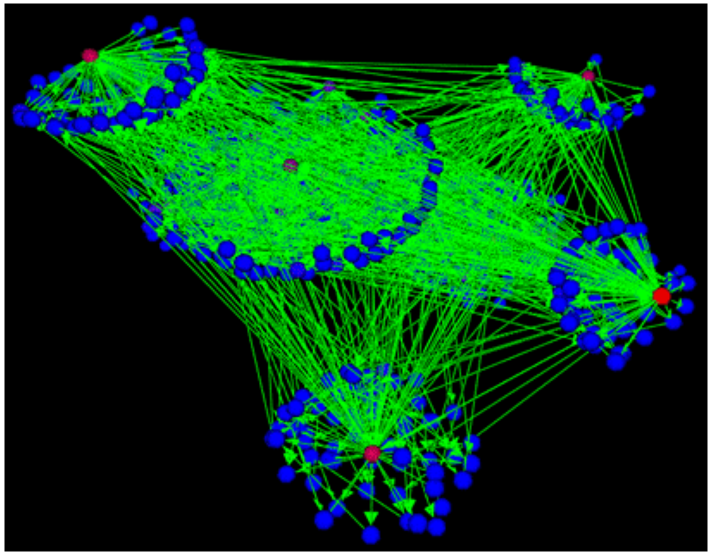
PL, meet Networking

Colin Scott, Andreas Wundsam, Scott Shenker

$\vdash \{A\}\ c\ \{B\}$



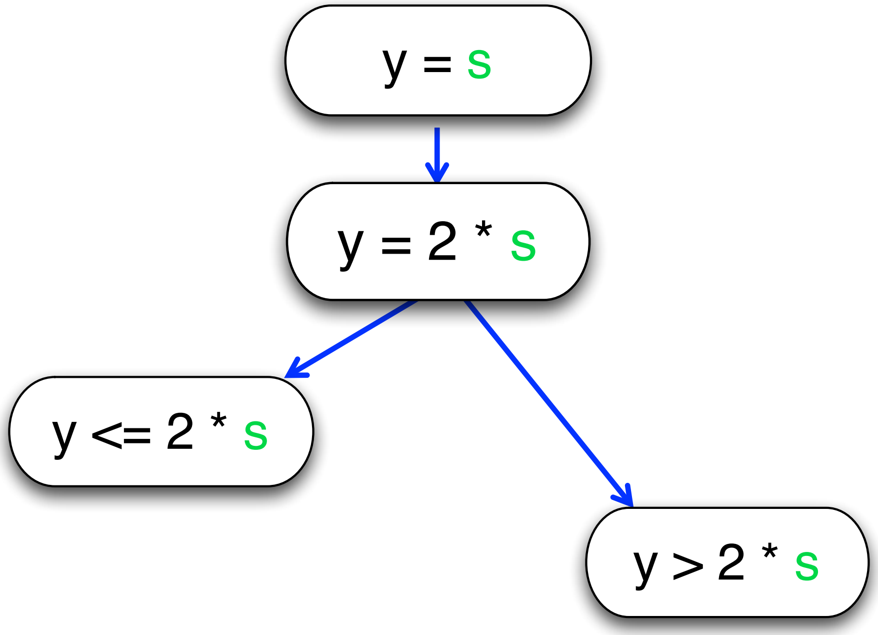
Overview

- Symbolic execution -> detect policy-violations
- Axiomatic semantics -> prove correctness

Symbolic Execution

- Goal: Which **code path** will be taken for a given input?

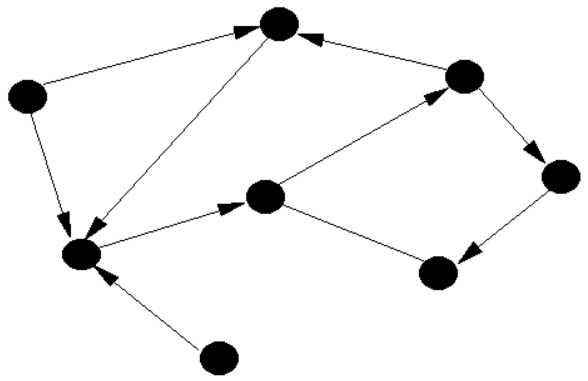
```
y = read()
y = 2 * y
if (y <= 12)
   fail()
else
   print("OK")
```


$$\begin{tikzcd}
& y = s \arrow[d] & \\
& y = 2 * s \arrow[dl] \arrow[dr] & \\
y <= 2 * s & & y > 2 * s
\end{tikzcd}$$

Formalism for Networking

Networks:

$$G = (V, E)$$



Packets:

$$h \in \{0,1\}^L = H$$

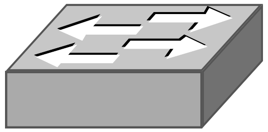
0110110101101
1001101110110
10011110001…

* Peyman et al., Header Space Analysis, NSDI '12

Formalism for Networking

Routers:

$$T:(H \times E) \rightarrow (H \times E_{\phi})$$



* Peyman et al., Header Space Analysis, NSDI '12

Formalism for Networking

Configuration:

$$\Psi = \{ \; T_1 \; \cdots \; T_n$$

Packet Forwarding:

$$\Psi^k(h,e) = \Psi(\ldots\Psi(\Psi(h,e))\ldots)$$

* Peyman et al., Header Space Analysis, NSDI '12

Symbolic Execution For Networking

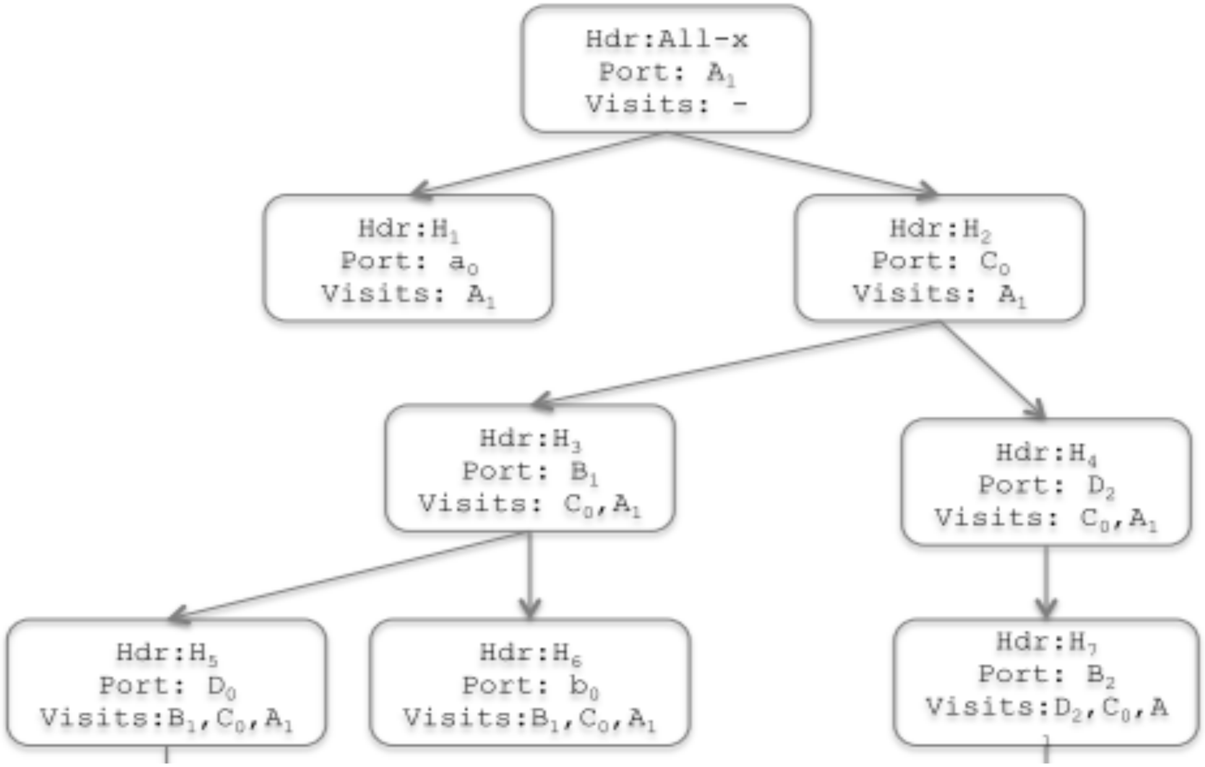
- Goal: Which **network path** will be taken for a given input?

Symbolic Execution For Networking

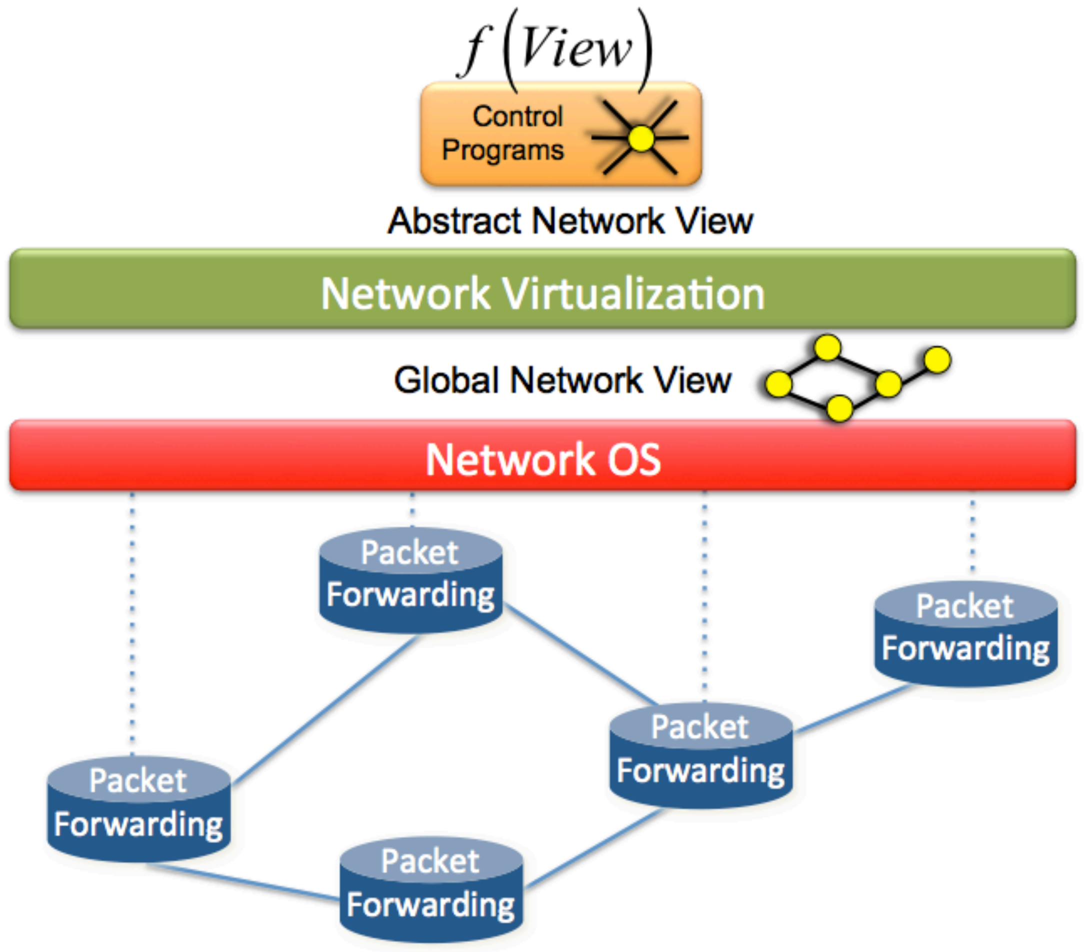
- Compute Ψ from routing tables
- For each host:
  - Insert symbolic packet x^L
  - Iteratively apply Ψ

Symbolic Execution For Networking

- End Result: Propagation Graph

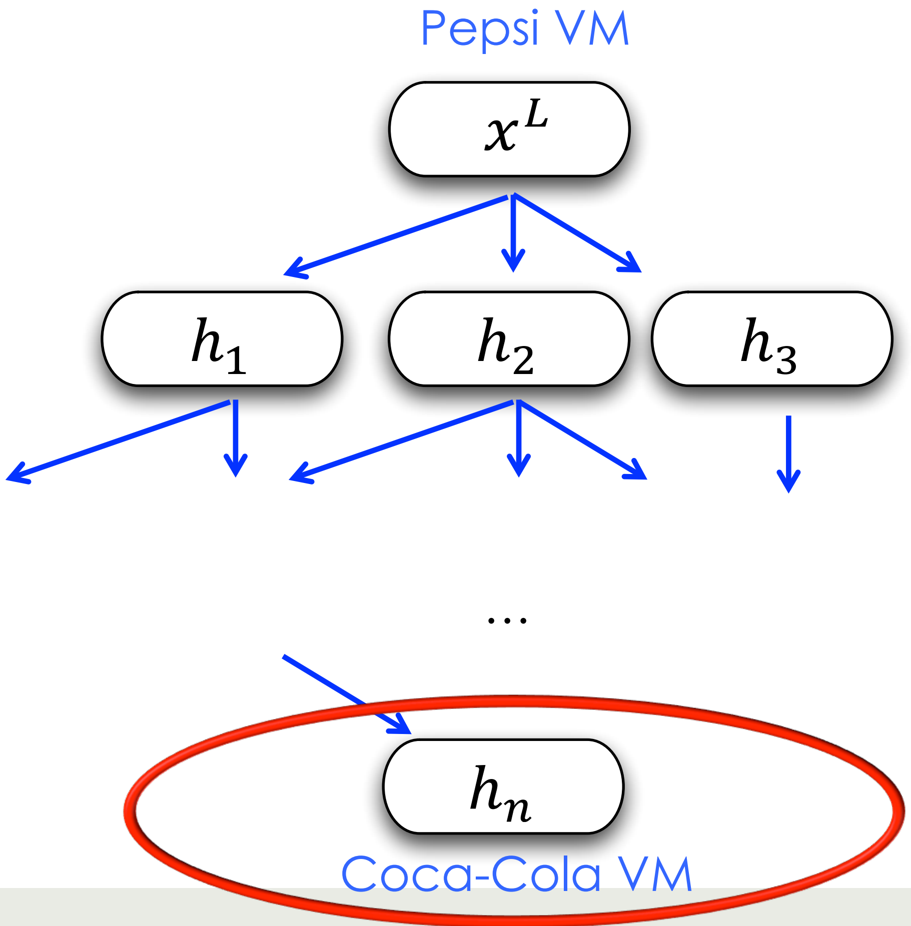


Brief Aside: Software-Defined Networking



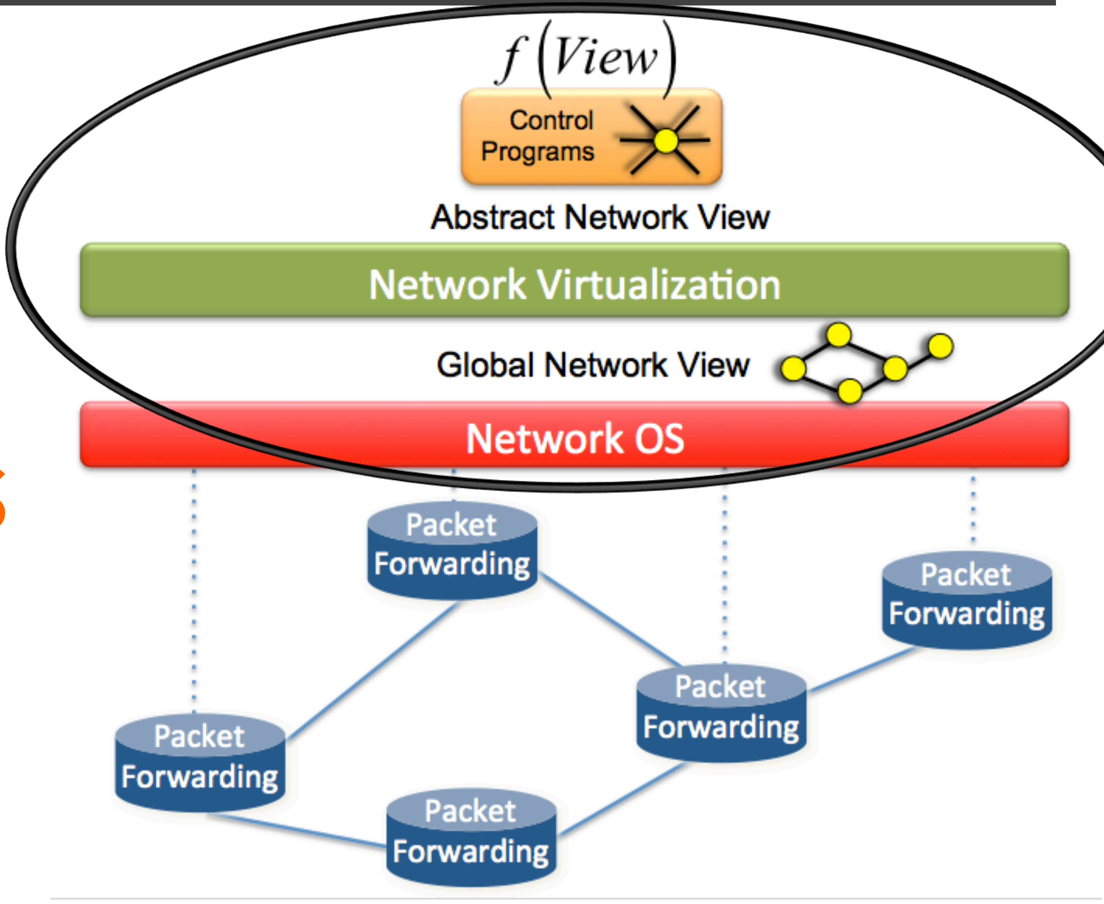
Example Policy-Violation

- "Pepsi can't talk to Coca-Cola"



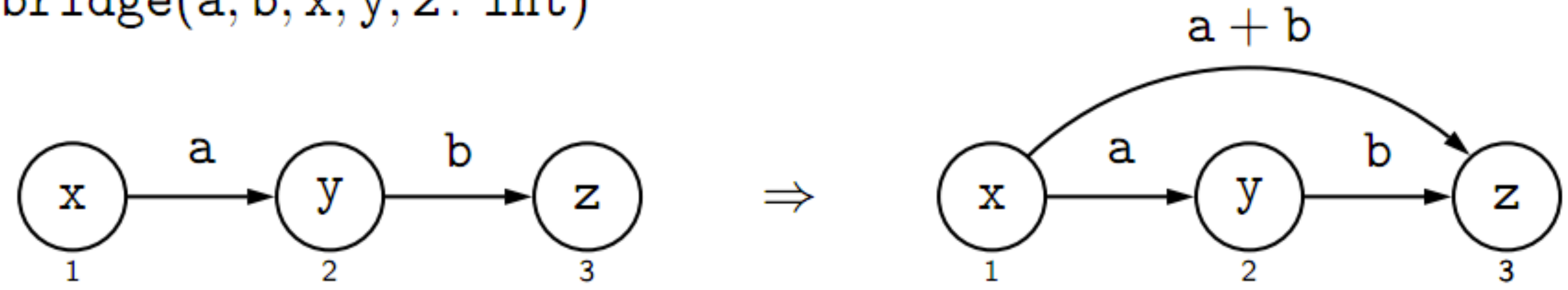
Proving Correctness

- Can we **prove** correctness of network software?



GP Graph Programming

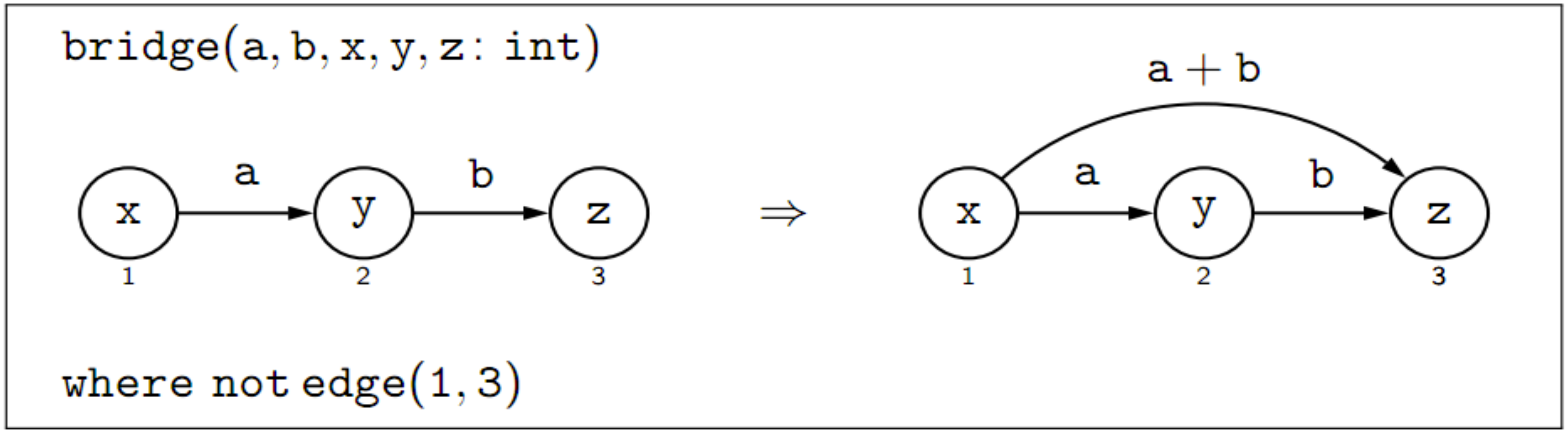
bridge(a, b, x, y, z: int)

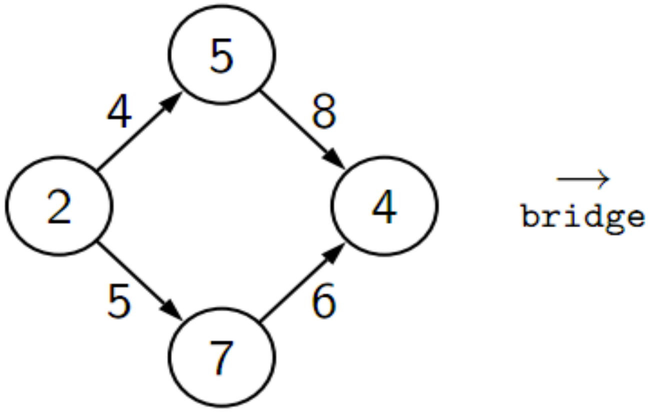


where not edge(1, 3)

* Poskitt, Plump, Hoare Style Verification of Graph Programs

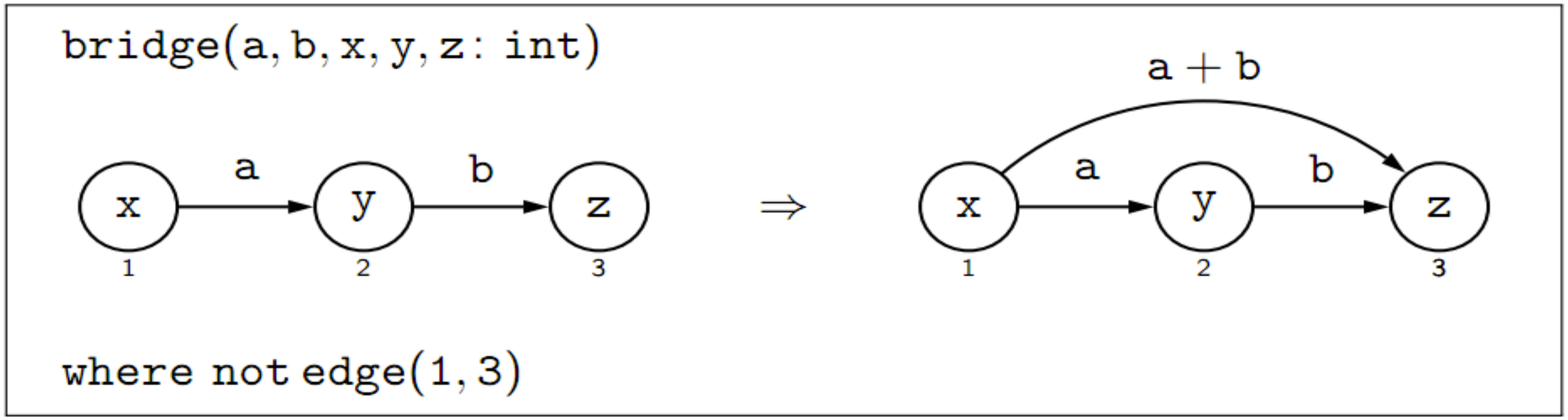
GP Graph Programming

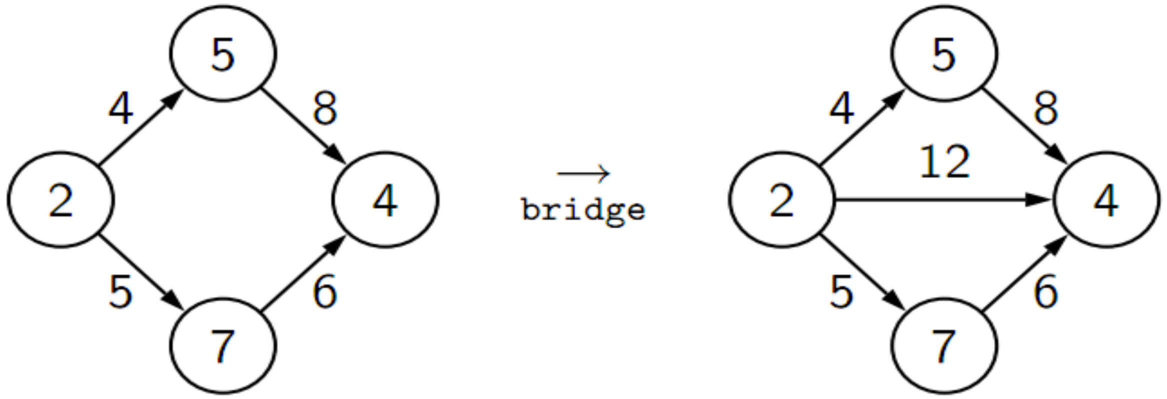




* Poskitt, Plump, Hoare Style Verification of Graph Programs

GP Graph Programming



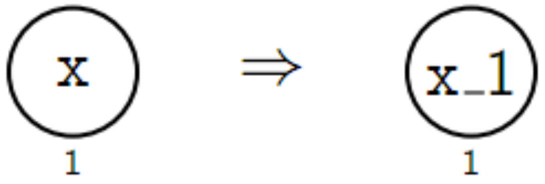


* Poskitt, Plump, Hoare Style Verification of Graph Programs

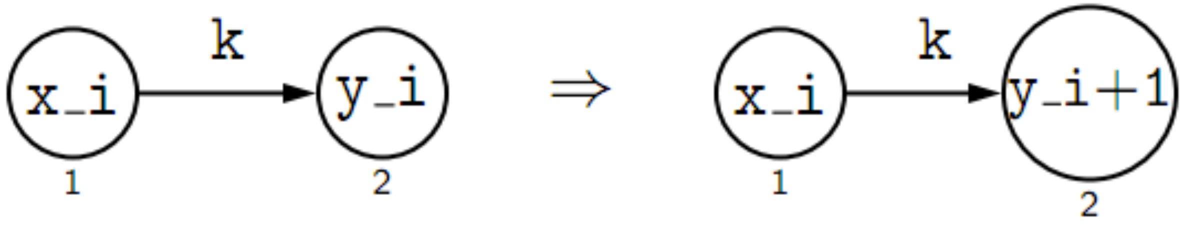
GP Graph Coloring

```
main = init!; inc!
```

```
init(x: int)
```



```
inc(i, k, x, y: int)
```



Proving Correctness

- Goal: Hoare-style axiomatic semantics to prove graph programs correct

Proving Correctness

- Assertions can range over anything (not just integers and booleans)

“there exists at least one non-looping edge”

- $\exists(\textcircled{x} \xrightarrow{k} \textcircled{y})$

Axiomatic Semantics

Sequential composition.

$$[\text{comp}]\ \frac{\{c\}\ P\ \{e\} \qquad \{e\}\ Q\ \{d\}}{\{c\}\ P;\ Q\ \{d\}}$$

Rule of consequence.

$$[\text{cons}]\ c \Longrightarrow c'\ \frac{\{c'\}\ P\ \{d'\}}{\{c\}\ P\ \{d\}}\ d' \Longrightarrow d$$

Axiomatic Semantics

As long as possible iteration.

$$[!]\ \frac{\{inv\}\ \mathcal{R}\ \{inv\}}{\{inv\}\ \mathcal{R}!\ \{inv \land \neg\mathsf{App}(\mathcal{R})\}}$$

Axiomatic Semantics

Rule schema application.

$$[\text{rule}]\ \frac{}{\{\mathsf{Pre}(r, c)\}\ r\ \{c\}}$$

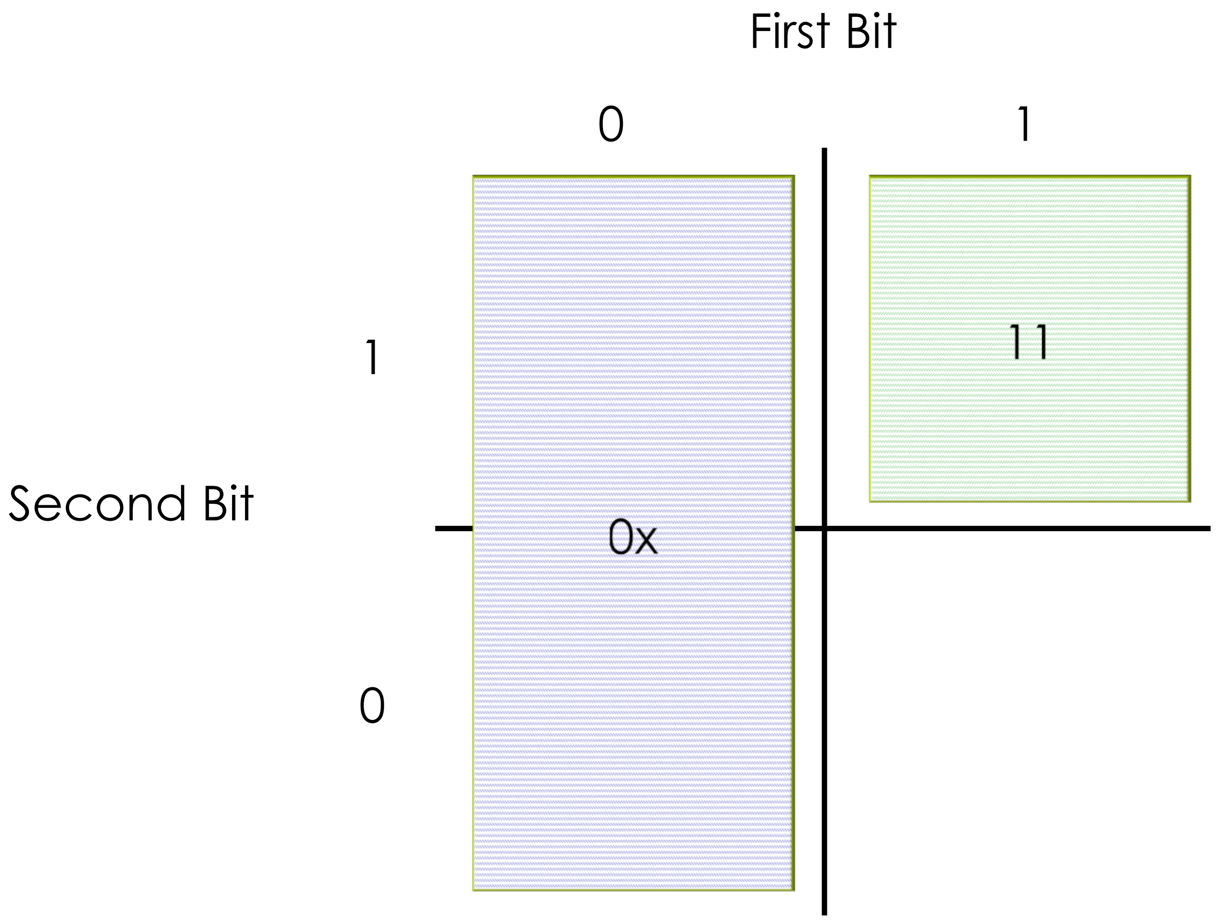
$$\text{[comp]}\dfrac{\text{[cons]}\dfrac{\text{[!]}\dfrac{\text{[cons]}\dfrac{\text{[rule]}\dfrac{}{\{\mathrm{Pre}(\texttt{init},e)\}\ \texttt{init}\ \{e\}}}{\{e\}\ \texttt{init}\ \{e\}}}{\{e\}\ \texttt{init!}\ \{e \wedge \neg\mathrm{App}(\{\texttt{init}\})\}}}{\{c\}\ \texttt{init!}\ \{d\}} \qquad \text{[!]}\dfrac{\text{[cons]}\dfrac{\text{[rule]}\dfrac{}{\{\mathrm{Pre}(\texttt{inc},d)\}\ \texttt{inc}\ \{d\}}}{\{d\}\ \texttt{inc}\ \{d\}}}{\{d\}\ \texttt{inc!}\ \{d \wedge \neg\mathrm{App}(\{\texttt{inc}\})\}}}{\{c\}\ \texttt{init!};\ \texttt{inc!}\ \{d \wedge \neg\mathrm{App}(\{\texttt{inc}\})\}}$$

$$\begin{aligned}
c &= \neg\exists(\ \textcircled{a}\ \mid \mathsf{type(a)} \neq \mathsf{int})\\
d &= \forall(\ \textcircled{a}\ , \exists(\ \textcircled{a}\ \mid \mathsf{a = b\_c} \wedge \mathsf{type(b,c) = int}))\\
e &= \forall(\ \textcircled{a}^{1}_{1}, \exists(\ \textcircled{a}^{1}_{1} \mid \mathsf{type(a) = int}) \vee \exists(\ \textcircled{a}_{1} \mid \mathsf{a = b\_c} \wedge \mathsf{type(b,c) = int}))\\
\neg\mathrm{App}(\{\texttt{init}\}) &= \neg\exists(\ \textcircled{x}\ \mid \mathsf{type(x) = int})\\
\neg\mathrm{App}(\{\texttt{inc}\}) &= \neg\exists(\ \textcircled{x\_i} \xrightarrow{k} \textcircled{y\_i} \mid \mathsf{type(i,k,x,y) = int})\\
\mathrm{Pre}(\texttt{init},e) &= \forall(\ \textcircled{x}_{1}\ \ \textcircled{a}_{2} \mid \mathsf{type(x) = int}, \exists(\ \textcircled{x}_{1}\ \ \textcircled{a}_{2} \mid \mathsf{type(a) = int})\\
&\quad \vee \exists(\ \textcircled{x}_{1}\ \ \textcircled{a}_{2} \mid \mathsf{a = b\_c} \wedge \mathsf{type(b,c) = int}))\\
\mathrm{Pre}(\texttt{inc},d) &= \forall(\ \textcircled{x\_i}_{1} \xrightarrow{k} \textcircled{y\_i}_{2}\ \ \textcircled{a}_{3} \mid \mathsf{type(i,k,x,y) = int},\\
&\quad \exists(\ \textcircled{x\_i}_{1} \xrightarrow{k} \textcircled{y\_i}_{2}\ \ \textcircled{a}_{3} \mid \mathsf{a = b\_c} \wedge \mathsf{type(b,c) = int}))
\end{aligned}$$

Summary

- PL techniques FTW!
- I adapted symbolic execution to find bugs in network software
- I demonstrated correctness proving for network algorithms

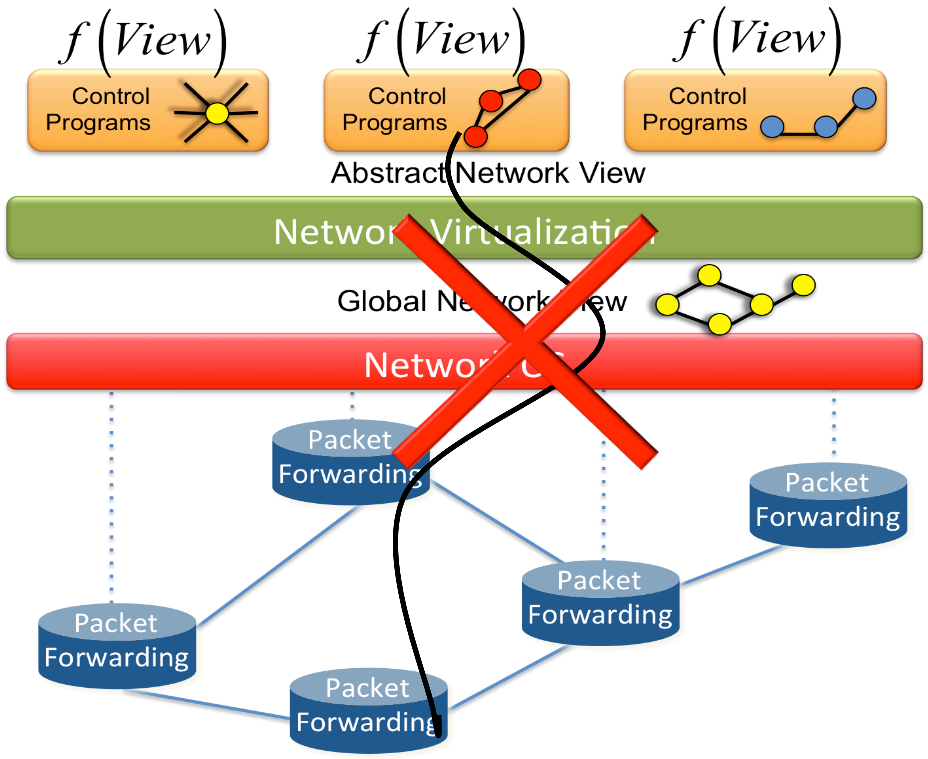
Symbolic Headerspace



Symbolic Execution For Networking

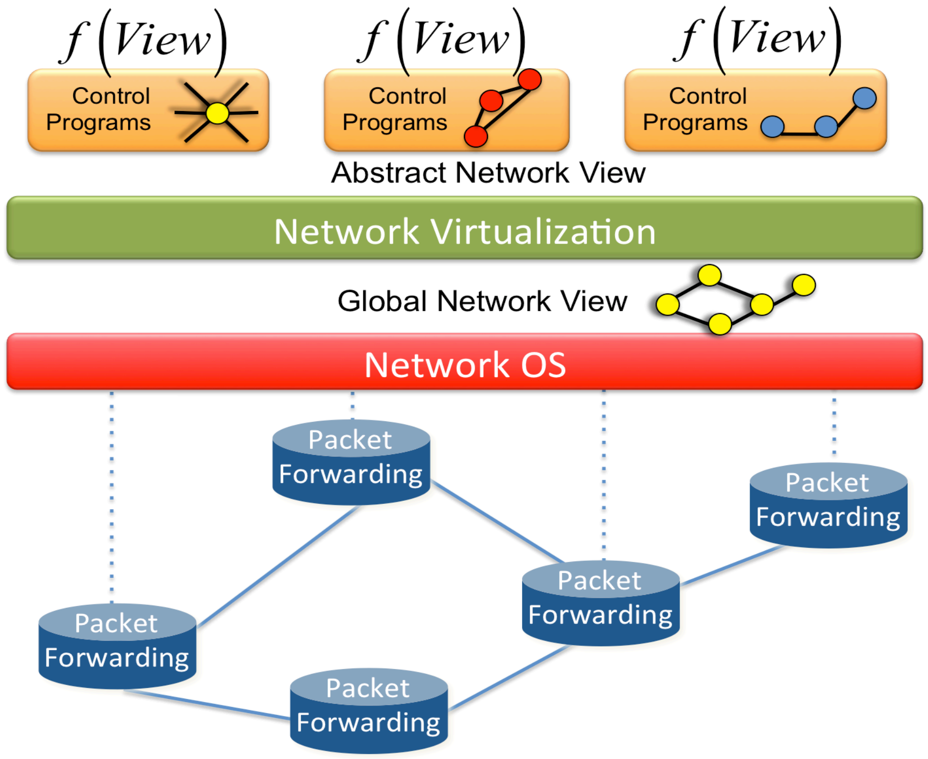
- Goal: detect **policy-violations**

- "Network doesn't do what I tell it to"



Symbolic Execution For Networking

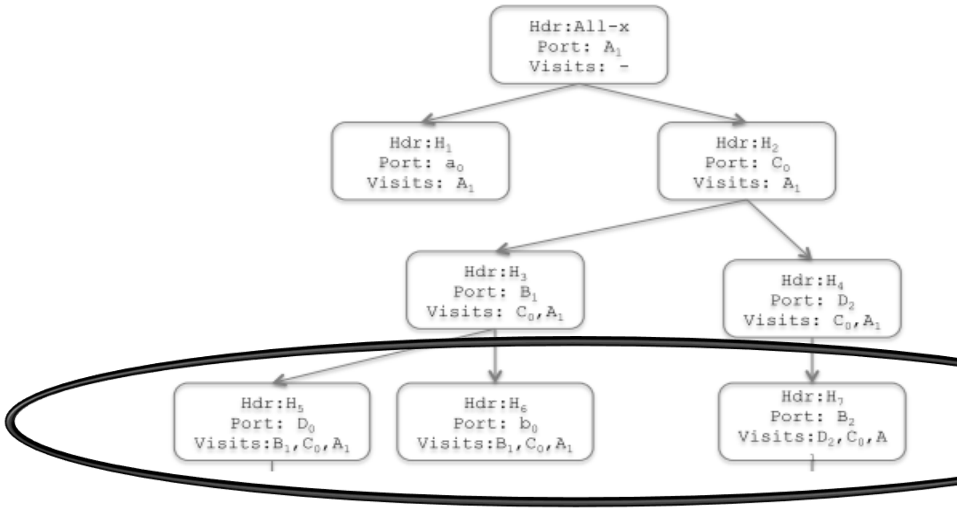
- Approach:
  - Demonstrate **isomorphism** between behavior of virtual view and physical network



Symbolic Execution For Networking

Network Behavior:

$$\Omega = \Phi^{\infty}$$



Hdr:All-x Port: A1 Visits: -
Hdr:H1 Port: a0 Visits: A1
Hdr:H2 Port: C0 Visits: A1
Hdr:H3 Port: B1 Visits: C0,A1
Hdr:H4 Port: D2 Visits: C0,A1
Hdr:H5 Port: D0 Visits:B1,C0,A1
Hdr:H6 Port: b0 Visits:B1,C0,A1
Hdr:H7 Port: B2 Visits:D2,C0,A

“Packets’ final locations”

Policy compliance

$$\Omega^{virtual} \sim \Omega^{physical}$$

"All paths in logical network should have a corresponding path in the physical network"

More GP constructs

- Sequential composition:
  P; Q
- If-then-else:
  if C then P else Q
- As-long-as-possible iteration:
  P!

Axiomatic Semantics

Set of rule schemata (none of which are applicable).

$$[\text{ruleset}_1]\ \frac{}{\{\neg \text{App}(\mathcal{R})\}\ \mathcal{R}\ \{\text{false}\}}$$

Set of rule schemata (when the non-applicability of a rule schema set is not implied by the precondition).

$$[\text{ruleset}_2]\ \frac{\{c\}\ r_1\ \{d\}\ \ldots\ \{c\}\ r_n\ \{d\}}{\{c\}\ \{r_1, \ldots, r_n\}\ \{d\}}$$