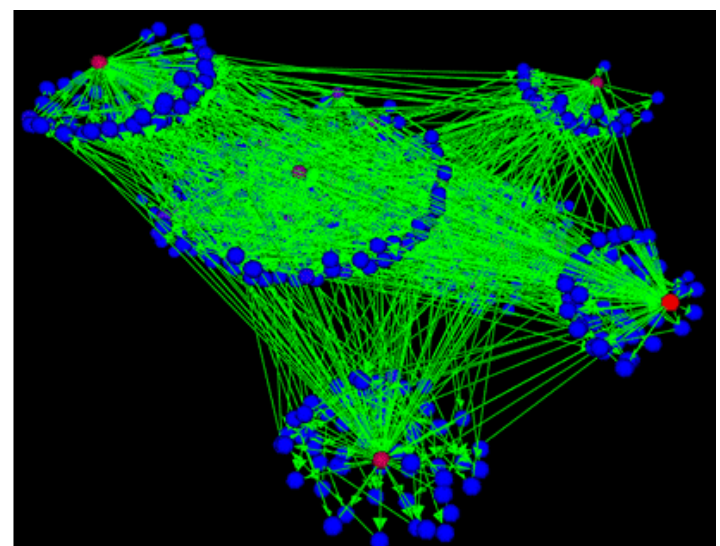
PL, meet Networking

Colin Scott, Andreas Wundsam, Scott Shenker

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



Overview

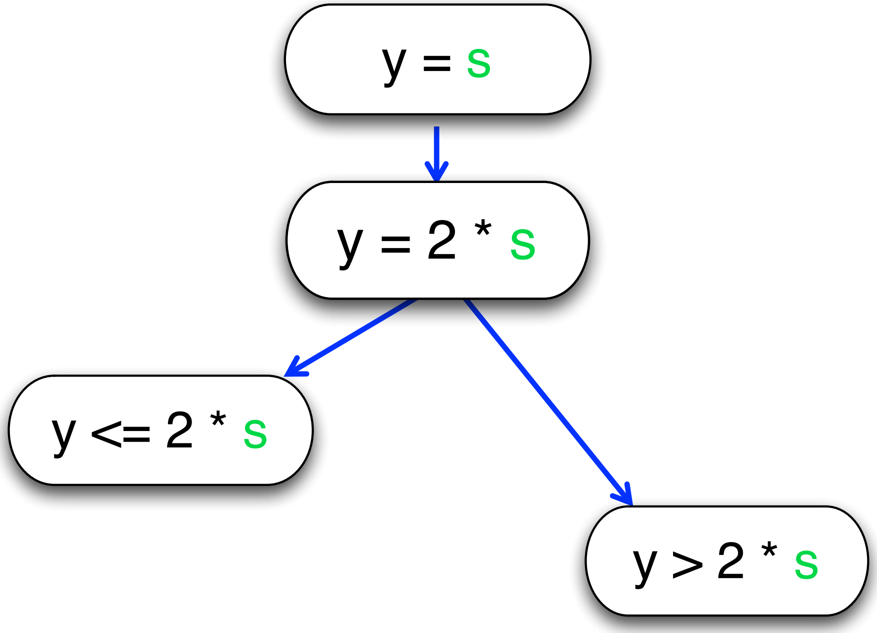
- Symbolic execution -> detect policy-violations

- Axiomatic semantics -> prove correctness

Symbolic Execution

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

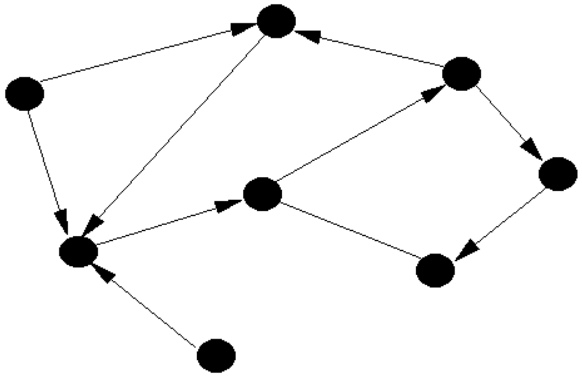
```
y = read()
y = 2 * y
if (y <= 12)
   fail()
else
   print("OK")
```



Formalism for Networking

Networks:

$$G = (V, E)$$



Packets:

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

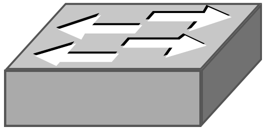
0110110101101
1001101110110
10011110001…

* 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:

$$\Psi = \left\{ \begin{array}{c} T_1 \\ \cdots \\ T_n \end{array} \right.$$

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

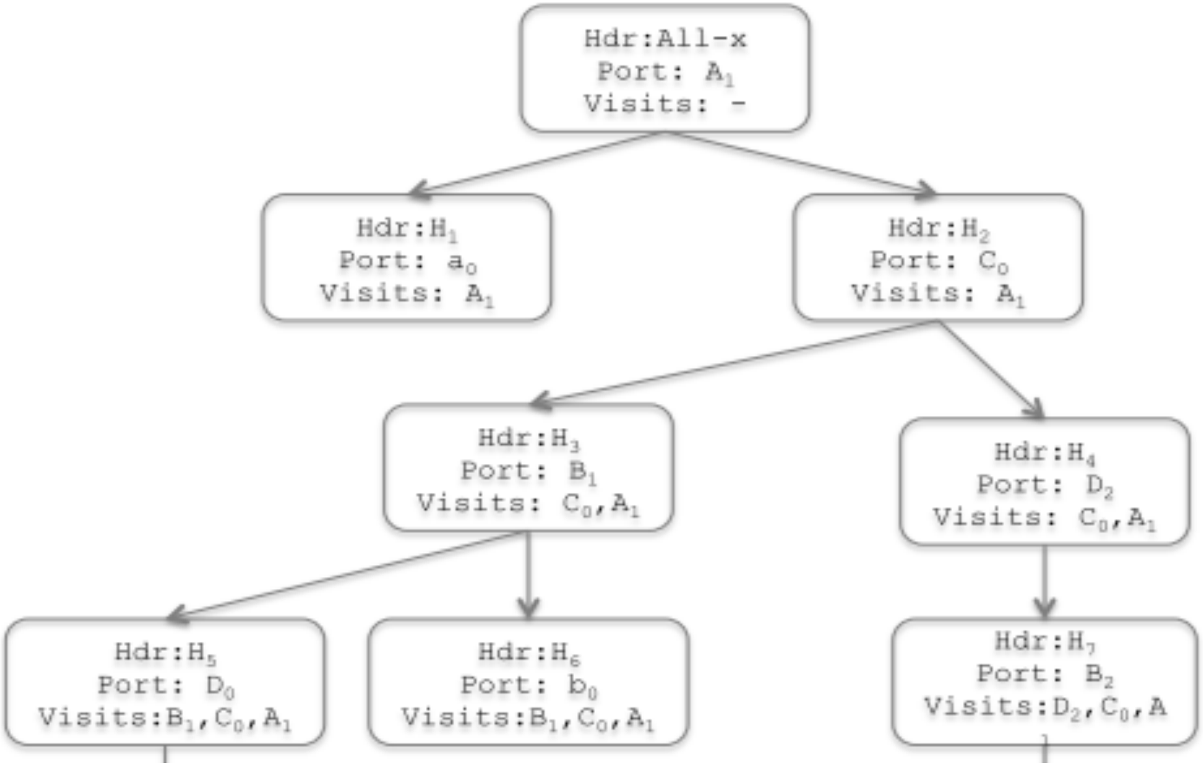
- Goal: Which **network path** will be taken for a given input?

Symbolic Execution For Networking

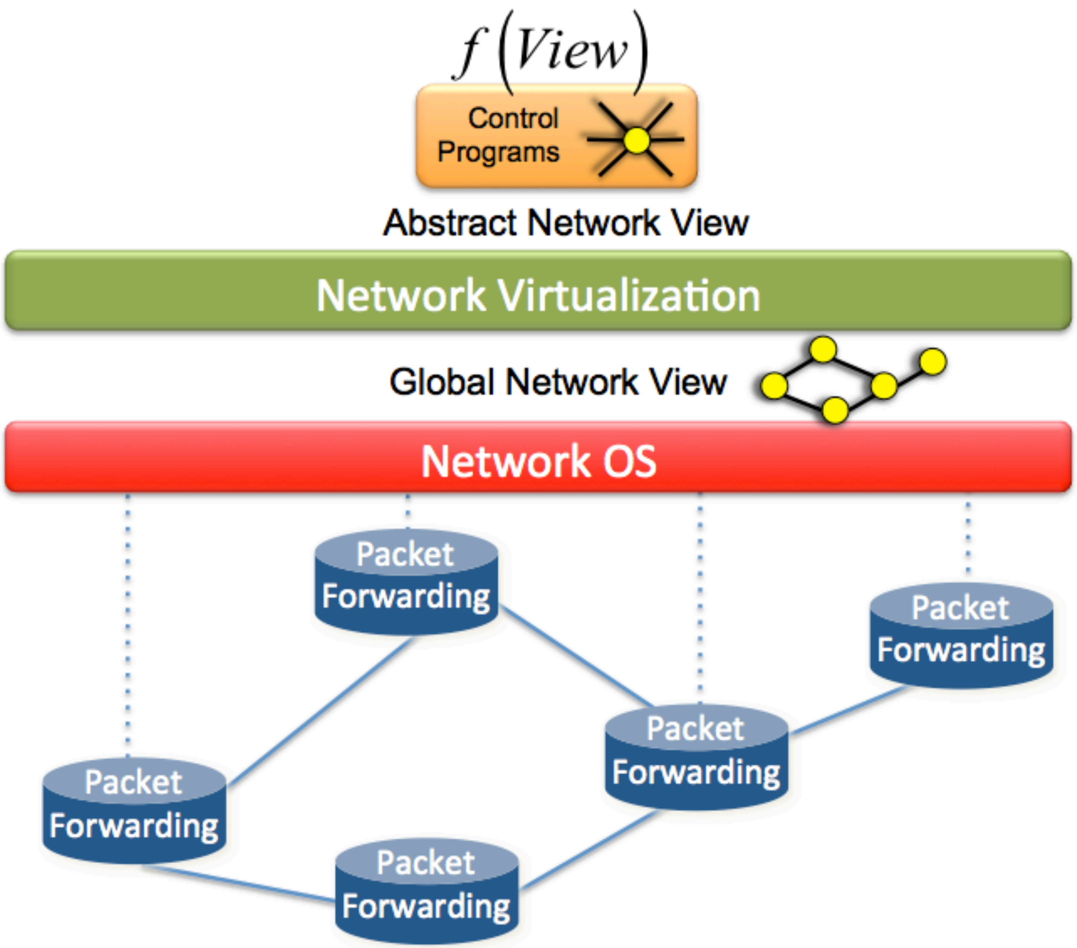
- Compute Ψ from routing tables
- For each host:
  - Insert symbolic packet x^L
  - Iteratively apply Ψ

Symbolic Execution For Networking

- End Result: Propagation Graph

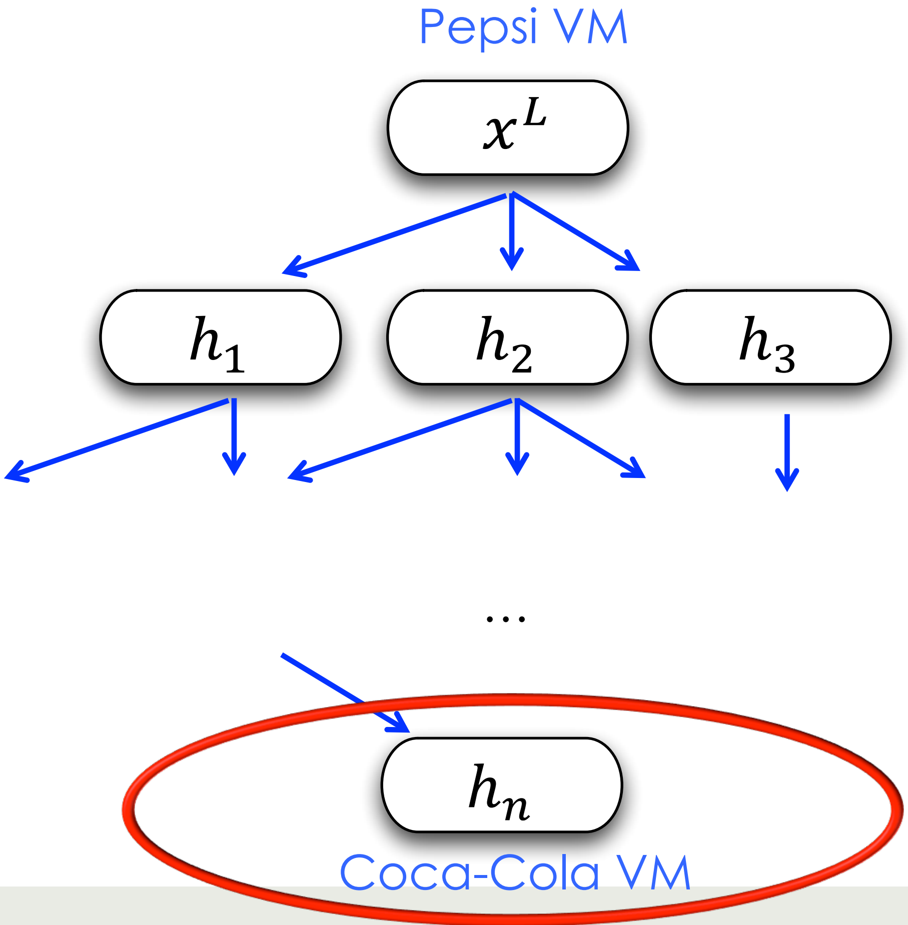


Brief Aside: Software-Defined Networking



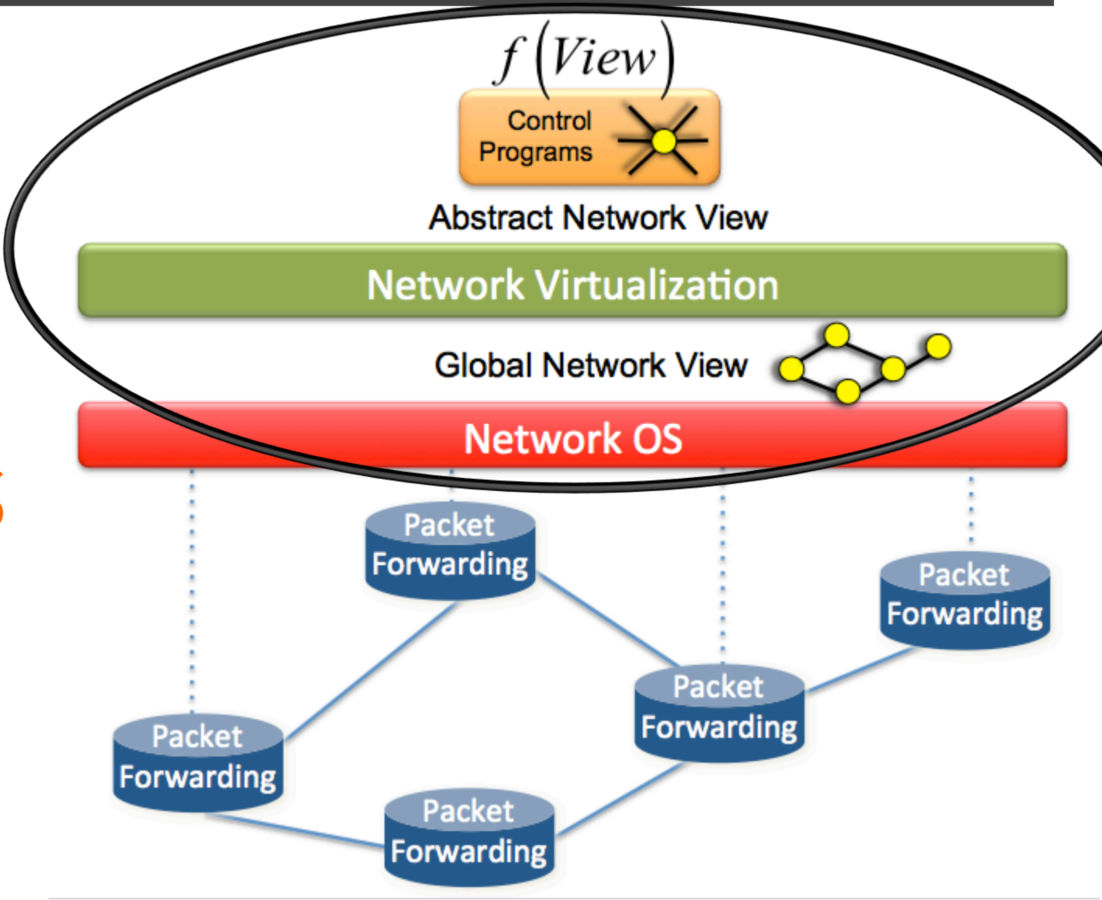
Example Policy-Violation

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



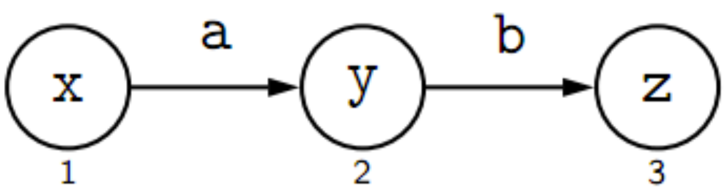
Proving Correctness

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

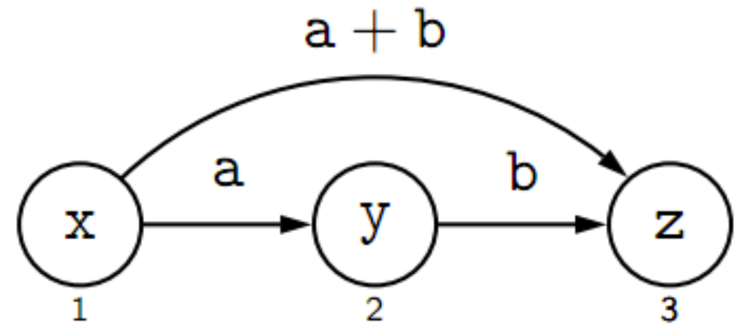


GP Graph Programming

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



$\Rightarrow$

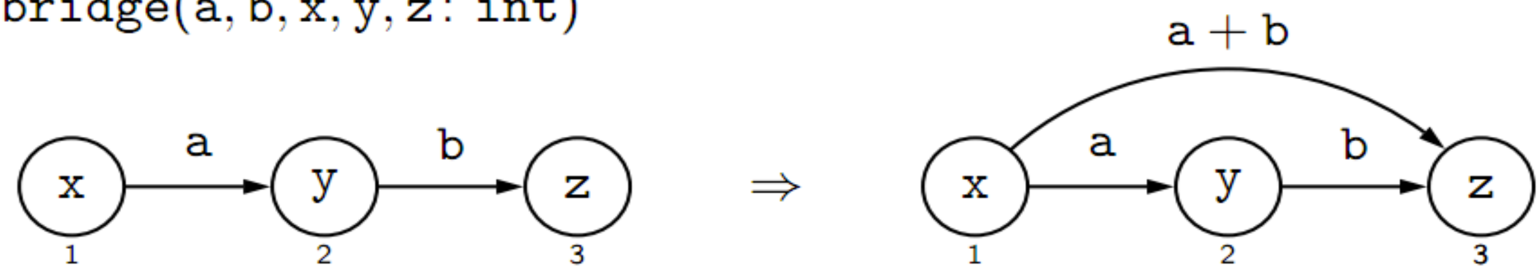


where not edge(1, 3)

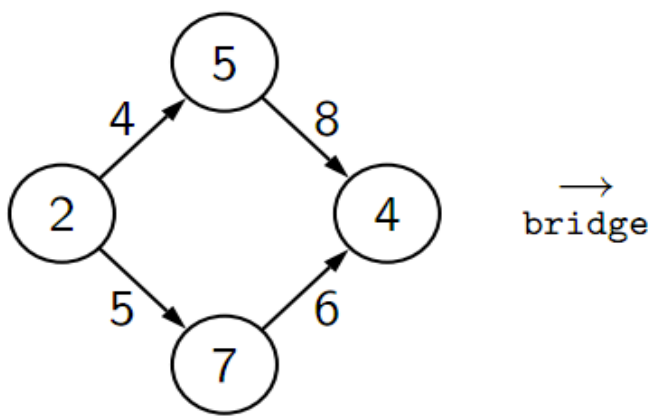
* Poskitt, Plump, Hoare Style Verification of Graph Programs

GP Graph Programming

$\mathtt{bridge}(\mathtt{a}, \mathtt{b}, \mathtt{x}, \mathtt{y}, \mathtt{z} \colon \mathtt{int})$

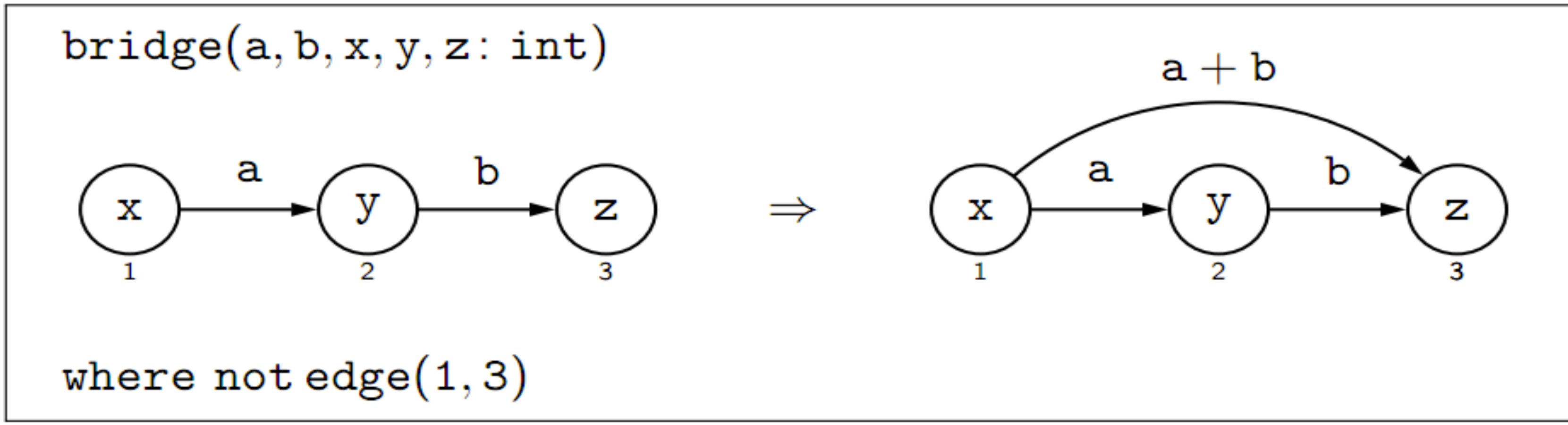


$\mathtt{where\ not\ edge}(1, 3)$



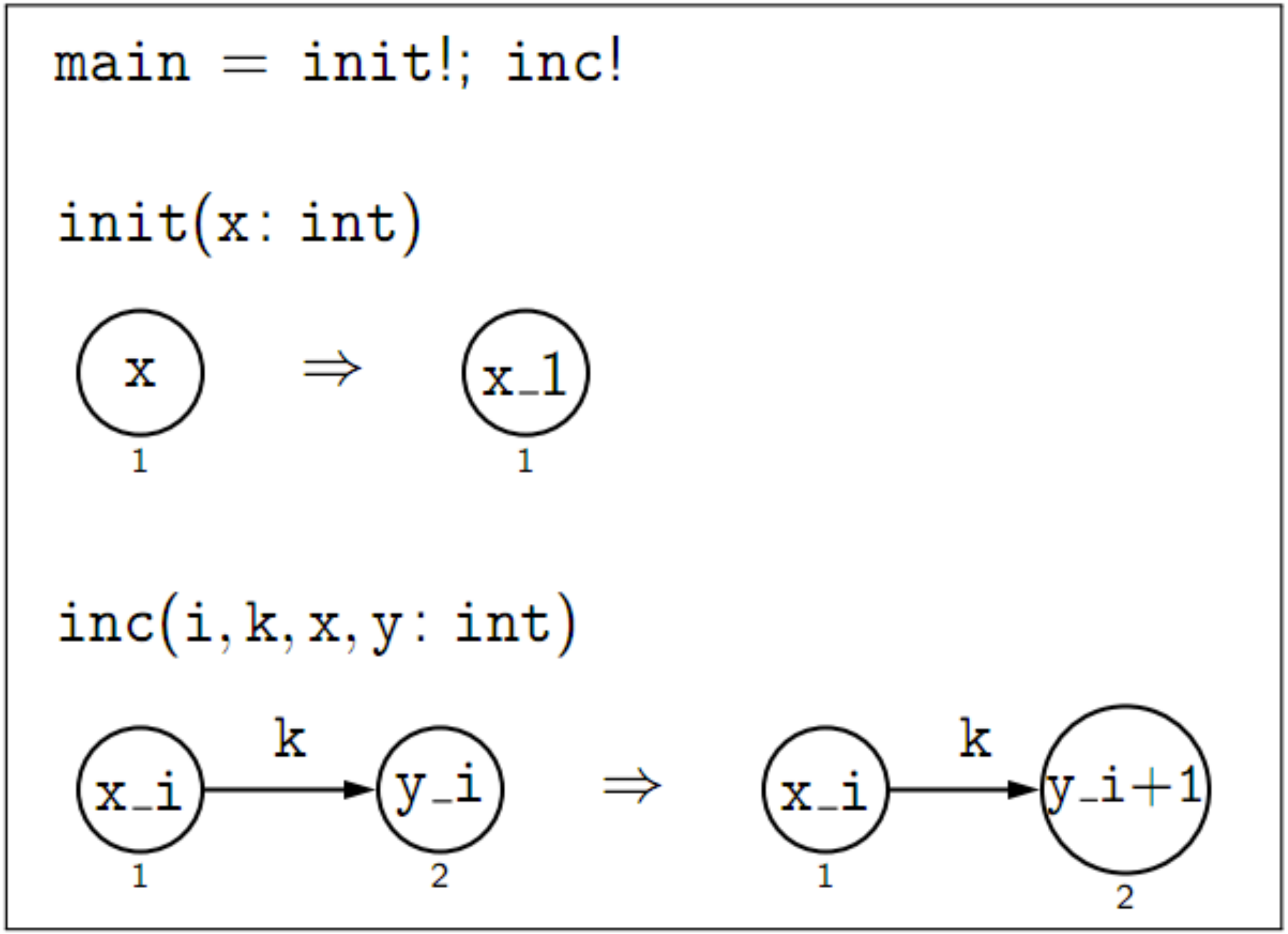
* Poskitt, Plump, Hoare Style Verification of Graph Programs

GP Graph Programming



* Poskitt, Plump, Hoare Style Verification of Graph Programs

GP Graph Coloring



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"

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

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.

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

Axiomatic Semantics

Rule schema application.

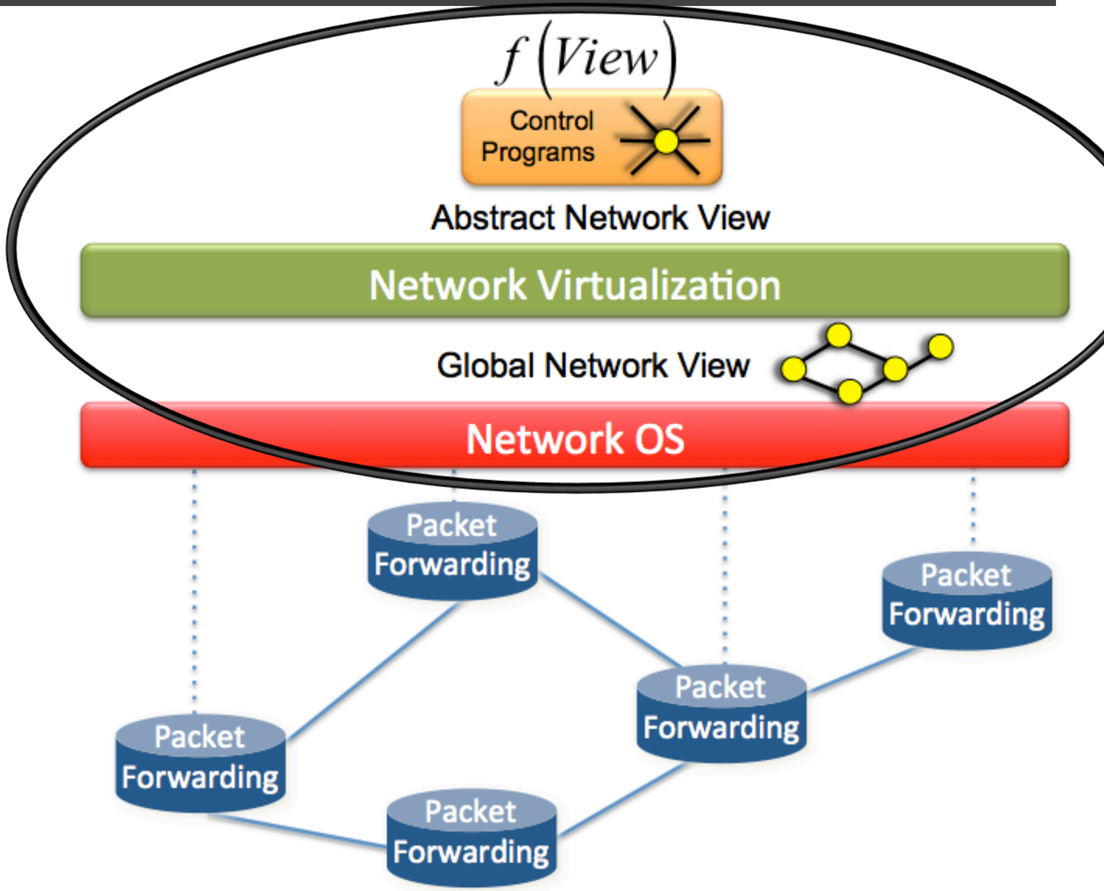
$$[\text{rule}]\ \frac{}{\{\text{Pre}(r, c)\}\ r\ \{c\}}$$

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

$$
\begin{array}{rcl}
c & = & \neg\exists(\ \mathsf{a}\ \mid \mathsf{type}(\mathsf{a}) \neq \mathsf{int}) \\
d & = & \forall(\ \mathsf{a}_1, \exists(\ \mathsf{a}_1 \mid \mathsf{a} = \mathsf{b\_c} \wedge \mathsf{type}(\mathsf{b}, \mathsf{c}) = \mathsf{int})) \\
e & = & \forall(\ \mathsf{a}_1, \exists(\ \mathsf{a}_1 \mid \mathsf{type}(\mathsf{a}) = \mathsf{int}) \vee \exists(\ \mathsf{a}_1 \mid \mathsf{a} = \mathsf{b\_c} \wedge \mathsf{type}(\mathsf{b}, \mathsf{c}) = \mathsf{int})) \\
\neg\mathrm{App}(\{\mathtt{init}\}) & = & \neg\exists(\ \mathtt{x}\ \mid \mathsf{type}(\mathtt{x}) = \mathsf{int}) \\
\neg\mathrm{App}(\{\mathtt{inc}\}) & = & \neg\exists(\ \mathtt{x\_i} \xrightarrow{\mathtt{k}} \mathtt{y\_i} \mid \mathsf{type}(\mathtt{i}, \mathtt{k}, \mathtt{x}, \mathtt{y}) = \mathsf{int}) \\
\mathrm{Pre}(\mathtt{init}, e) & = & \forall(\ \mathtt{x}_1\ \ \mathsf{a}_2 \mid \mathsf{type}(\mathtt{x}) = \mathrm{int}, \exists(\ \mathtt{x}_1\ \ \mathsf{a}_2 \mid \mathsf{type}(\mathsf{a}) = \mathsf{int}) \\
 & & \vee\ \exists(\ \mathtt{x}_1\ \ \mathsf{a}_2 \mid \mathsf{a} = \mathsf{b\_c} \wedge \mathsf{type}(\mathsf{b}, \mathsf{c}) = \mathsf{int})) \\
\mathrm{Pre}(\mathtt{inc}, d) & = & \forall(\ (\mathtt{x\_i})_1 \xrightarrow{\mathtt{k}} (\mathtt{y\_i})_2\ \ \mathtt{a}_3 \mid \mathsf{type}(\mathtt{i}, \mathtt{k}, \mathtt{x}, \mathtt{y}) = \mathsf{int}, \\
 & & \exists(\ (\mathtt{x\_i})_1 \xrightarrow{\mathtt{k}} (\mathtt{y\_i})_2\ \ \mathtt{a}_3 \mid \mathsf{a} = \mathsf{b\_c} \wedge \mathsf{type}(\mathsf{b}, \mathsf{c}) = \mathsf{int}))
\end{array}
$$

Proving Correctness

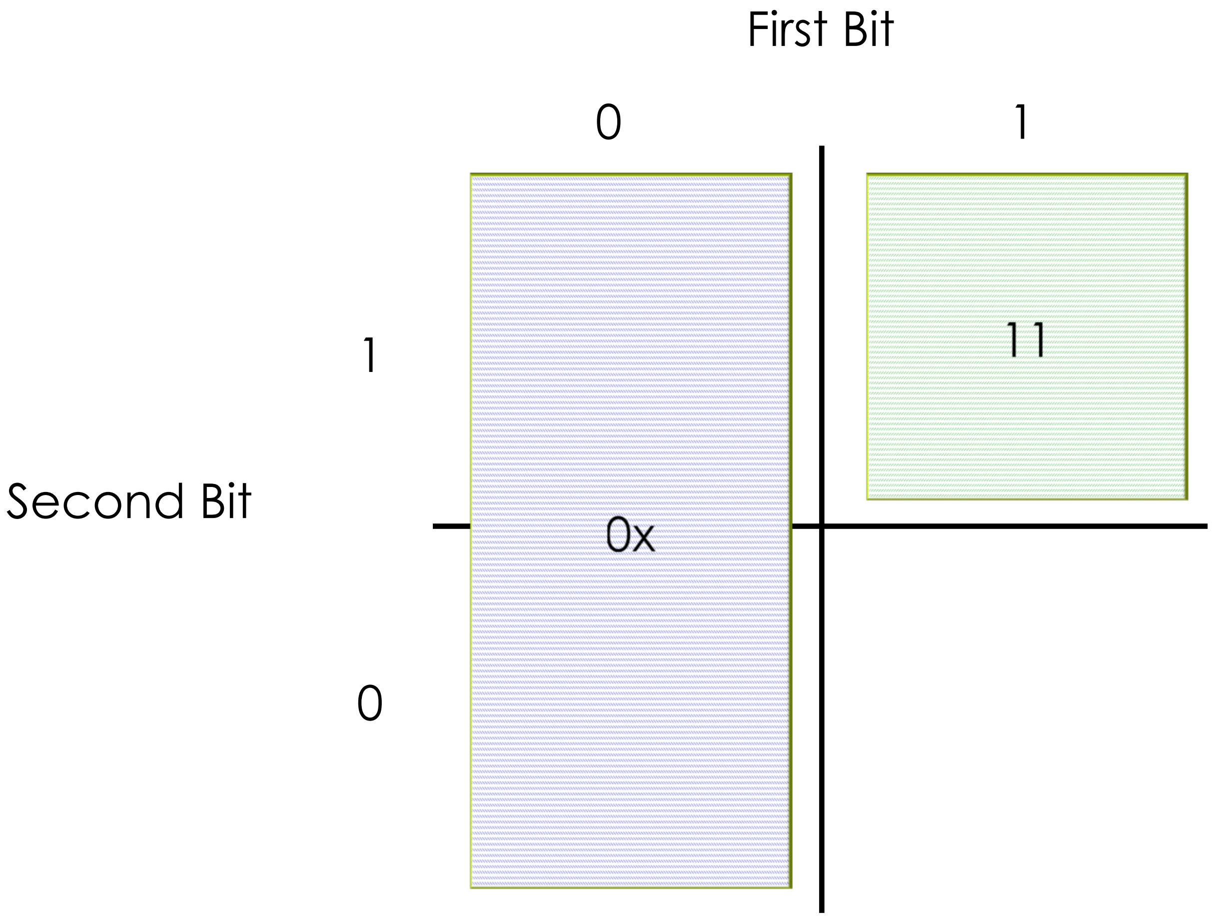
- Prove all the things!



Summary

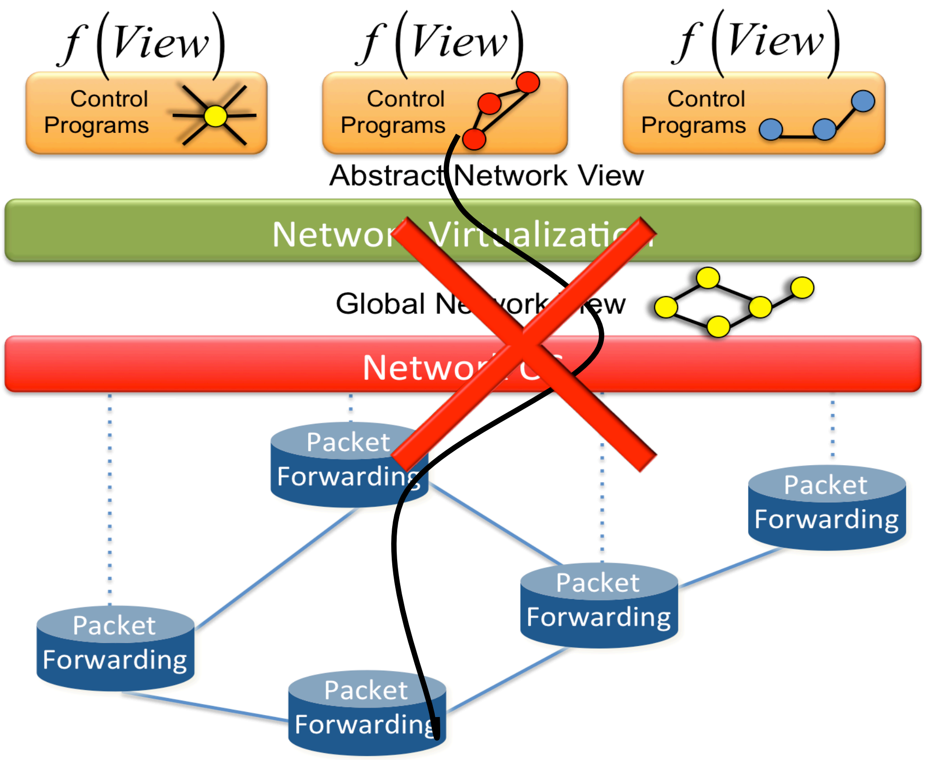
- PL techniques FTW!
- I adapted symbolic execution to find bugs in network software
- I demonstrated correctness proving for network algorithms

Symbolic Headerspace



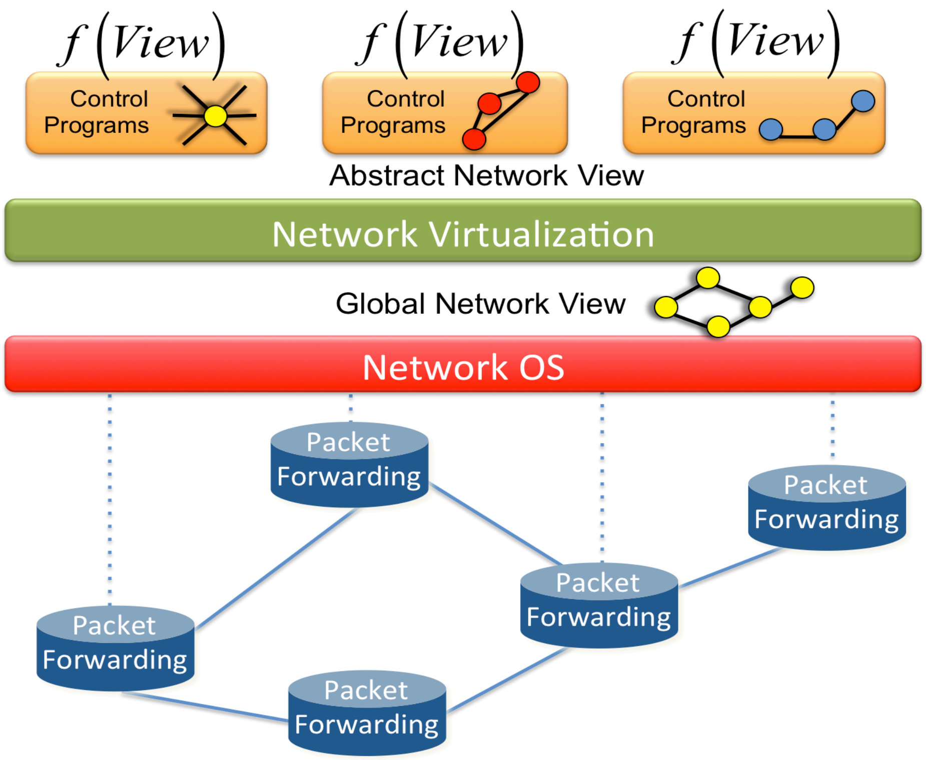
Symbolic Execution For Networking

- Goal: detect **policy-violations**
- "Network doesn't do what I tell it to"



Symbolic Execution For Networking

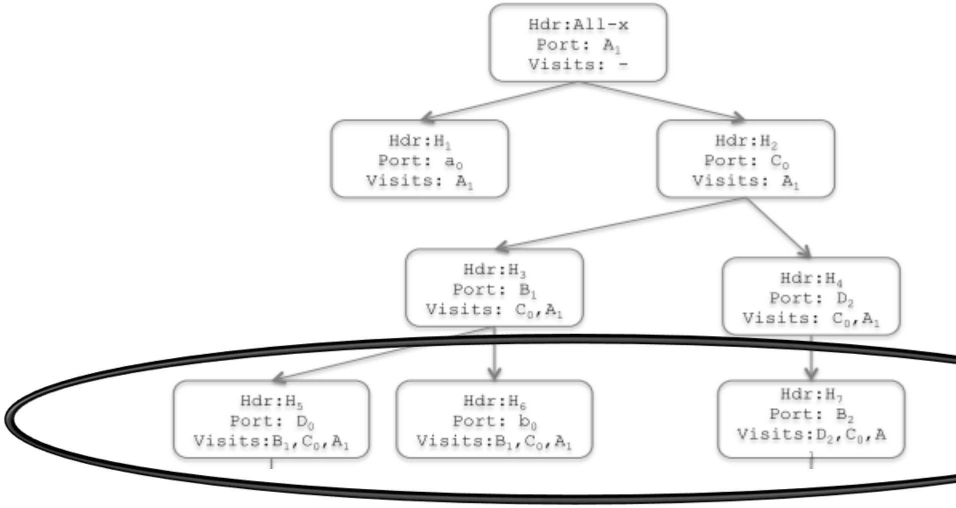
- Approach:
  - Demonstrate **isomorphism** between behavior of virtual view and physical network



Symbolic Execution For Networking

Network Behavior:

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



"'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).

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