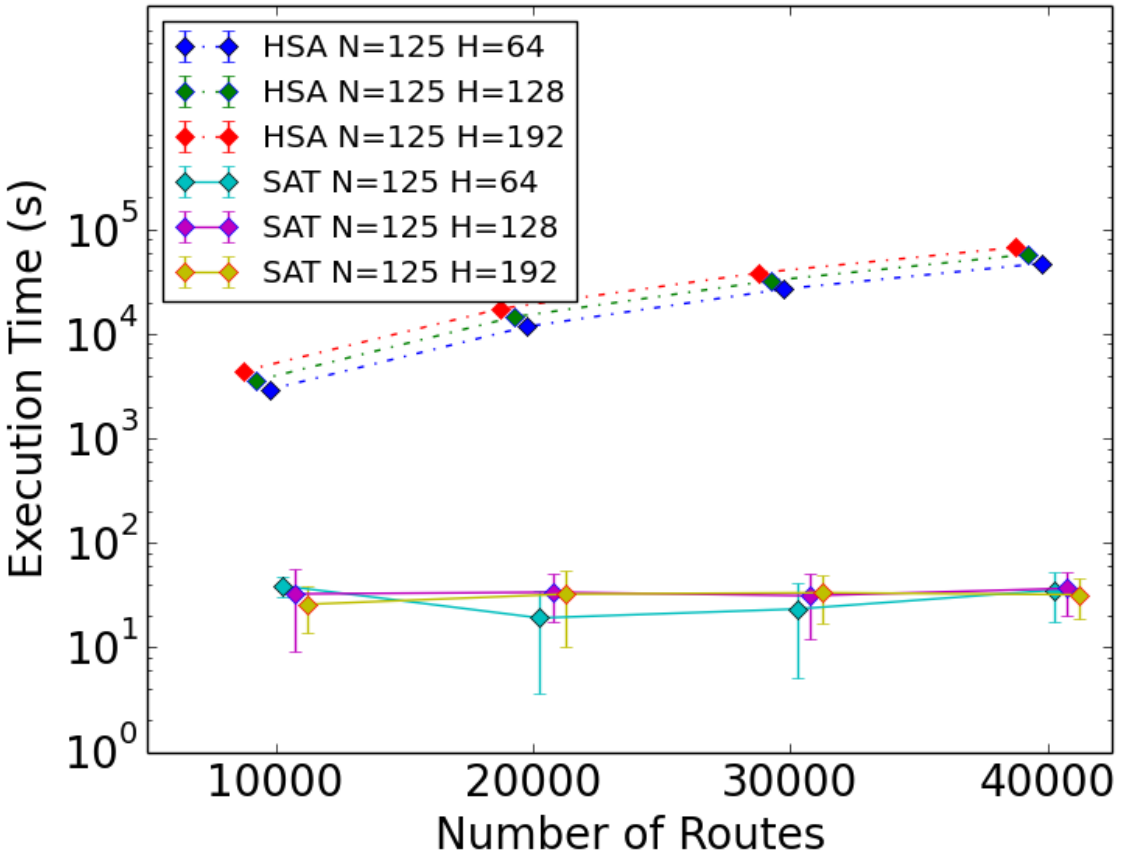
Evaluation

Setup

- SAT solver: Minisat
- Stanford backbone network
  - 16 routers with full network functions (VLAN, ACL, …)
  - ≈ 15,000 rules
  - 129 seconds to find a forwarding loop
    - Header Space Analysis (HSA): 758 seconds
      - Uses Ternary Symbolic Simulation
- Synthetic benchmarks for scalability experiments
  - Fat tree topology
  - Shortest path routing
  - Depth-first-search to generate matching rules
  - Vary
    - # of switches: N
    - # of routes: P
    - # of packet header bits: H

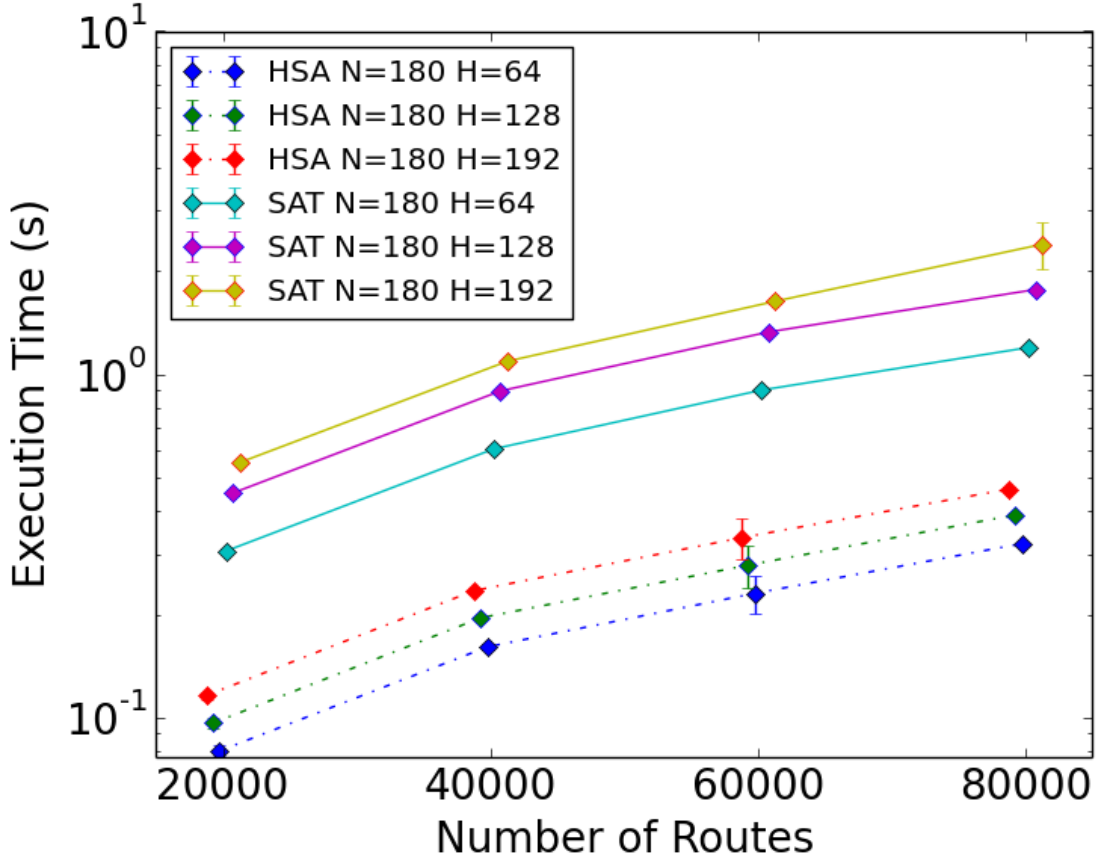
Evaluation

- Property
  - Forwarding loop check
- Setup
  - Vary
    - # Routes
    - # of Header bits
  - HSA: Header Space Analysis
  - SAT: SAT-based method
- Observations
  - Sub-exponential growth with number of routes
  - Low dependence on header size



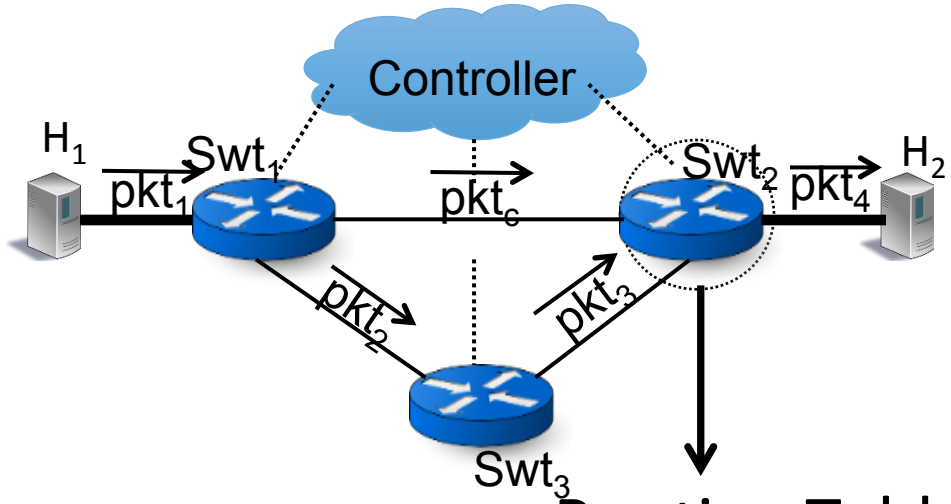
Evaluation

- Property
  - Reachability check
- Setup
  - Vary
    - # Routes
    - # of Header bits
  - HSA: Header Space Analysis
  - SAT: SAT-based method
- Observations
  - Sub-exponential growth with number of routes
  - Low dependence on header size



Small number of equivalence classes of packets

Controller Verification: Challenges



Routing Table

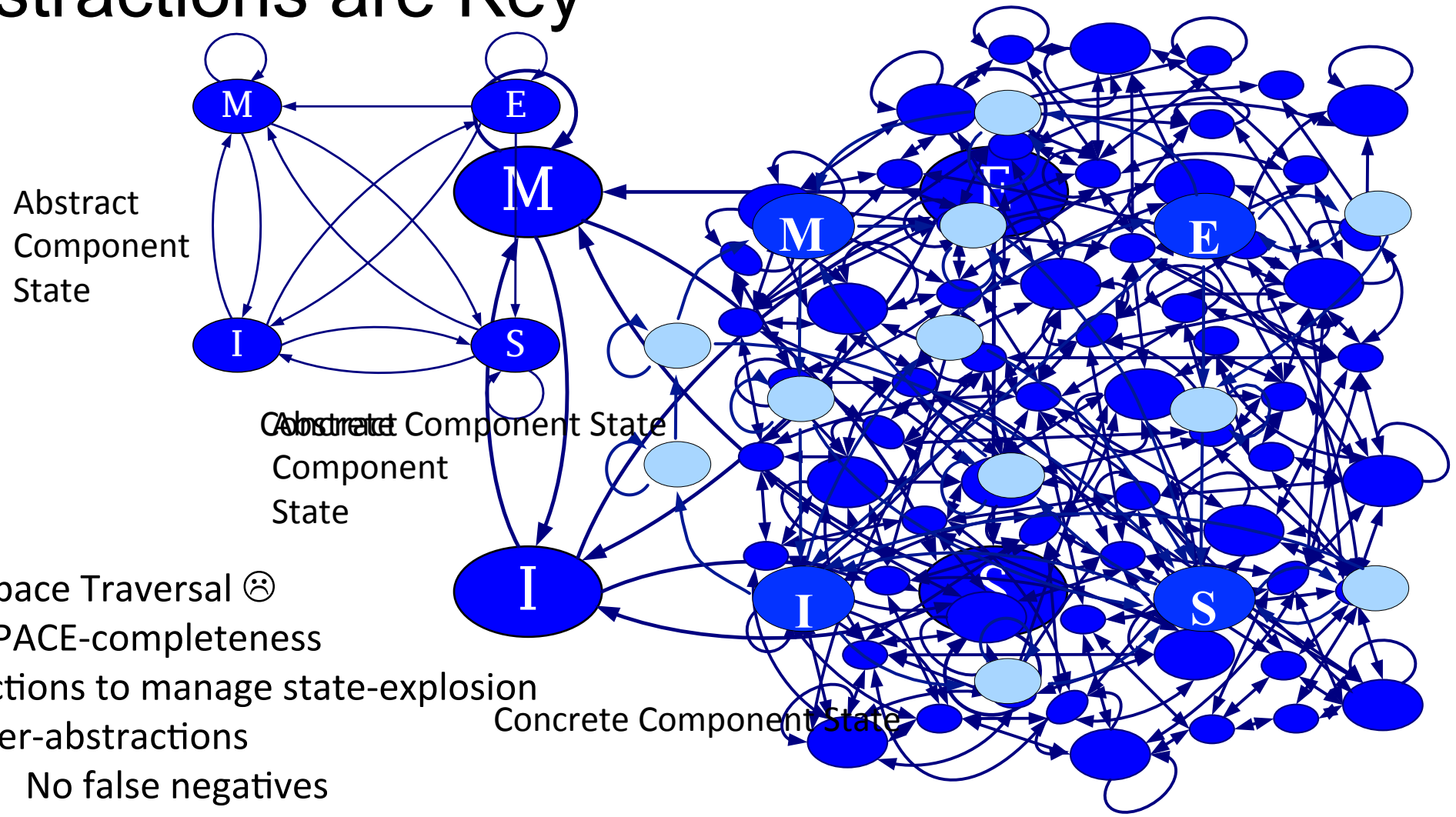
outPort(inPkt) =

$Port_1$: inPkt.src = $Host_1$
$Port_2$: inPkt.src = $Host_3$
$Port_3$: inPkt.src = $Host_k$

$Port_p$: inPkt.src = $Host_r$
$Port_q$: inPkt.src = $Host^a$

- Large number of packets alive in network
  - Large buffer state
  - Large interleaving state
- Large number of rules installed in switches
  - Large network state

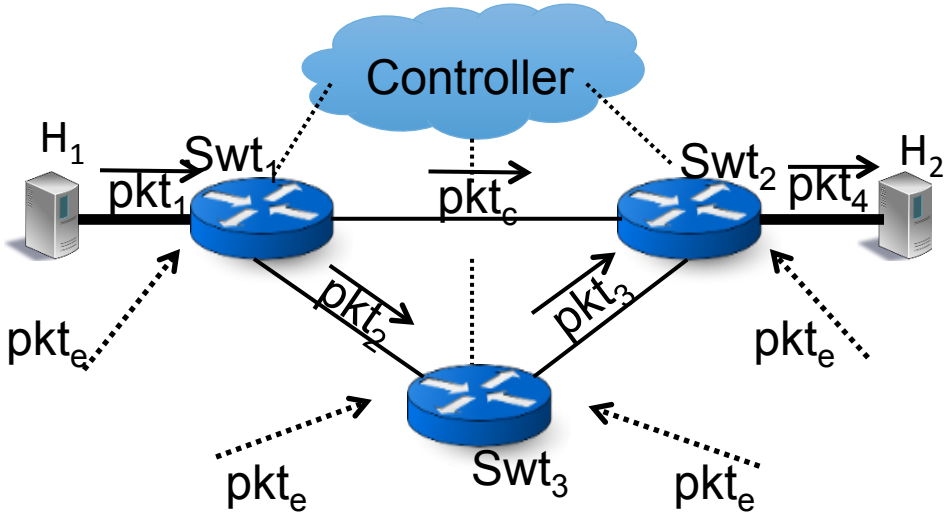
Abstractions are Key



- State Space Traversal ☹
  - PSPACE-completeness
- Abstractions to manage state-explosion
  - Over-abstractions
    - No false negatives

Figure Source: Valeria Bertacco

Abstraction: Handling Large Number of Packets



Environment packets (pkt_e) simulate the affect of an unbounded number of packets.

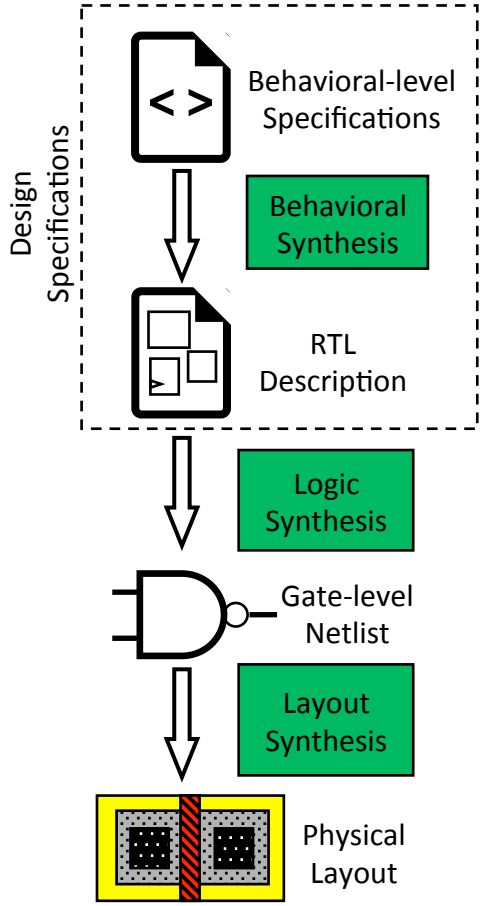
Evaluation

- Verified a learning switch
  - No packet gets into a loop in the network
- A buggy stateful firewall example
  - No source host gets unnecessarily blocked by the firewall
  - Detected known bug: a host did get blocked

D. Sethi, S. Narayana and S. Malik, “Abstractions for Model Checking SDN Controllers,” FMCAD 2013

Synthesis

Hardware



- Compile-time optimization of circuits

S. Zhang, F. Ivancic, C. Lumezanu, Y. Yuan, A. Gupta and S. Malik, "An Adaptable Rule Placement for Software-Defined Networks," *2014 DSN*

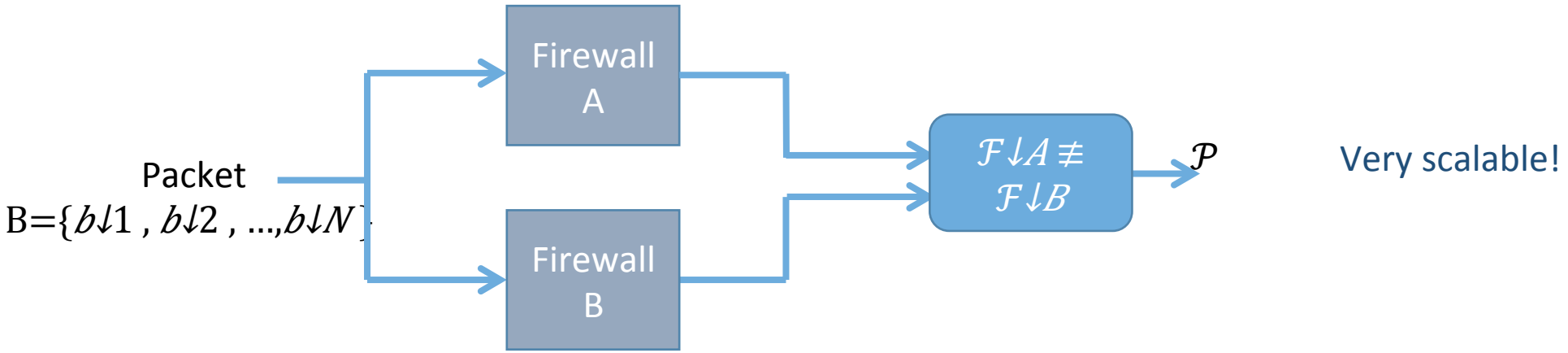
Software

- Program sketching [ASPLOS'06, ICSE'10, …]
  - Fill in program holes

S. Narain, G. Levin, S. Malik, V. Kaul, "Declarative Infrastructure Configuration Synthesis and Debugging," *2008, Journal of Network and Systems Management*

From Verification to Synthesis: Firewall Case Study

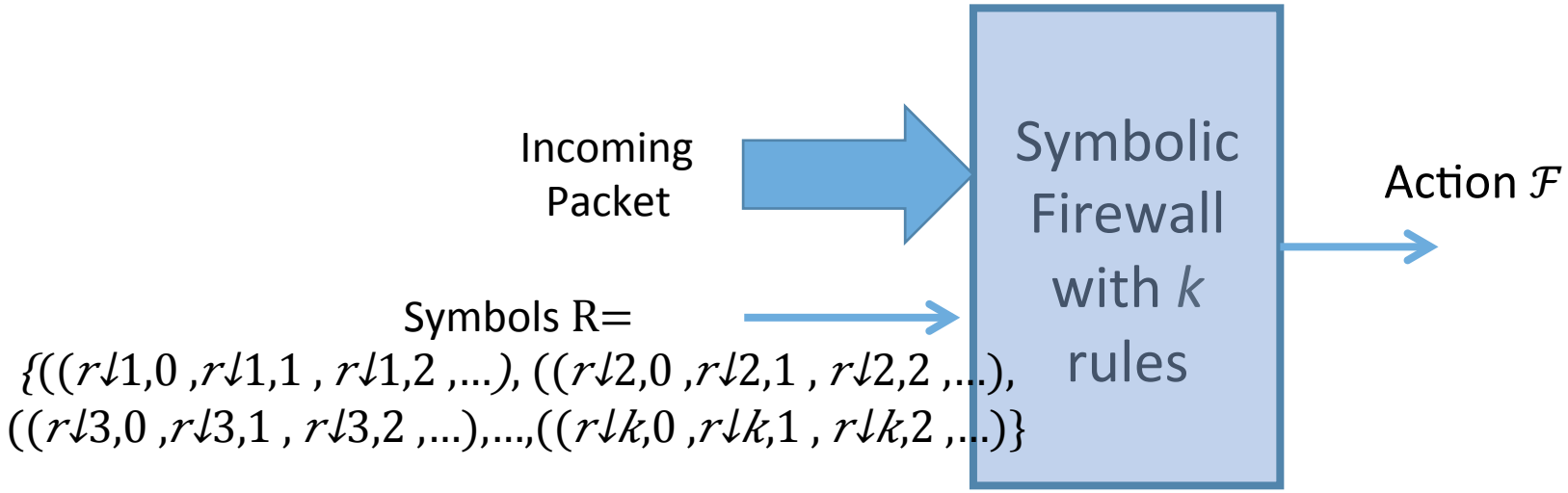
- Firewall Equivalence Checking
  - $\mathcal{P}=\mathcal{F}\downarrow A \not\equiv \mathcal{F}\downarrow B$
    - $\mathcal{P}$ satisfiable → not equivalent
    - $\mathcal{P}$ unsatisfiable → equivalent



Very scalable!

Firewall Synthesis

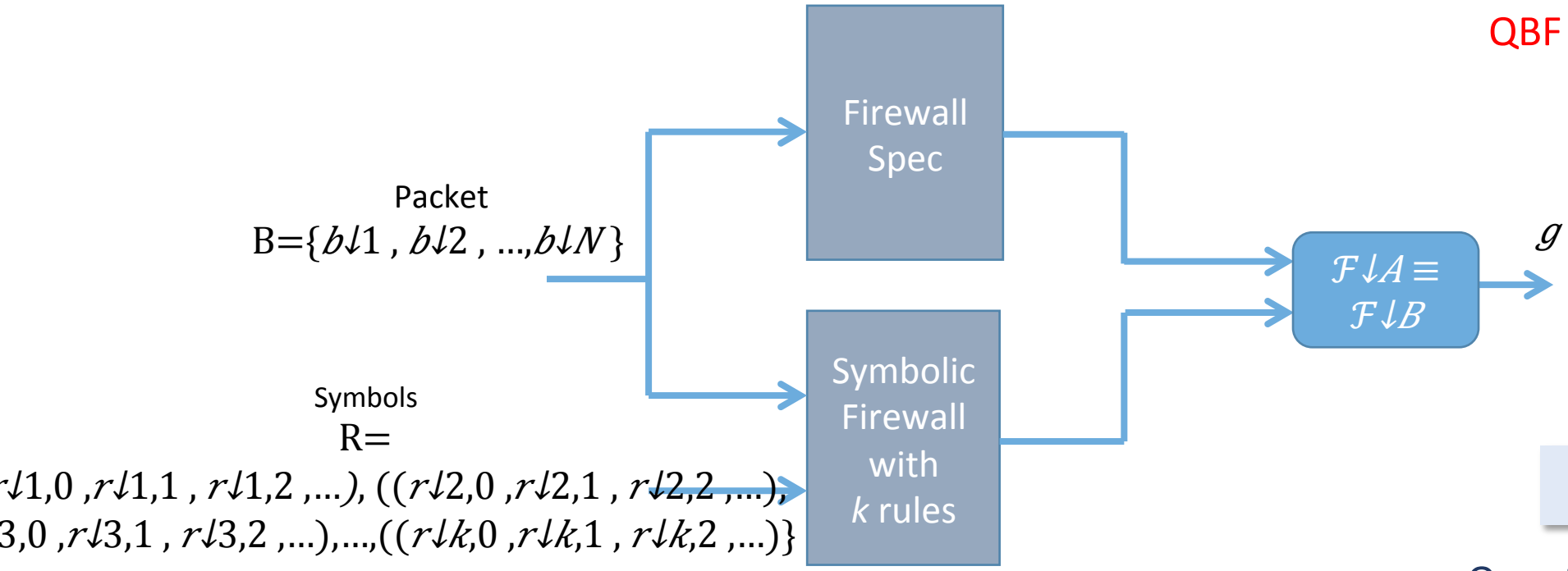
- Firewall Synthesis
  - Firewall with the fewest rules for a given specification
- Symbolic Firewalls
  - Represents all firewalls with *k* rules



- Each assignment to R specifies one firewall

Firewall Synthesis

Watch the complexity barrier!
QBF is PSPACE-complete



$\exists R \forall B \ (g)$

Quantified Boolean Formula (QBF)

Similar to program sketching

- Find an R, if one exists, such that for all B, g holds
- Binary search for minimum k
- Practical QBF (and special purpose) solvers do not scale well

Summary

Verification

- Scalability barriers
  - NP-complete vs. PSPACE-complete
- Implementation verification is invaluable
- Abstractions are key

Synthesis

- Compile-time optimization opportunities
- Patch in holes
  - Debugging
    - Fixing configuration files
  - Network updates
- Large-scale synthesis?