Network Verification
Solvers, Symmetries, Surgeries

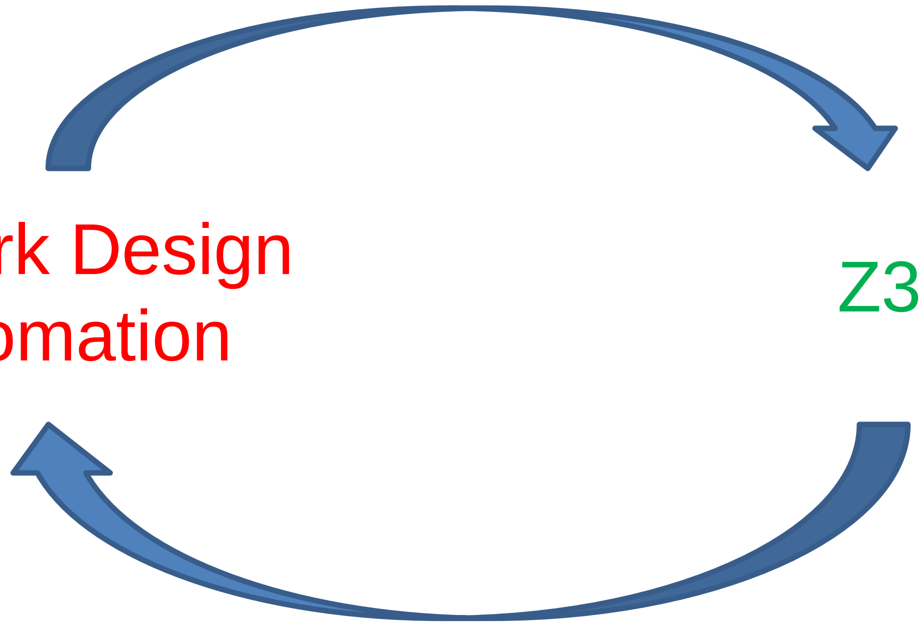
Nikolaj Bjørner

NetPL, August, 2016

Microsoft Research

Networking needs:

Configuration Sanity/Synthesis, Programming, Provisioning



Network Design Automation

Z3

Z3 advances:

Bit-vector Reasoning ~ Header Spaces

Reachability Checking, Quantitative Reasoning

Symbolic Analysis with Z3

$$x^2 + y^2 < 1 \; and \; xy > 0.1 \quad \Rightarrow \quad \text{sat}, \overbrace{x = \frac{1}{8}, y = \frac{7}{8}}^{\text{Solution/Model}}$$

$$x^2 + y^2 < 1 \; and \; xy > 1 \quad \Rightarrow \quad \text{unsat, Proof}$$

Is execution path *P* feasible?

SAGE

W
I
T
N
E

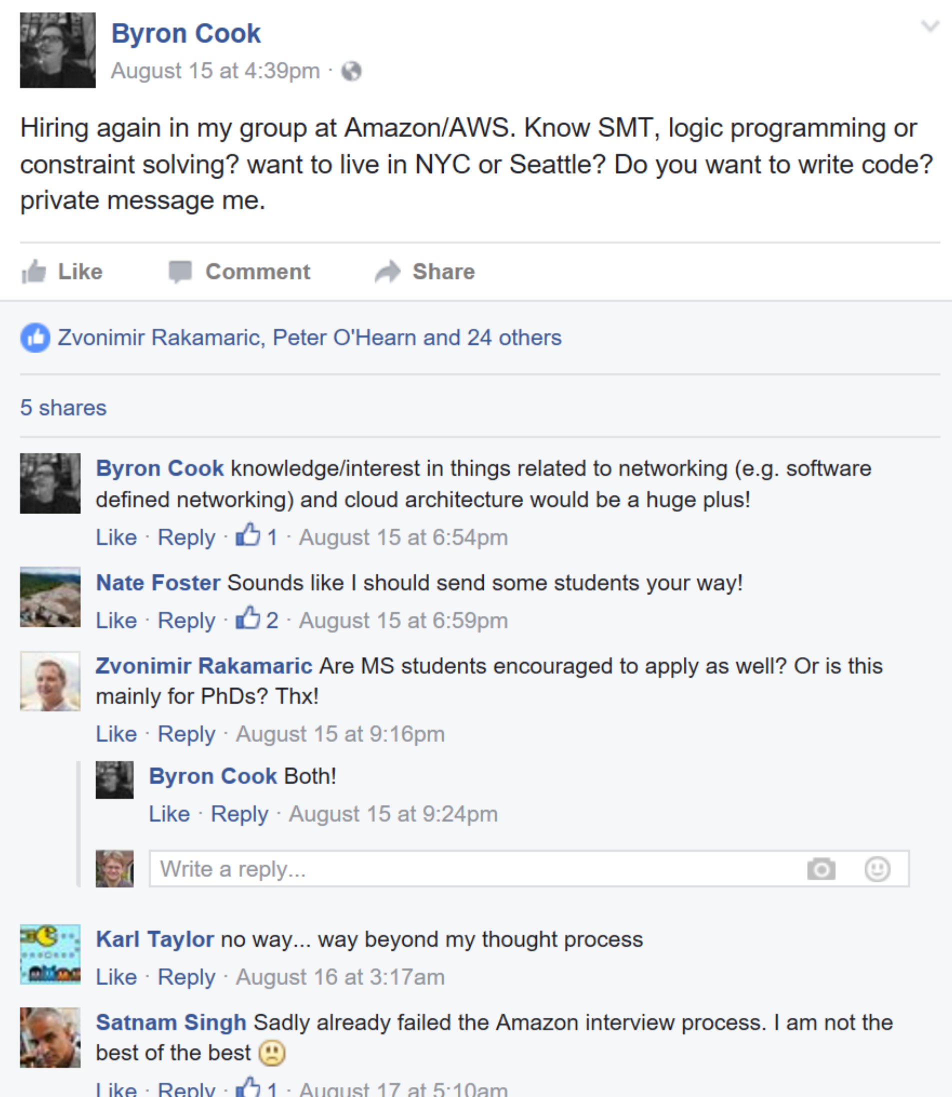
Z3 solved more than **10 billion** constraints created by SymEx tools including SAGE checking Win8,10 and Office

ti

Does Policy Satisfy Contract?

Z3 used by Pex,
Static Driver Verifier,
many other tools

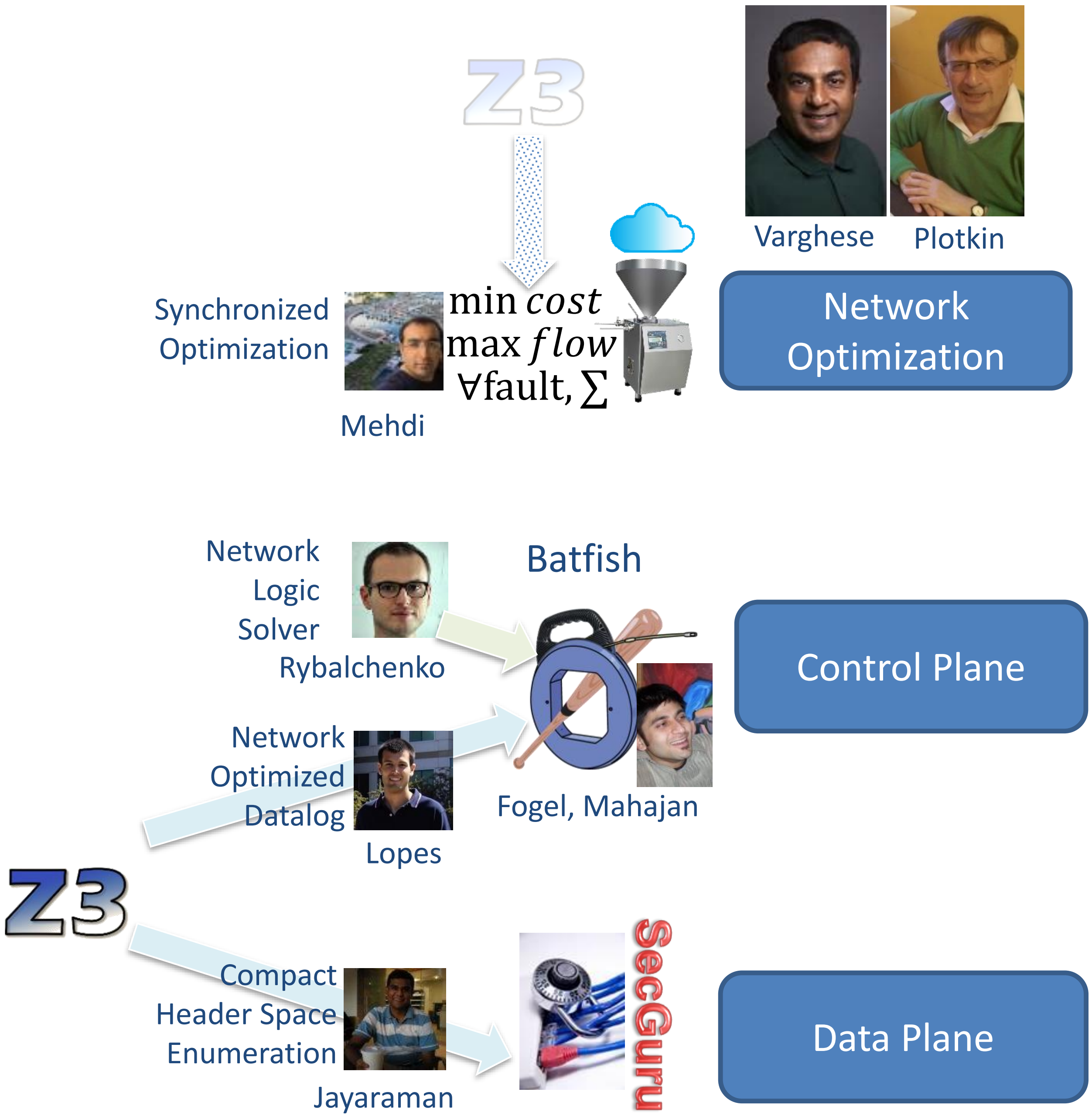
Our competition also likes symbolic solving ☺



Microsoft Azure and MSR are

always hiring.

Top engineering and research orgs with big and long term bets.



Application	Research
Network buildout Traffic Engineering	Flows and Fault analysis Some secret sauce ☺.
Reachability in IP networks	Network Optimized Datalog Symmetries and surgeries
Sanity checking of Data plane Configuration	Models of Bit-vector formulas Contracts & Netw. Beliefs

Calculus and Solvers

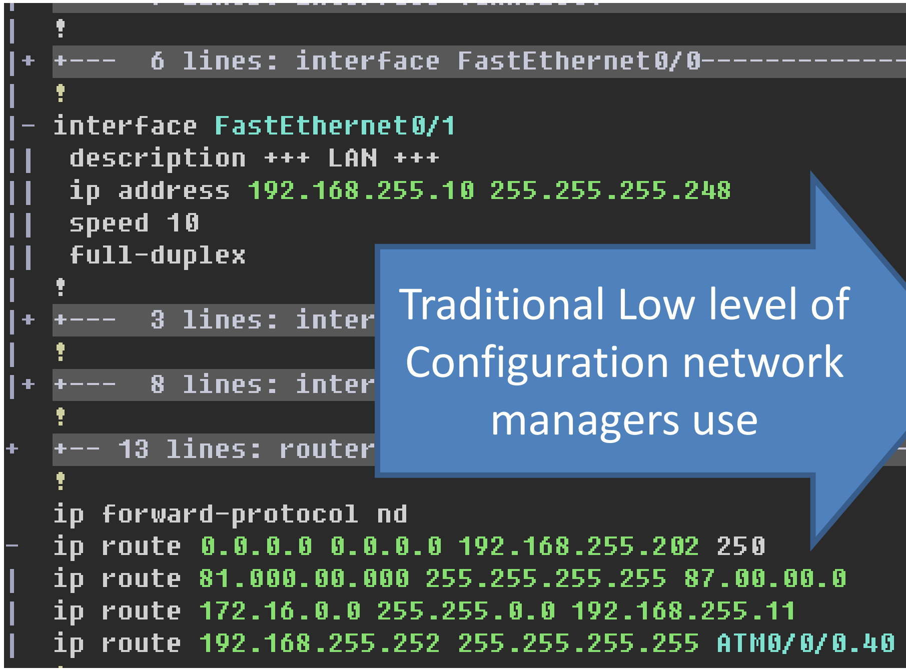
Application	Calculus	Solver
SecGuru: Access Control Routing Validation Static configurations for Border Gateway Protocol	Satisfiability Modulo Theories for Bit-vectors	SAT
Checking *beliefs* in networks	Network Optimized Datalog	Datalog for Header Spaces
	Network Symmetries and Surgeries	Tries for Header Space partitioning
Verifying SDN controllers	Quantified logical formulas	Instantiation based reasoning

Verification: Values and Obstacles

	Hardware	Software	Networks
	Chips	Devices (PC, phone)	Service
Bugs are:	Burned into silicone	Exploitable, workarounds	Latent, Exposed
Dealing with bugs:	Costly recalls	Online updates	Live site incidents
Obstacles to eradication:	Design Complexity	Code churn, legacy, false positives	Topology, configuration churn
Value proposition	Cut time to market	Safety/OS critical systems, Quality of code base	Meet SLA, Utilize bandwidth, Enable richer policies

SecGuru

Policies as Logical Formulas



```
|    !
|+ +---   6 lines: interface FastEthernet0/0-------------
|    !
|- interface FastEthernet0/1
||  description +++ LAN +++
||  ip address 192.168.255.10 255.255.255.248
||  speed 10
||  full-duplex
|    !
|+ +---   3 lines: inter
|    !
|+ +---   8 lines: inter
|    !
+  +-- 13 lines: router
   !
   ip forward-protocol nd
-  ip route 0.0.0.0 0.0.0.0 192.168.255.202 250
|  ip route 81.000.00.000 255.255.255.255 87.00.00.0
|  ip route 172.16.0.0 255.255.0.0 192.168.255.11
|  ip route 192.168.255.252 255.255.255.255 ATM0/0/0.40
```

Traditional Low level of Configuration network managers use

Precise Semantics as formulas

$$Allow: \quad (10.20.0.0 \le srcIp\ 10.20.31.255) \wedge (157.55.252.0 \le dstIp \le 157.55.252.255) \wedge (protocol = 6)$$

$$Deny: \quad (65.52.244.0 \le dstIp \le 65.52.247.255) \wedge (protocol = 4)$$

Combining semantics

Contracts/ Policies

Z3

$$\left(\bigvee_i Allow_i\right) \wedge \left(\bigwedge_j \neg Deny_j\right)$$

Semantic Diffs

Access Control

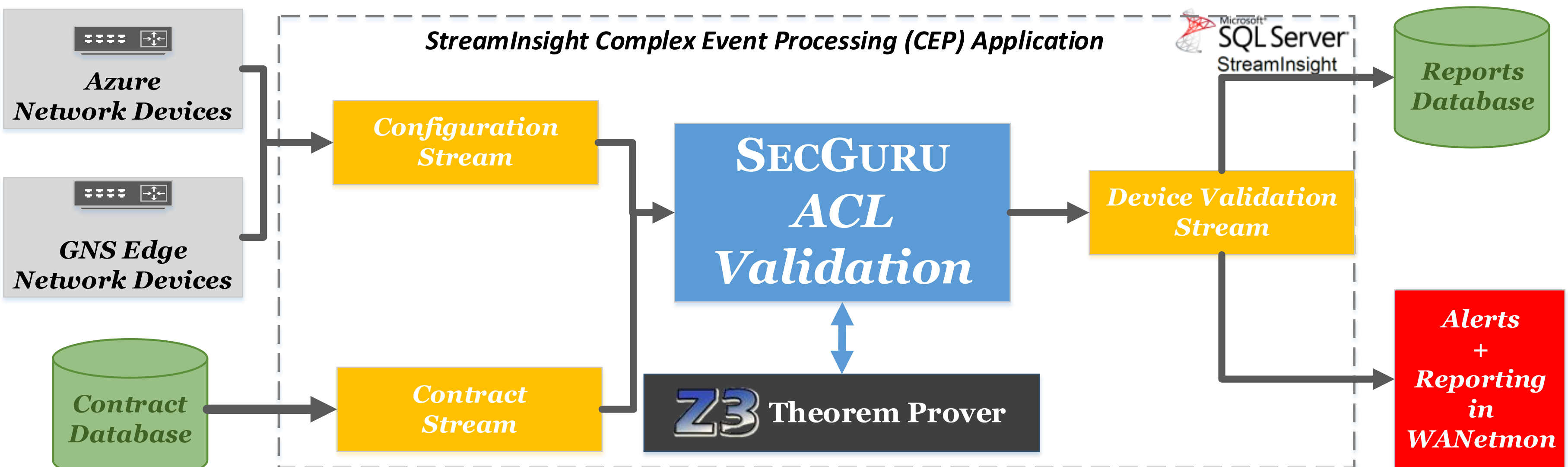
Contract:

DNS ports on DNS servers are **accessible** from tenant devices over both TCP and UDP.

Contract:

The SSH ports on management devices are **inaccessible** from tenant devices.

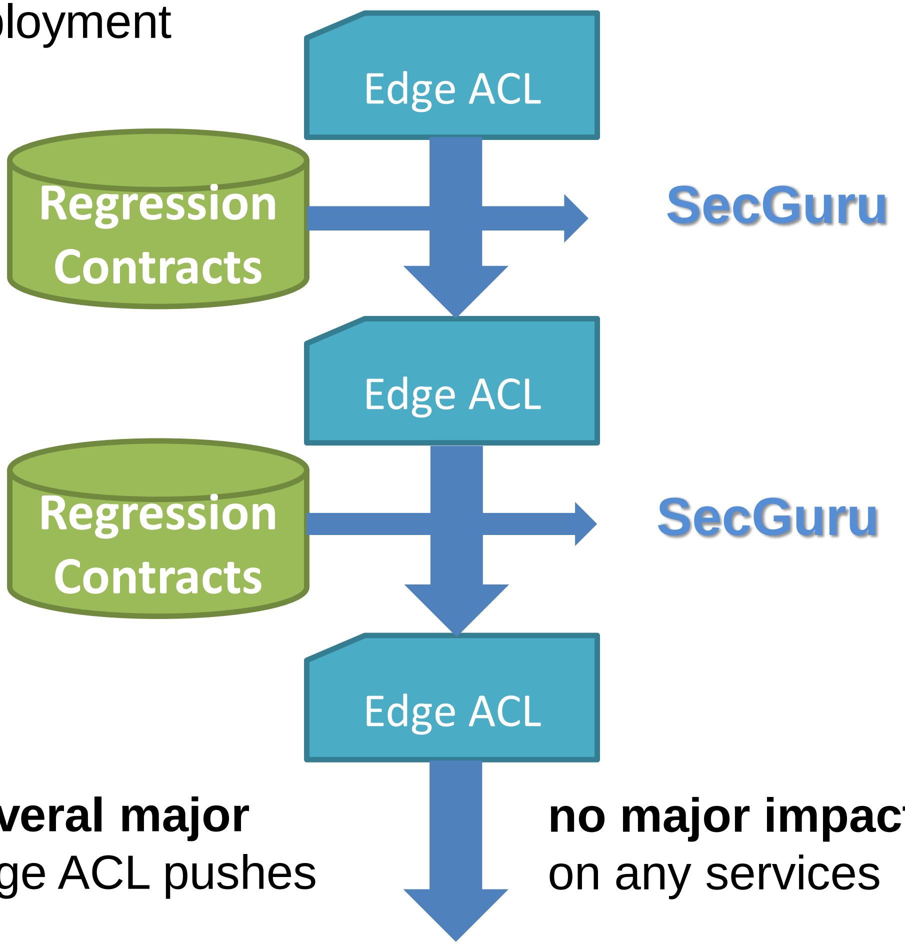
SecGuru workflow



Windows Azure Network Monitoring Infrastructure

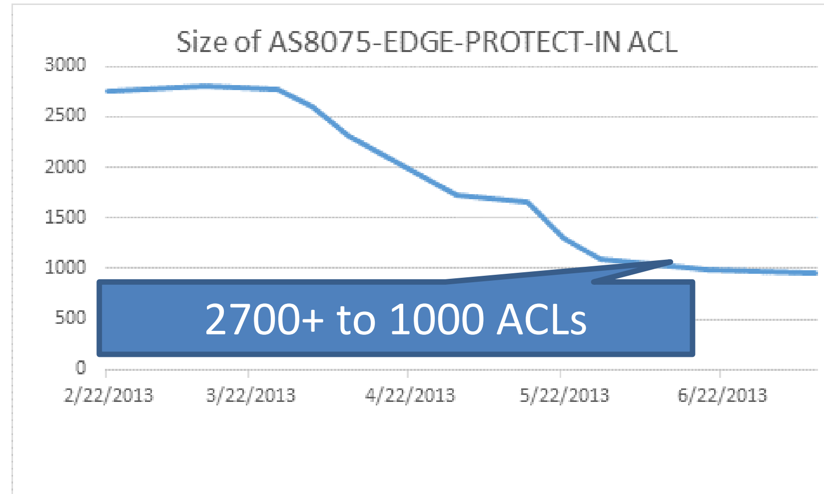
SecGuru for GNS edge ACLs

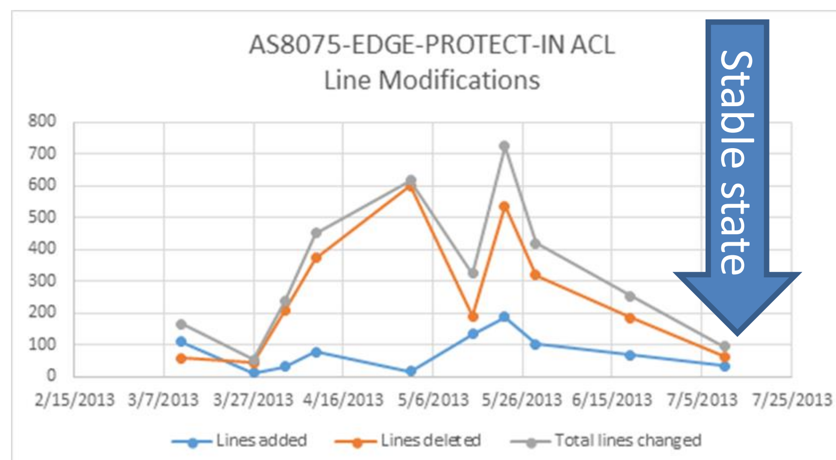
Regression test suite + SecGuru check correctness of Edge ACL prior to deployment



Several major Edge ACL pushes

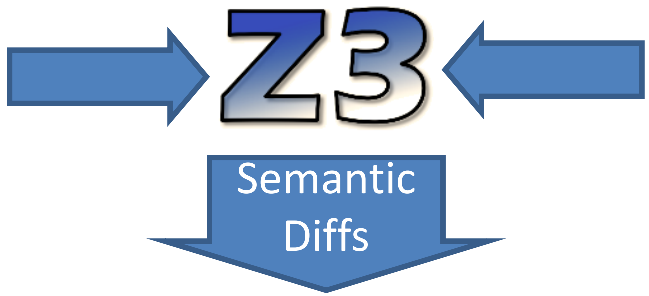
no major impact on any services





Beyond Z3: a *new* idea to go from one violation to all violations

$$\left(\bigvee_i Allow_i\right) \wedge \left(\bigwedge_j \neg Deny_j\right)$$

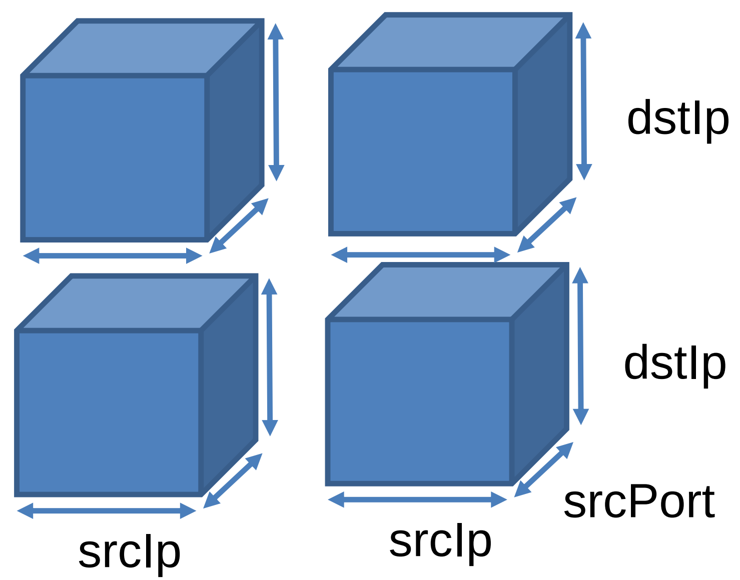


$$\neg\left[\left(\bigvee_m Allow_m\right) \wedge \left(\bigwedge_n \neg Deny_n\right)\right]$$

$$srcIp = 10.20.0.0/16, 10.22.0.0/16$$
$$dstIp = 157.55.252.000/24, 157.56.252.000/24$$
$$port = 80, 443$$

Representing solutions

- $2 * 2^{16} * 2 * 2^{8} * 2 \; = \; 2^{27}$ single solutions, or
- 8 products of contiguous ranges, or
- A single product of ranges



SecGuru contains optimized algorithm for turning single solutions into all (product of ranges)

Verifying Forwarding Rules with SecGuru

Routes

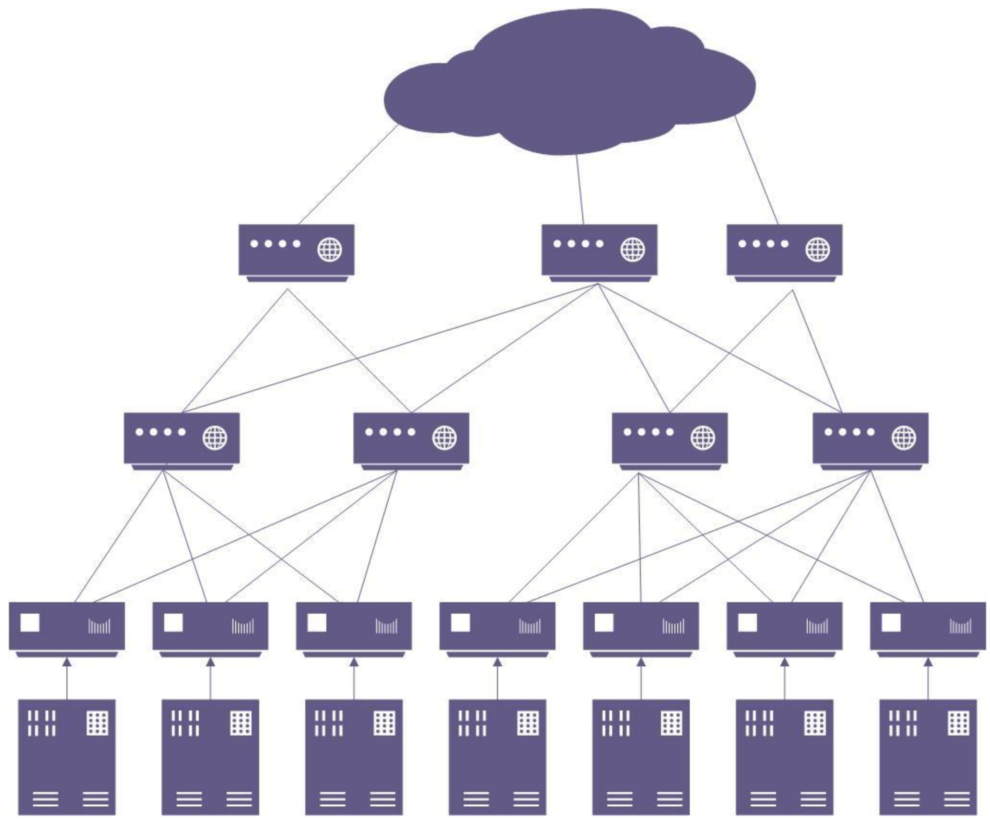
```
B E    0.0.0.0/0 [200/0] via 100.91.176.0, n1
                         via 100.91.176.2, n2

B E    10.91.114.0/25 [200/0] via 100.91.176.125, n3
                              via 100.91.176.127, n4
                              via 100.91.176.129, n5
                              via 100.91.176.131, n6
B E    10.91.114.128/25 [200/0] via 100.91.176.125, n3
                                via 100.91.176.131, n6
                                via 100.91.176.133, n7
  ...
```

Logic

$$
\begin{array}{l}
Router \equiv \\
\quad \textbf{if } \ldots \\
\quad \textbf{if } dst = 10.91.114.128/25 \textbf{ then } n_3 \vee n_6 \vee n_7 \textbf{ else} \\
\quad \textbf{if } dst = 10.91.114.0/25 \textbf{ then } n_3 \vee n_4 \vee n_5 \vee n_6 \textbf{ else} \\
\quad n_1 \vee n_2
\end{array}
$$

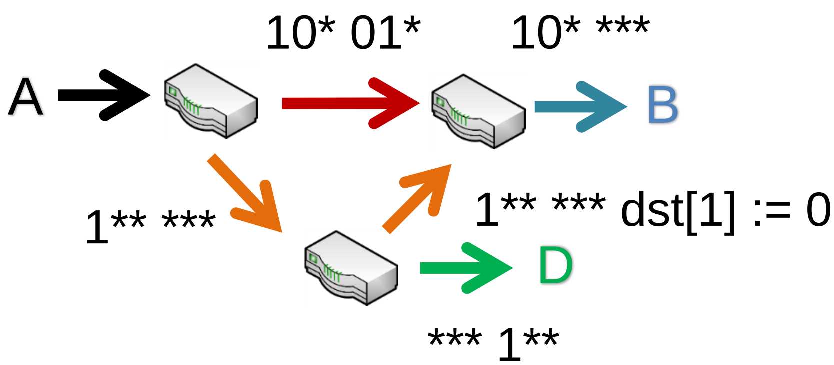
Contract

$$
\begin{array}{l}
Cluster(dst) \Rightarrow \\
\qquad Router_1(dst) \equiv Router_2(dst)
\end{array}
$$



Network Reachability

Checking *beliefs* in Dynamic Networks



$$
\begin{aligned}
G_{12} &:= dst = 10\star \wedge src = 01\star \\
G_{13} &:= \neg G_{12} \ \wedge\ dst = 1\star\star \\
G_{2B} &:= dst = 10\star \\
G_{3D} &:= src = 1\star\star \\
G_{32} &:= \neg G_{3D} \ \wedge\ dst = 1\star\star \\
Id &:= src' = src \ \wedge\ dst' = dst \\
Set0 &:= src' = src \ \wedge\ dst' = dst[2]\ 0\ dst[0]
\end{aligned}
$$

Which packets can reach B from A?

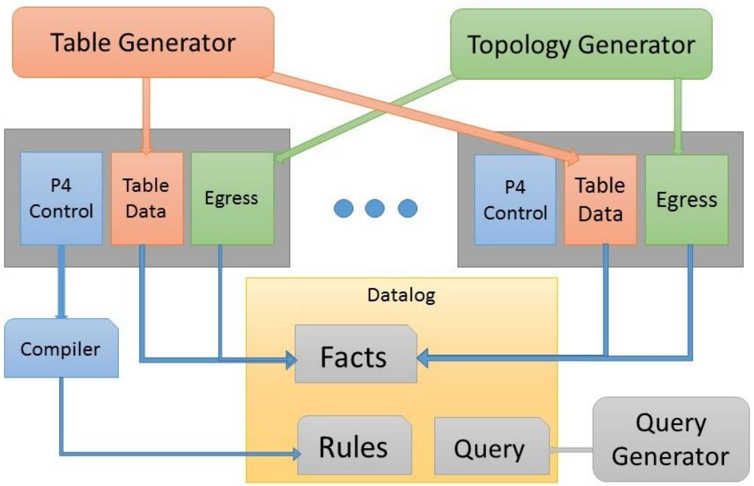
Datalog useful for encoding a broad range of queries. We use *belief* for a class of general properties that one may expect to hold of networks.

Sample belief: packets flow through middle-box

[Lopes, B, Godefroid, Jayaraman, Varghese NSDI’15]

$$
\begin{aligned}
& & & B(dst, src) \\
R1(dst, src) &\ :- & & G_{12} \wedge Id \wedge R2(dst', src') \\
R1(dst, src) &\ :- & & G_{13} \wedge Id \wedge R3(dst', src') \\
R2(dst, src) &\ :- & & G_{2B} \wedge Id \wedge B(dst', src') \\
R3(dst, src) &\ :- & & G_{3D} \wedge Id \wedge D(dst', src') \\
R3(dst, src) &\ :- & & G_{32} \wedge Set0 \wedge R2(dst', src') \\
A(dst, src) &\ :- & & R1(dst, src) \\
& \ ? & & A(dst, src)
\end{aligned}
$$

Applying NoD to $P4_{14}$



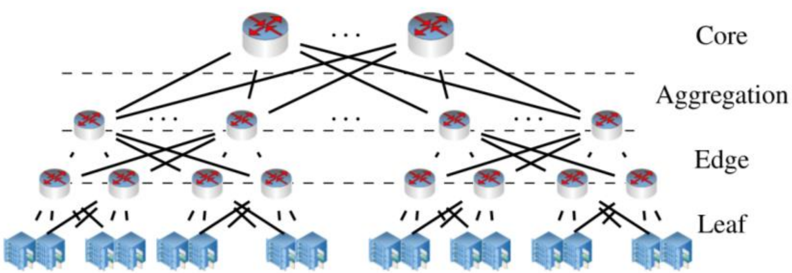

```
reach(S’) :-
 reach(S),
 router_processing(S, S’).

router_processing(S, S’) :-
  reset_local_data(S, S0),
  start(S0, S1),
  egress(S1.local.addr, S1.std_md.egress_spec, Next, Port),
  S’ = { S1 with std_md.ingress_port = Port, local.addr = Next }.

reset_local_data(S, S’) :-
  S’ = { S with local_md = 0, std_md = 0, parsed = 0 }.
```

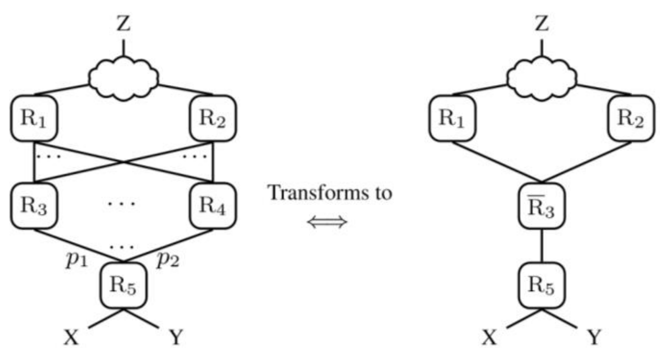
$$\text{apply}\ \frac{\begin{array}{c}\texttt{table}\ table\ \{reads\ actions\} \in Prog \\ act \in actions \\ vals = add\_entry\_table\_act(S.reads) \\ S,\mathcal{E} \xrightarrow{act(vals)} S' \\ S',\mathcal{E} \xrightarrow{stmt} S''\end{array}}{S,\mathcal{E} \xrightarrow{\texttt{apply}(table)\{act\ \{stmt\}\}} S''}$$

+ P4 code + Config → NoD

[Lopes, Rybalchenko, B, McKeown, Talayco, Varghese]

Scaling Network Verification using Symmetry and Surgery





A Theory of Network Dataplanes

- $out : Nodes \rightarrow 2^{Ports}$
- $Port := \{n.i \mid n \in Nodes, i \in out(n)\}$
- $links: Port_N \rightarrow Nodes$
- $h@n.i \longrightarrow h'@n'.i'$
  $\in Trans$
  $\subseteq (Header \times Port) \times (Header \times Port)$

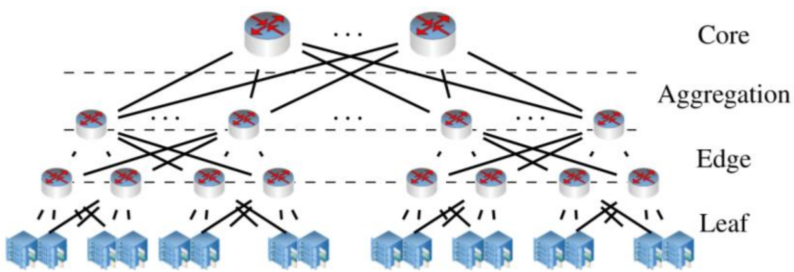
  Such that $n' = links(n.i), i' \in out(n')$

A basis for defining bisimulation relations:

$$h@n.i \sim h'@n'.i'$$

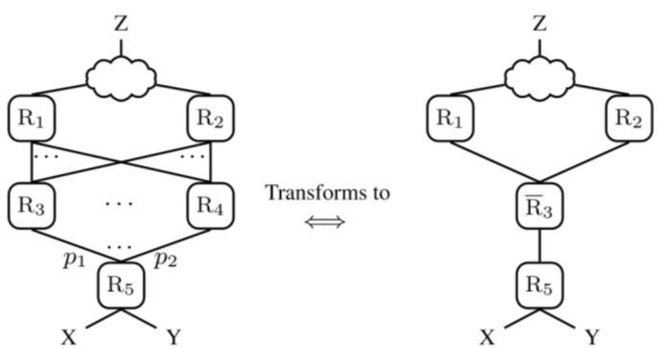
[*Plotkin, B, Lopes, Rybalchenko, Varghese, POPL 16*]

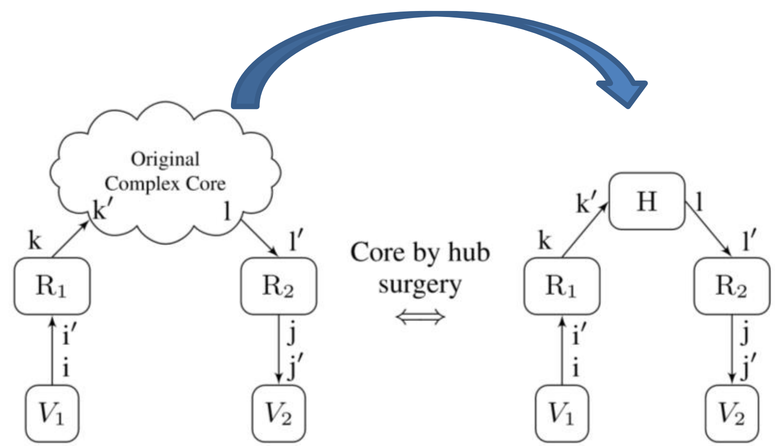
Scaling Network Verification using Symmetry and Surgery



A Toolbox of Network Transformations

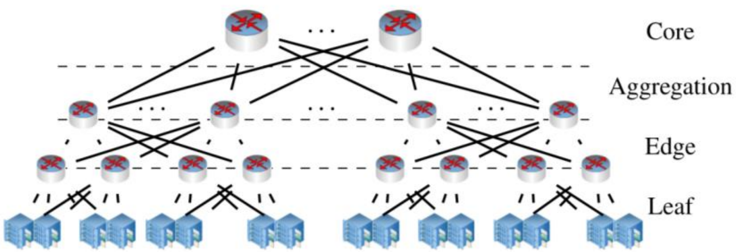
Example: Replace a core of a network by a single hub:





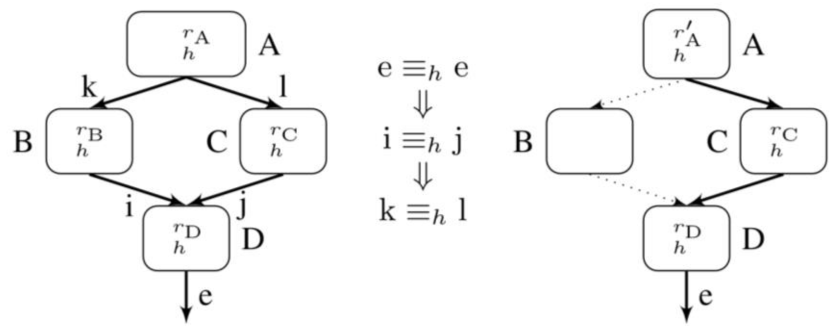
[*Plotkin, B, Lopes, Rybalchenko, Varghese, POPL 16*]

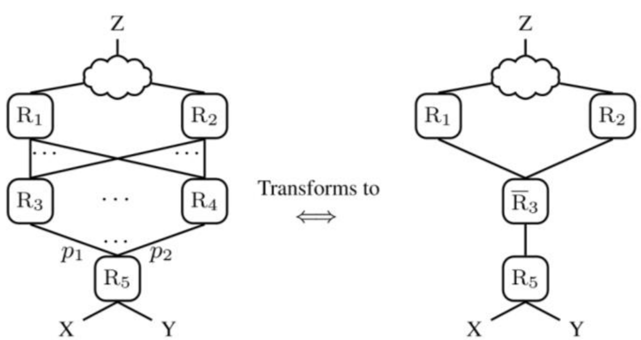
Scaling Network Verification using Symmetry and Surgery



Scaling comprehensive Network Verification

Example: Move rules from B to C if forwarding is the same.



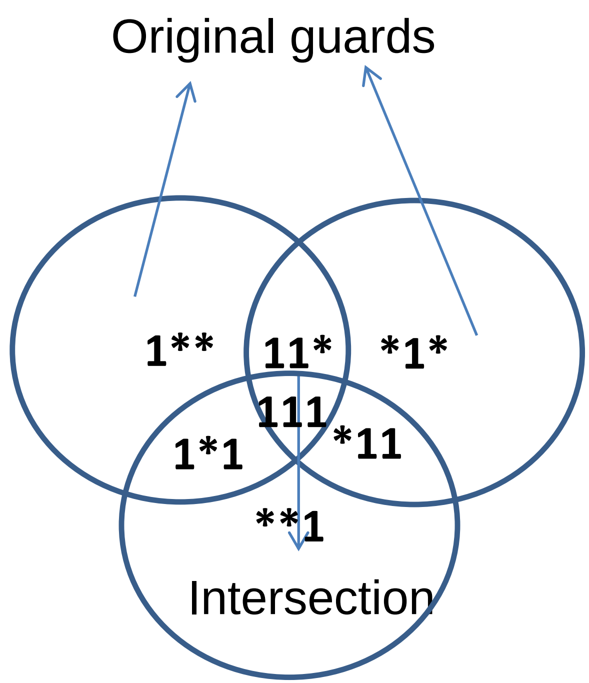


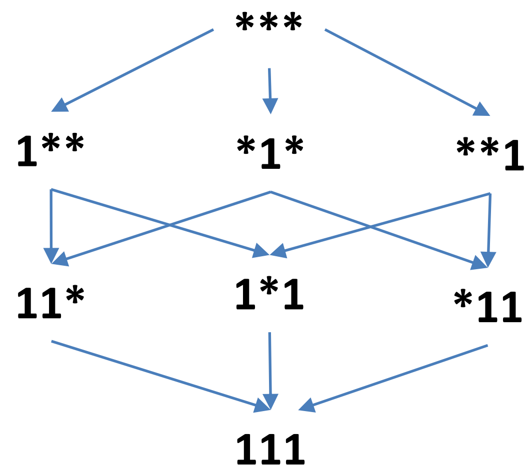
Relies on efficient representation of header equivalence classes.

Router Rules Venn Diagrams ddNF

Forwarding rules

1**	via port1
1	via port2
**1	via port3
***	via port2





[B, Juniwal, Mahajan, Seshia, Varghese MSR-TR]

Summary

Much is about Configuration Correctness:

- Is intent captured? (SecGuru)
- Usage (NoD + P4)
- Synthesis (Control Plane)
- Bandwidth Use and Provisioning (QNA)

Modern packet switched networks a good use case for PL + Symbolic Methods