Google Cloud

Automated Switch Validation with Path-Complete Testing at Scale

Shaan Nagy (UCSD*), Ali Kheradmand (Google)

P4 Workshop, October 2025

* Work done while at Google

P4-BAR:

Automatic Switch Validation

P4-Based Automated Reasoning[1] (P4-BAR)

Mission: Ensure switch stack works as expected (before deployment).

DVaaS: Dataplane Validation as a Service[2]

P4-Symbolic: Automated Test Packet Generation[3]

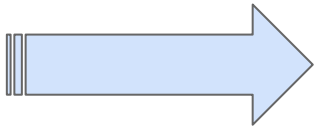
[1] Kheradmand, A., & Sistla, M. (2024). *Scaling P4-Based Automated Reasoning (Performance and Coverage).* (P4 Workshop '24)
[2] Smolka, S., & DiLorenzo, J. (2024). *P4-Based Automated Reasoning (P4-BAR) for the (Networking) Masses!* (P4 Workshop '24)
[3] Dak Albab, K.,et al. (2022). SwitchV: Automated SDN switch validation with P4 models. (SIGCOMM '22)

P4-Symbolic: ***Automatic*** Dataplane Test Generation

P4-Symbolic: **Automatically** derive tests from a **formal specification** of how the switch should work

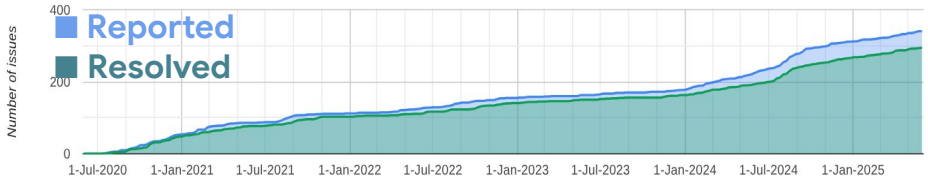
- Comprehensive coverage
- Effortless evolution

P4-BAR Team Impact



300+ (unique) bugs found so far
(and many more bugs prevented)

200+ from P4-Symbolic

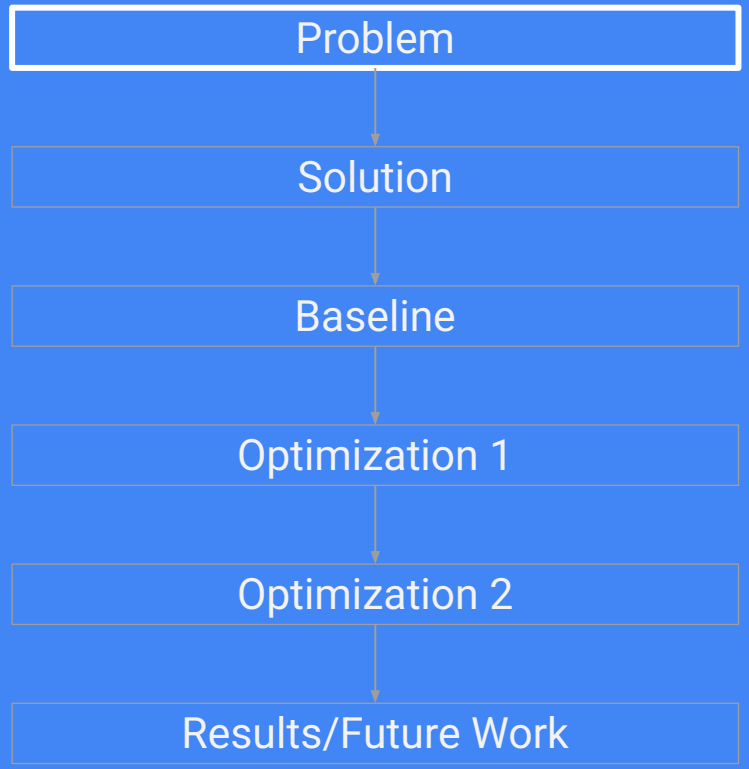


SwitchV: Automated SDN Switch Validation with P4 Models (SIGCOMM'22)

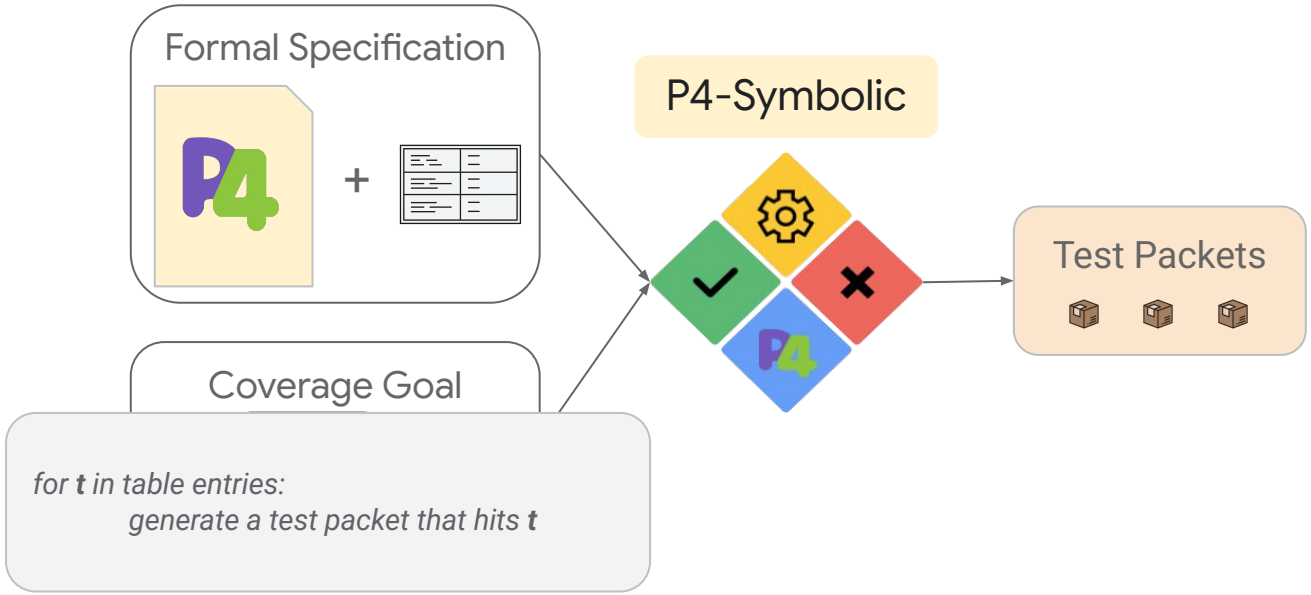
Kinan Dak Albab, Jonathan Dilorenzo, Stefan Heule, Ali Kheradmand, Steffen Smolka, Konstantin Weitz, Muhammad Tirmazi, Jiaqi Gao, Minlan Yu

Today’s Problem:

Situational bugs can escape test coverage.



Problem: Per-Table Entry Test Coverage Can Miss Bugs



Problem: Per-Table Entry Test Coverage Can Miss Bugs

Production Scenario

ACL Table

IPv4 Packet →  → **PASS**

ACL Table

IPv6 Packet →  → **FAIL**

Bug requires
specific combination of conditions!

Problem: Per-Table Entry Test Coverage Can Miss Bugs

Production Scenario

Nearly missed bug

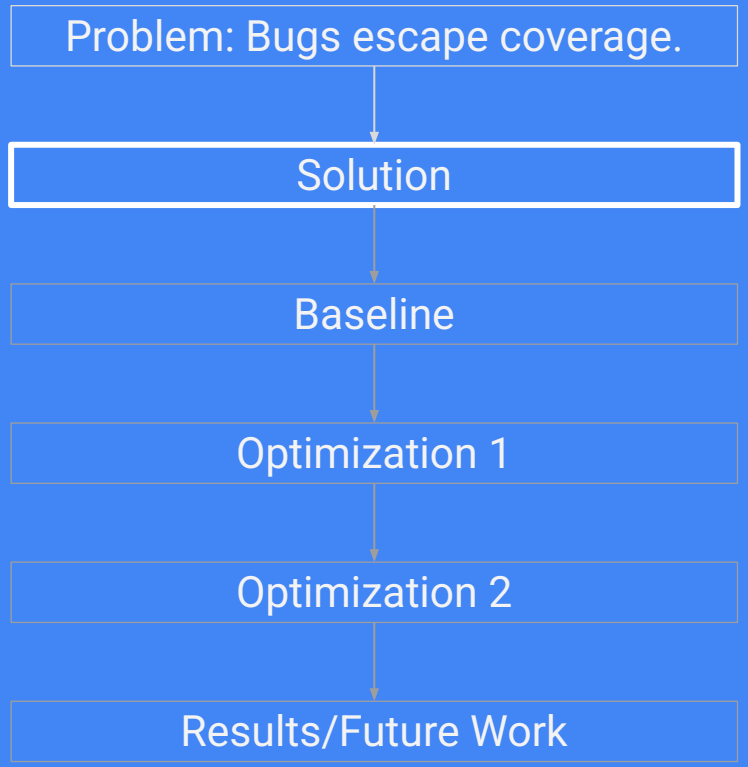
Main Problem:

How can we catch bugs that only appear for specific combinations of conditions?

Bug requires specific combination of conditions!

Solution:

Generate tests for path coverage.



Solution: Path Coverage

Problem: Some bugs require a combination of conditions to appear.

Q: Can we test every combination of conditions?

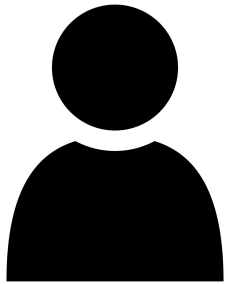
A: No! There are too many combinations.

Q: What combinations of conditions are important to test?

A: The ones that have different behaviors in the  model!

Solution: Path Coverage

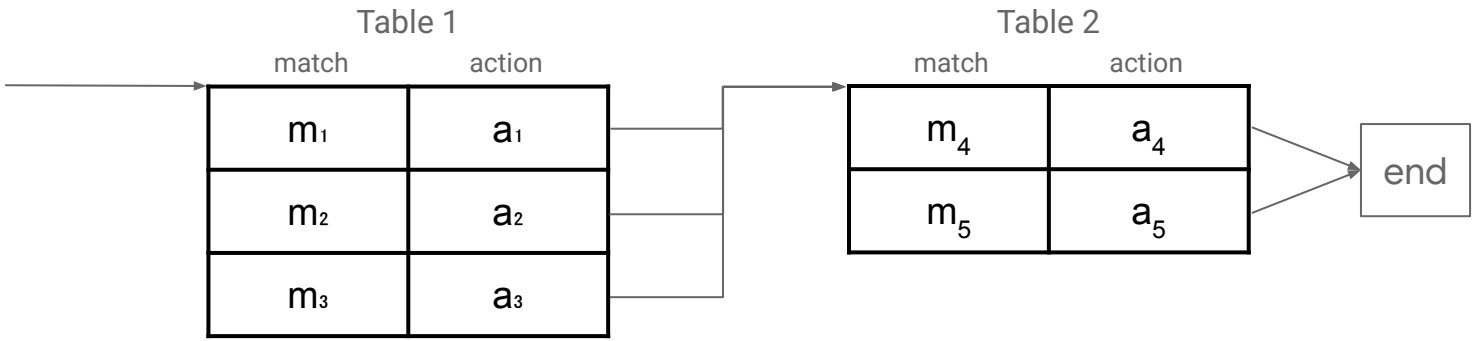
Generate tests that cover **every execution path** through the P4 model of the switch.



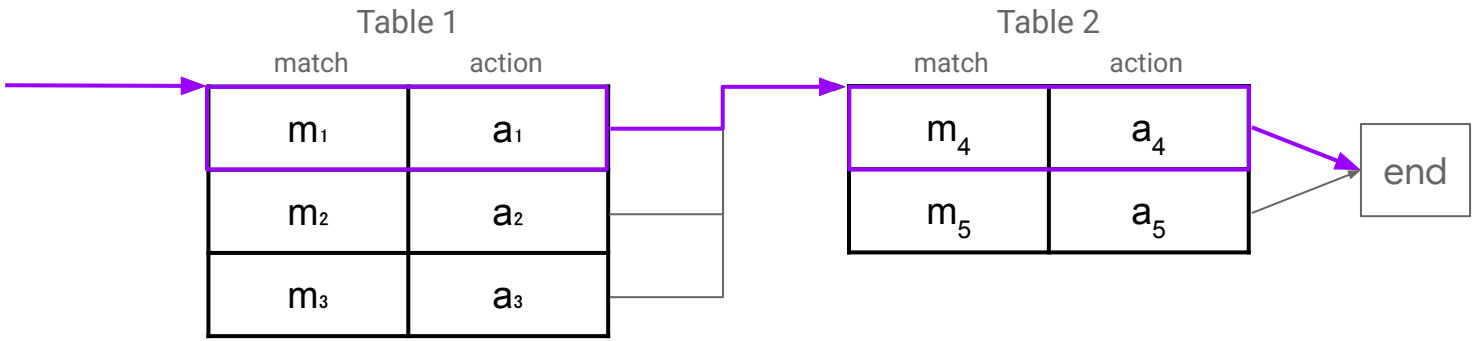
"Ultimate" coverage:

Tests every kind of packet that the switch should handle differently, according to the P4 model.

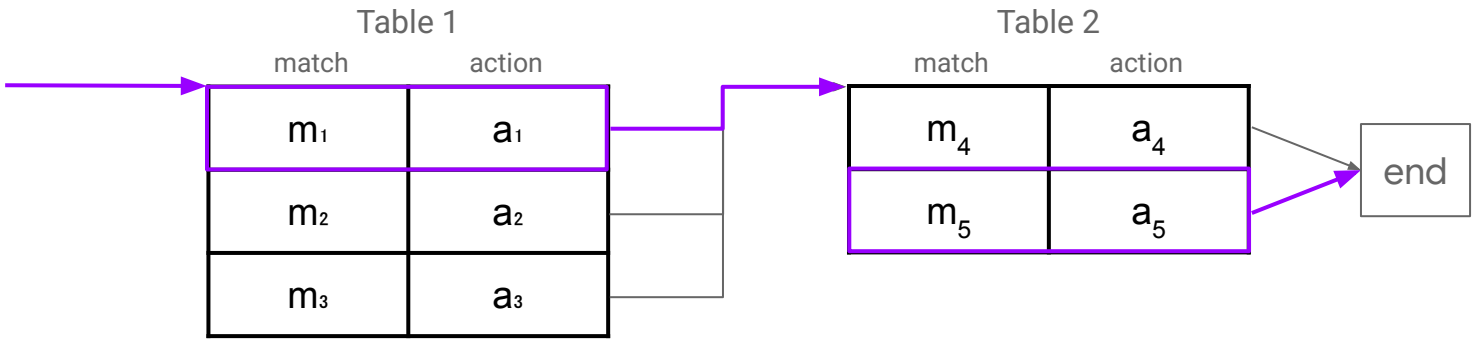
Path Coverage



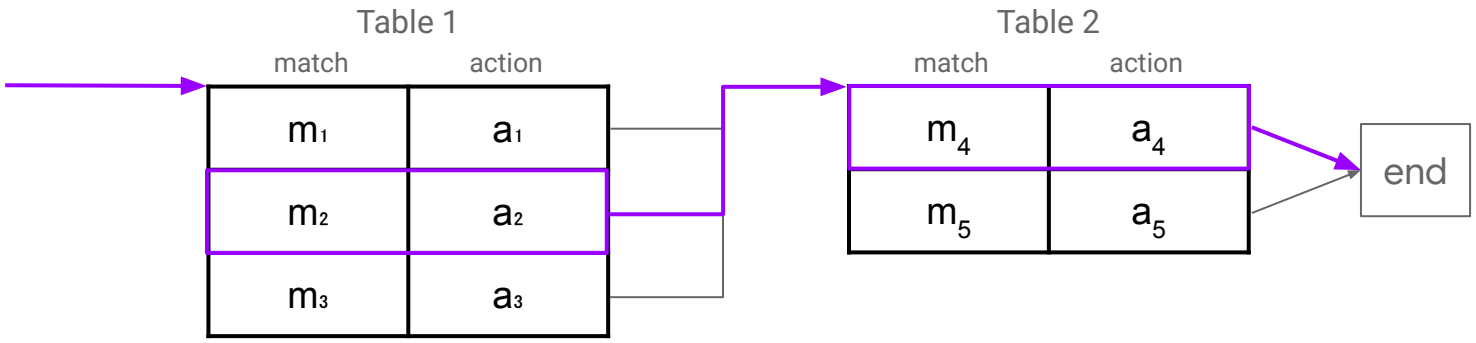
Path Coverage



Path Coverage



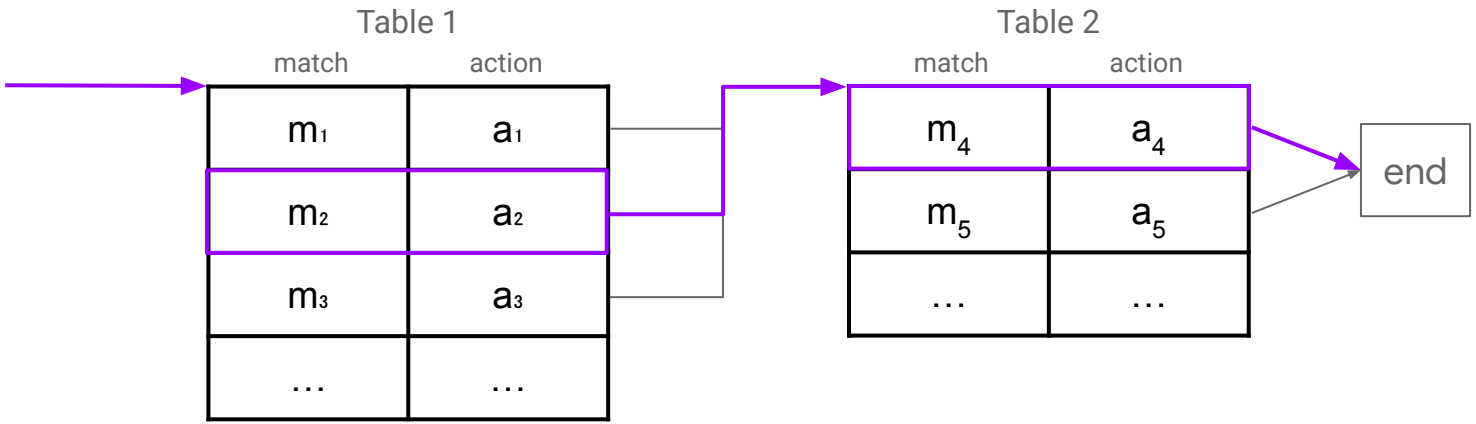
Path Coverage



3 rows x 2 rows

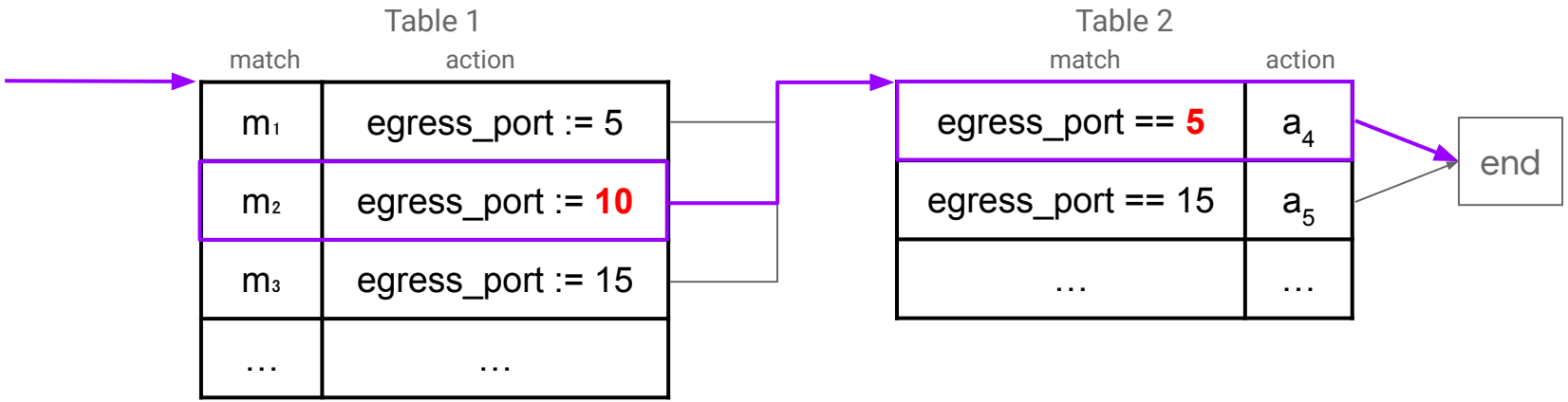
= 6 paths

Path Coverage



= 6M paths!!!

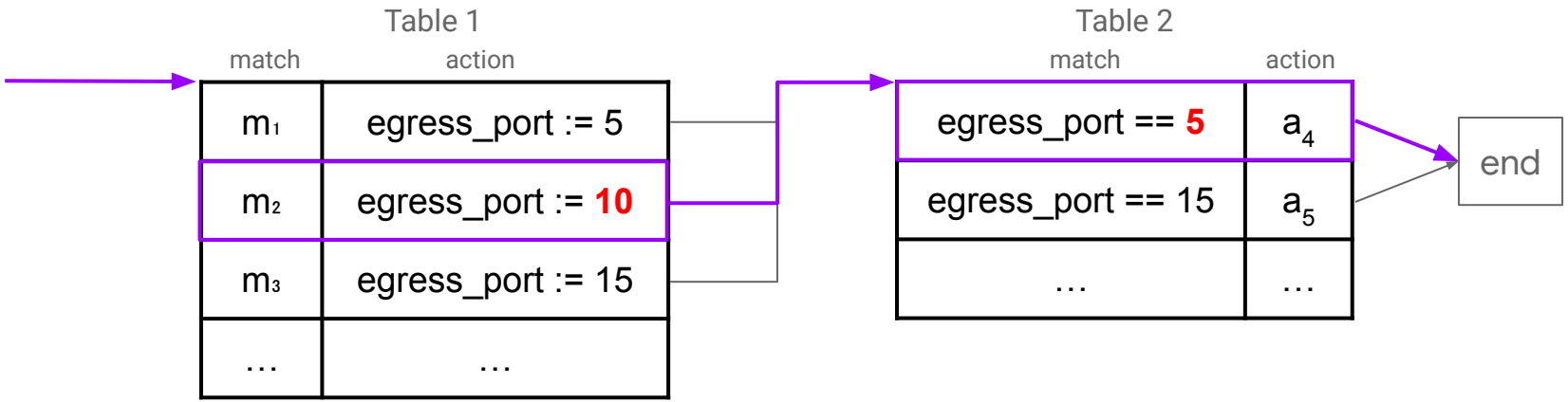
Path Coverage



3,000 rows x 2,000 rows

= 6M paths!!!

Path Coverage



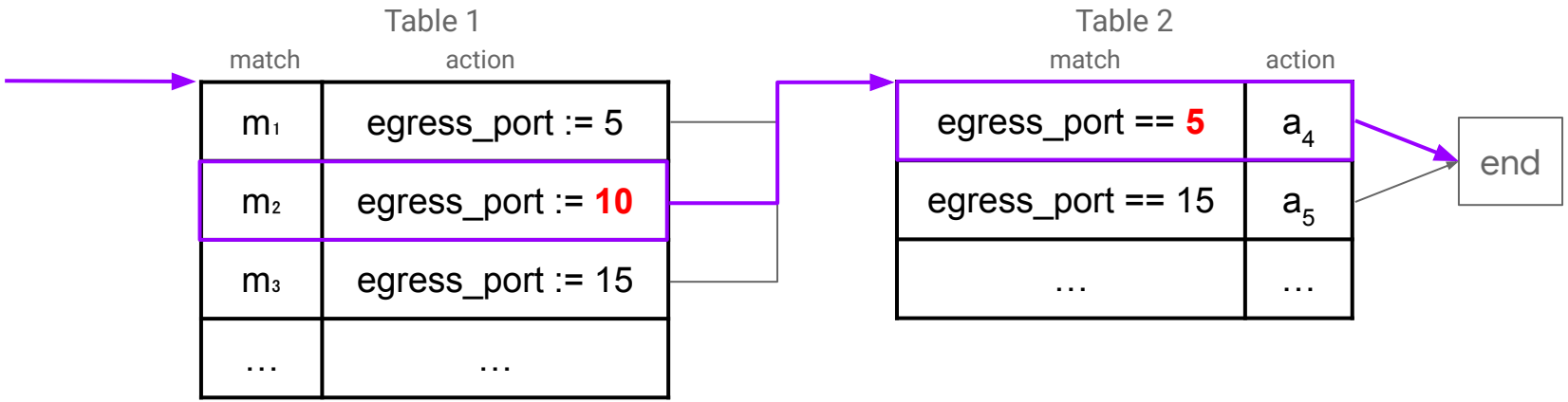
3,000 rows x 2,000 rows

Total paths = 6M

Valid paths << 6M

Path Coverage

Q: How to explore only the valid paths?



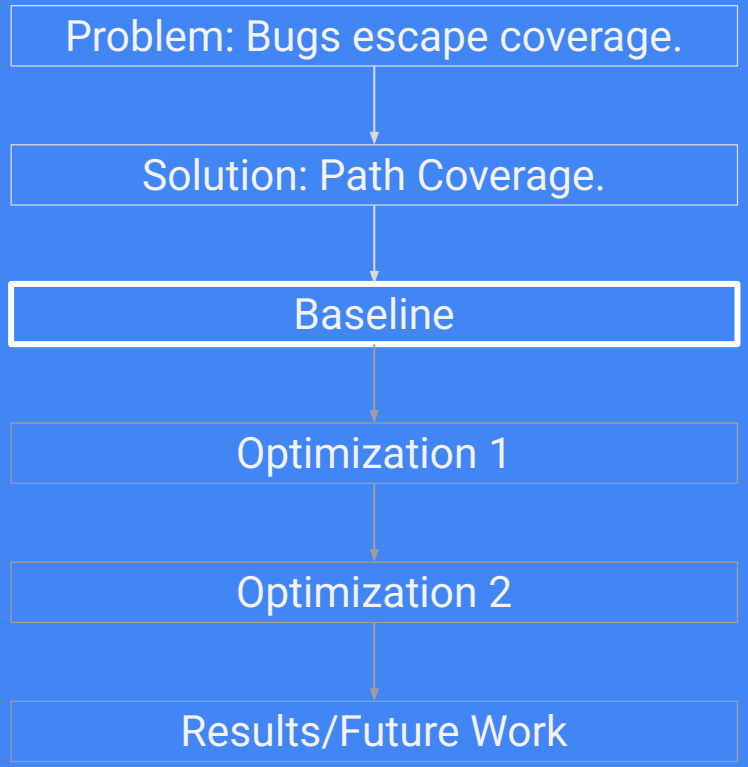
3,000 rows x 2,000 rows

Total paths = 6M

Valid paths << 6M

Baseline:

How do we generate tests that cover all paths?



Baseline: Depth-First Traversal + “Symbolic Execution”

Precondition –
Input packet p takes path if
p.ipv4: 10.0.8.0/24 && *1 == 1* && …

SMT Solver
[z3]

ipv4 Table

match	action
ipv4: 10.0.8.0/24	set_nexthop(1)
ipv4: 10.6.8.0/24	set_nexthop(1)
vrf == 5	set_nexthop(2)
…	…

Nexthop Table

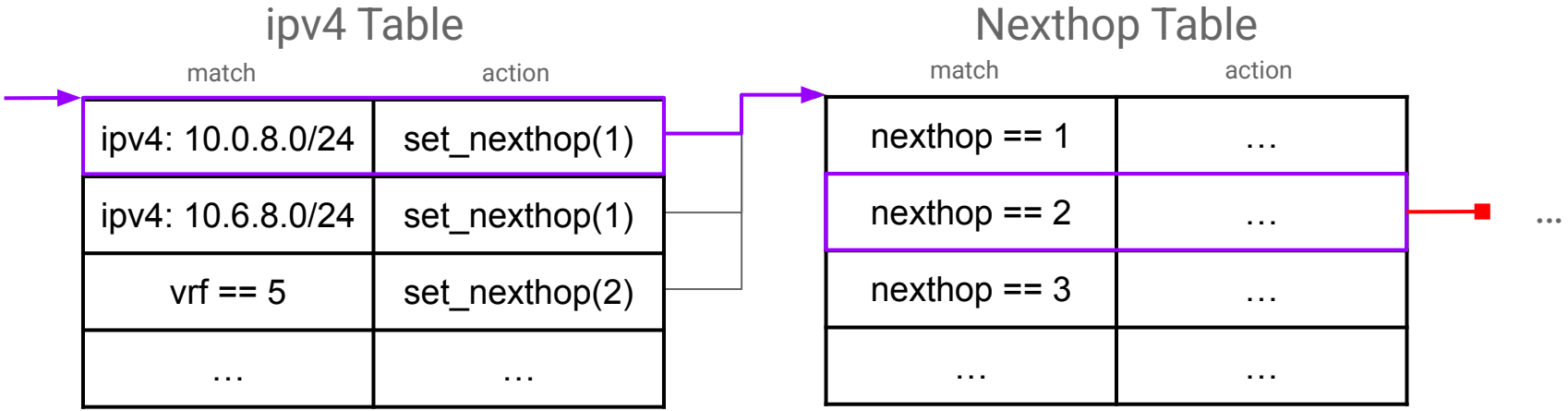
match	action
nexthop == 1	…
nexthop == 2	…
nexthop == 3	…
…	…

…

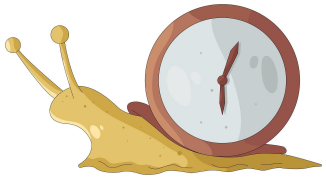
Baseline: Depth-First Traversal + “Symbolic Execution”

Precondition –
Input packet p takes path if
p.ipv4: 10.0.8.0/24 && ***1 == 2***

Do not continue this path!



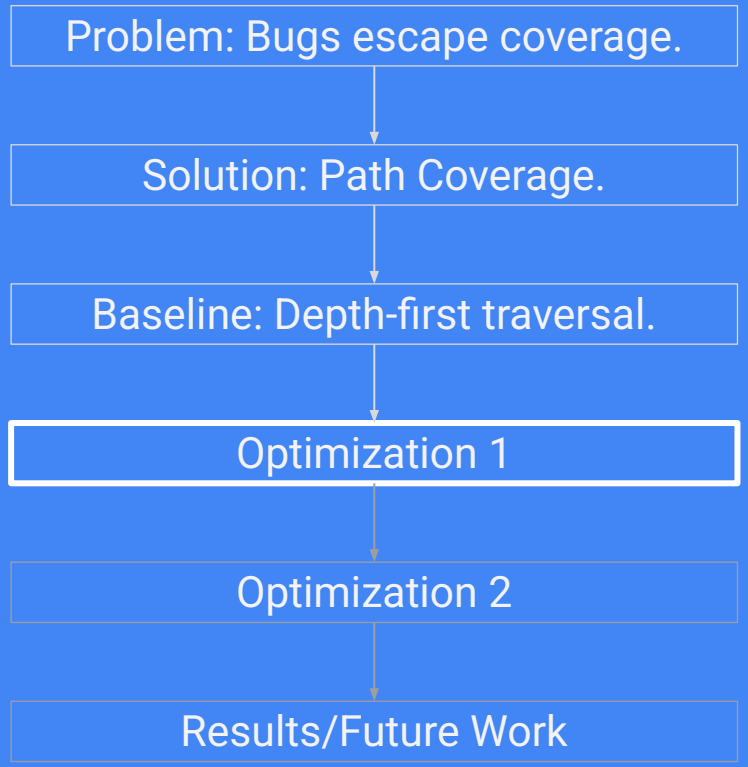
Baseline: Too slow!



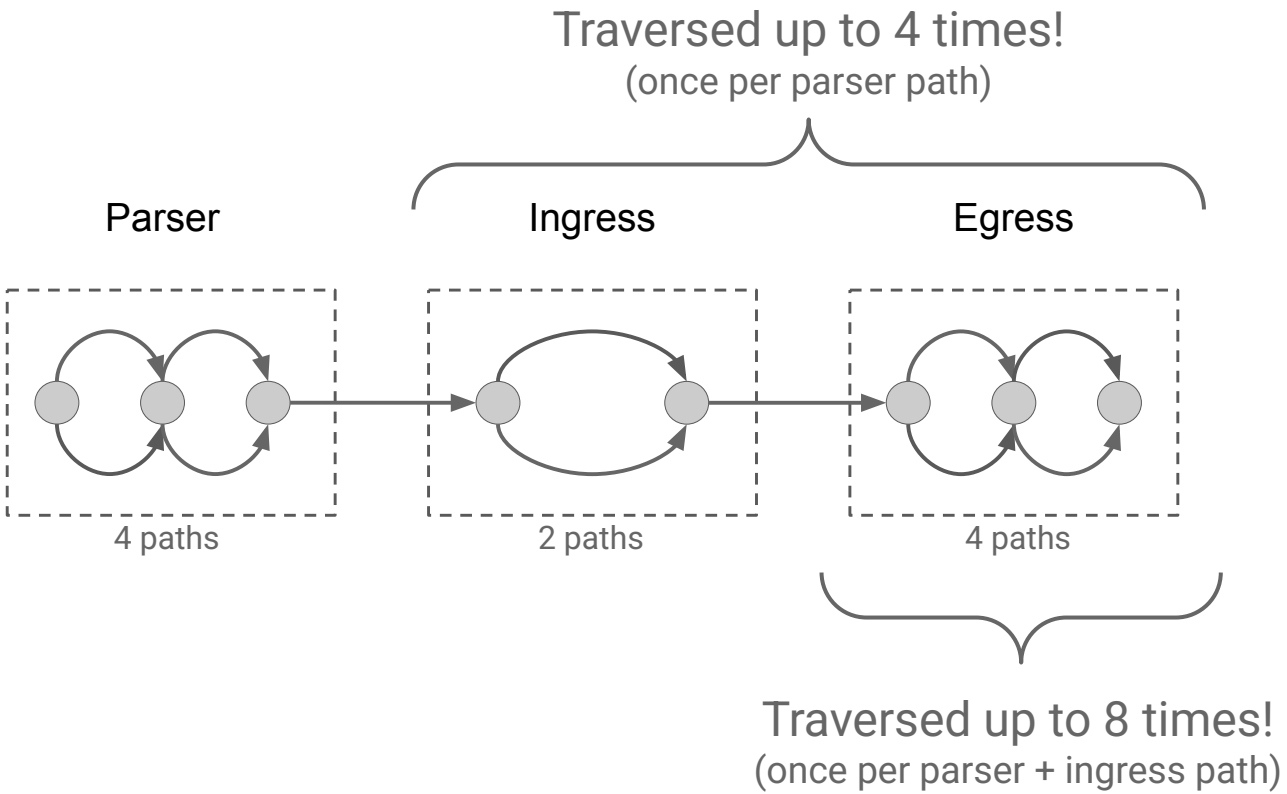
>15 hrs to generate packets for Clos Stage 2

Optimization 1:

Reduce runtime with divide and conquer.

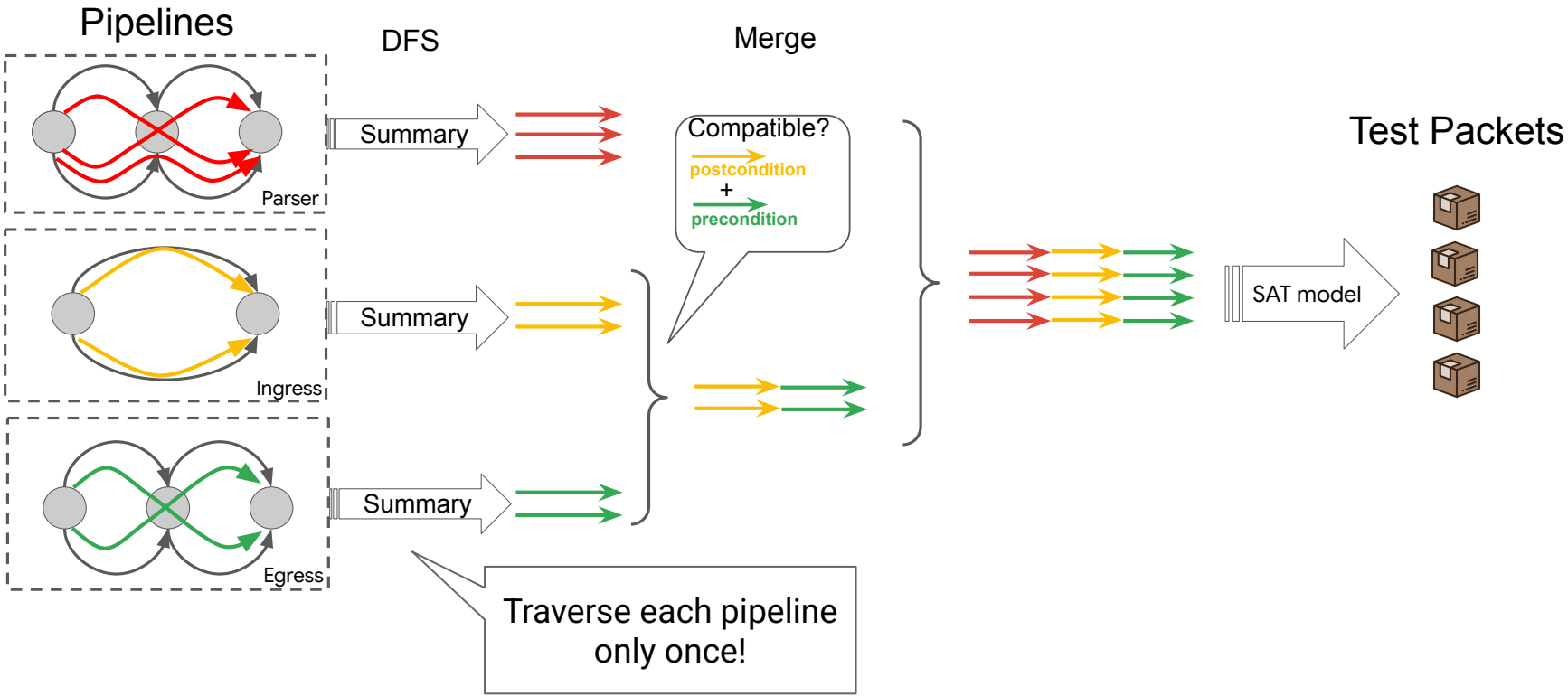


First Optimization: DFS + Pipeline Summaries



*Zheng, Naiqian, et al. "Meissa: scalable network testing for programmable data planes." Proceedings of the ACM SIGCOMM 2022 Conference. 2022.

First Optimization: DFS + Pipeline Summaries



*Zheng, Naiqian, et al. "Meissa: scalable network testing for programmable data planes." Proceedings of the ACM SIGCOMM 2022 Conference. 2022.

First Optimization: DFS + Pipeline Summaries

Time to Generate Path Conditions

Snapshot	# Table Entries	Depth-First Traversal	Depth-First Traversal + Pipeline Summaries	# Packets
Clos Stage 2	~800	⏰ (15h timeout)	**8h**	2M
Clos Stage 3	~800	⏰ (15h timeout)	**2.8h**	860K

Results on other snapshots are mixed.

First Optimization: DFS + Pipeline Summaries

Observations

1. 

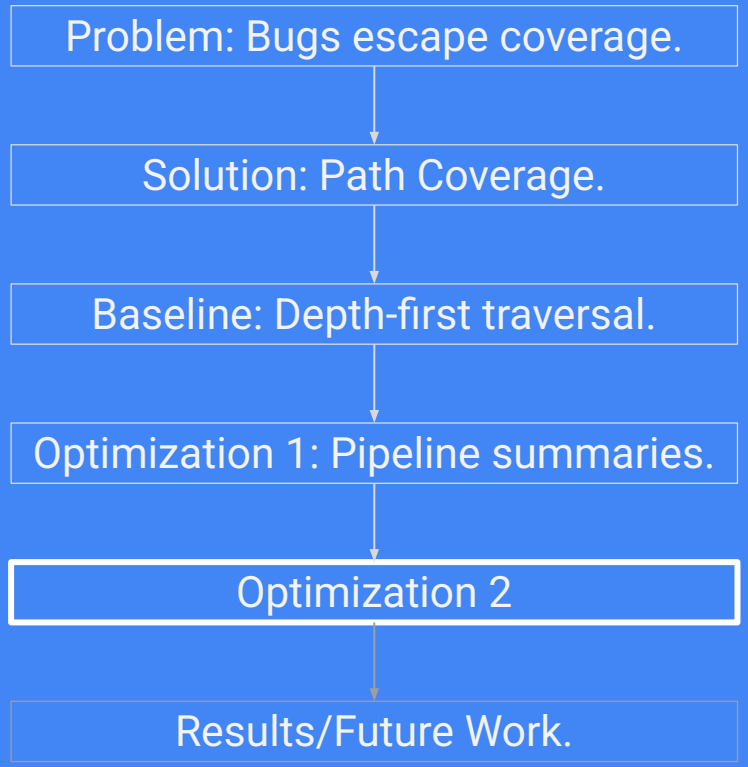
2. **Current infra (simulation+packet injection): > 2 days to run 2M tests**

Time to Generate Path Conditions

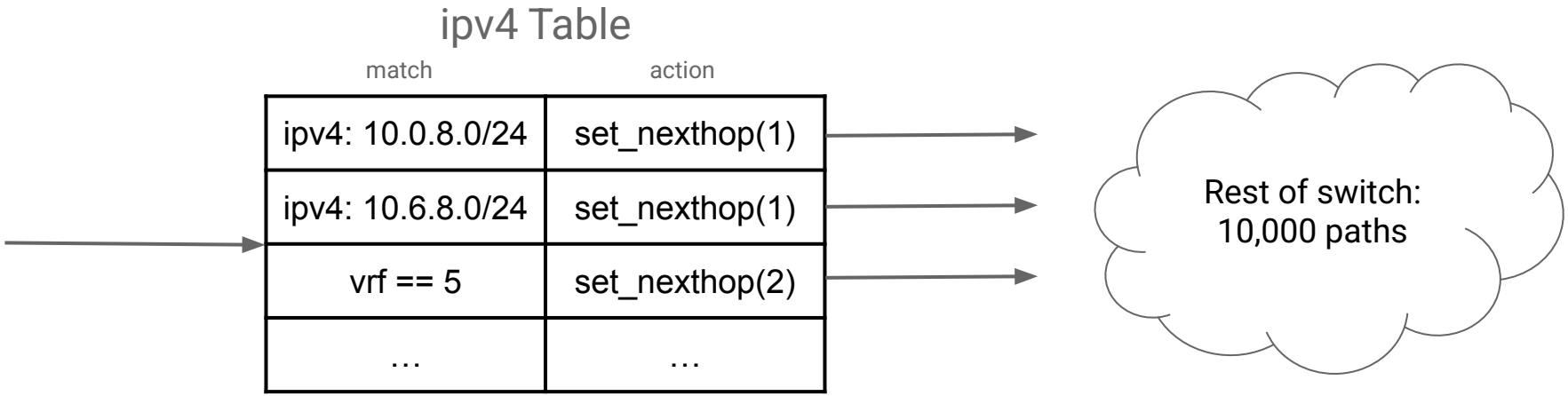
Snapshot	# Table Entries	Depth-First Traversal	Depth-First Traversal + Pipeline Summaries	# Packets
Clos Stage 2	~800	⏰ (15h timeout)	8h	2M
Clos Stage 3	~800	⏰ (15h timeout)	2.8h	860K

Optimization 2:

Reduce number of paths by merging similar table entries.



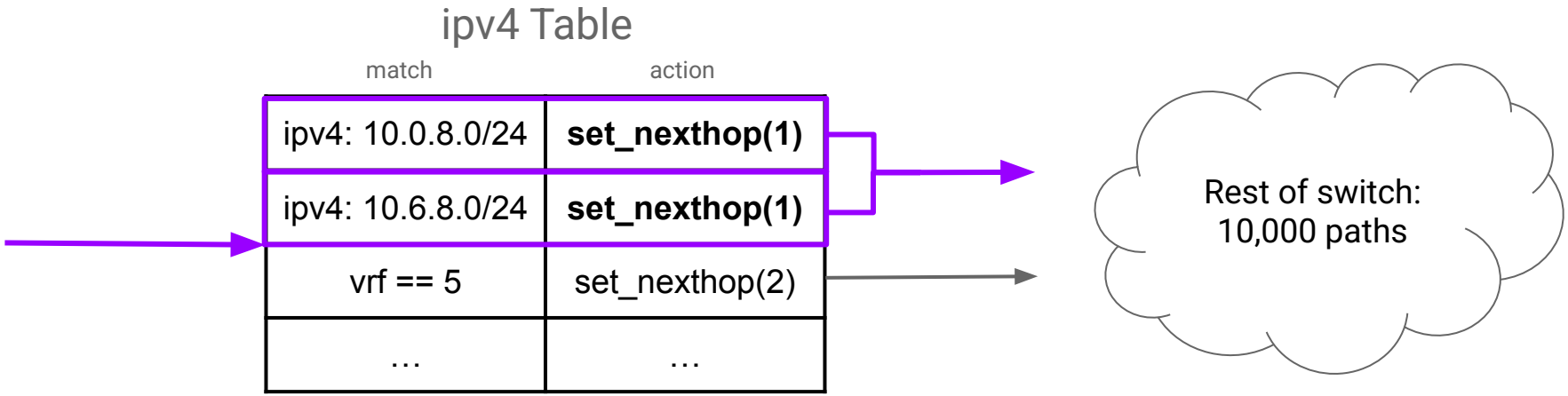
Observation: Some Paths are Near-Identical



1,000 entries * 10,000 later paths

= 10M paths

Observation: Some Paths are Near-Identical

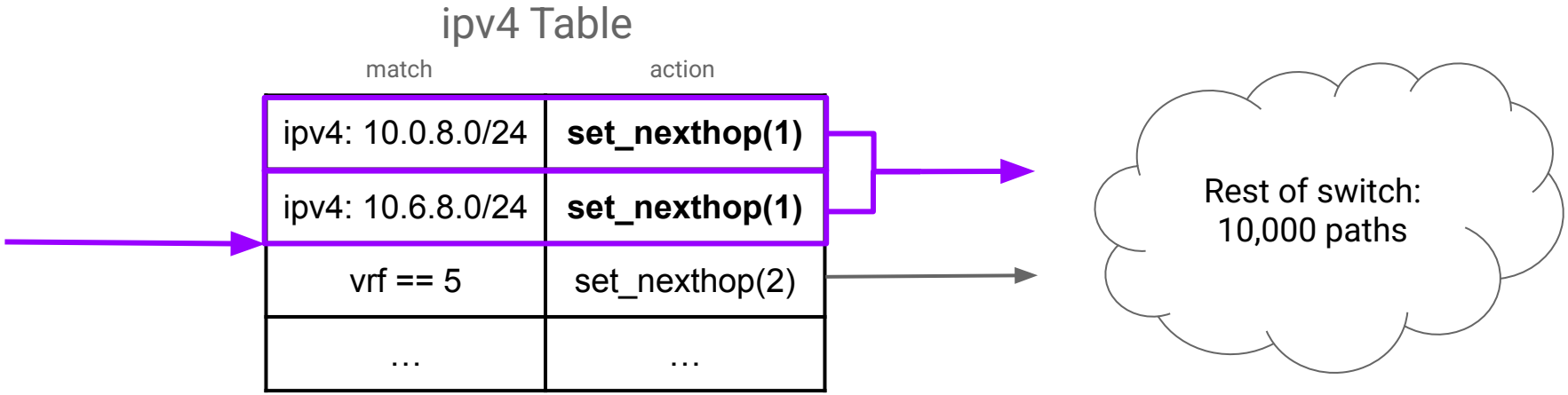


1,000 entries * 10,000 later paths

= 10M paths

Observation: Some Paths are Near-Identical

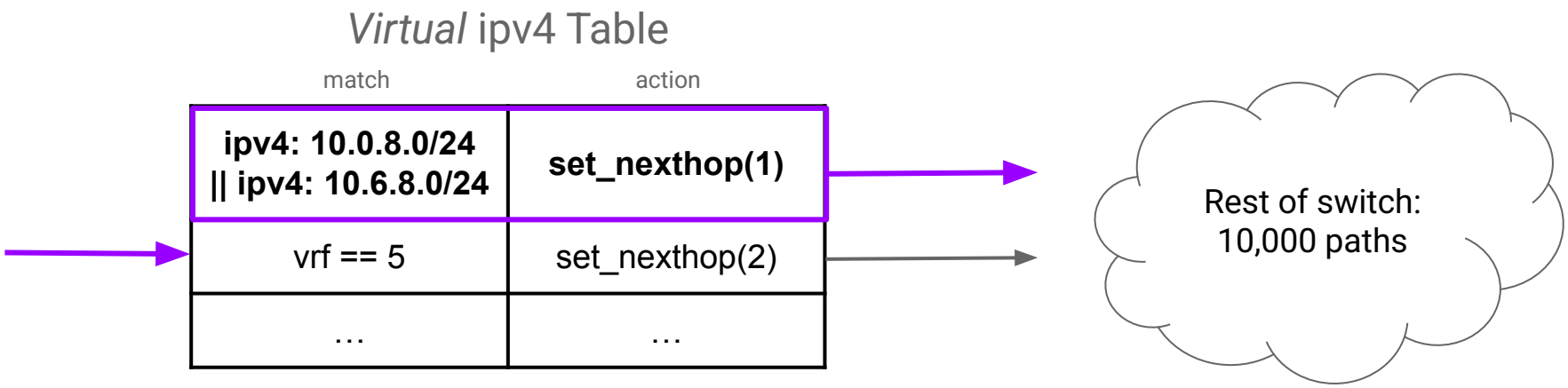
Paths that differ by only “similar” table entries probably trigger the same bugs.



1,000 entries * 10,000 later paths

= 10M paths

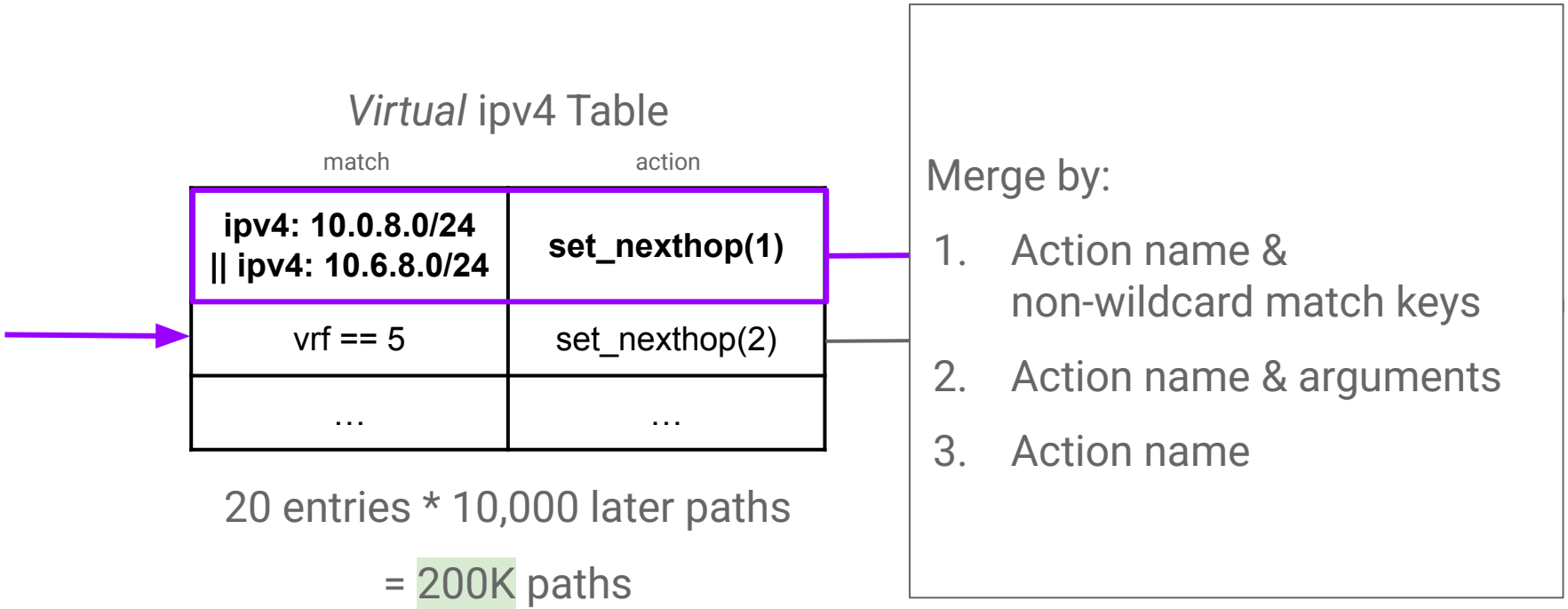
Optimization: Merge “Similar” Table Entries



20 entries * 10,000 later paths

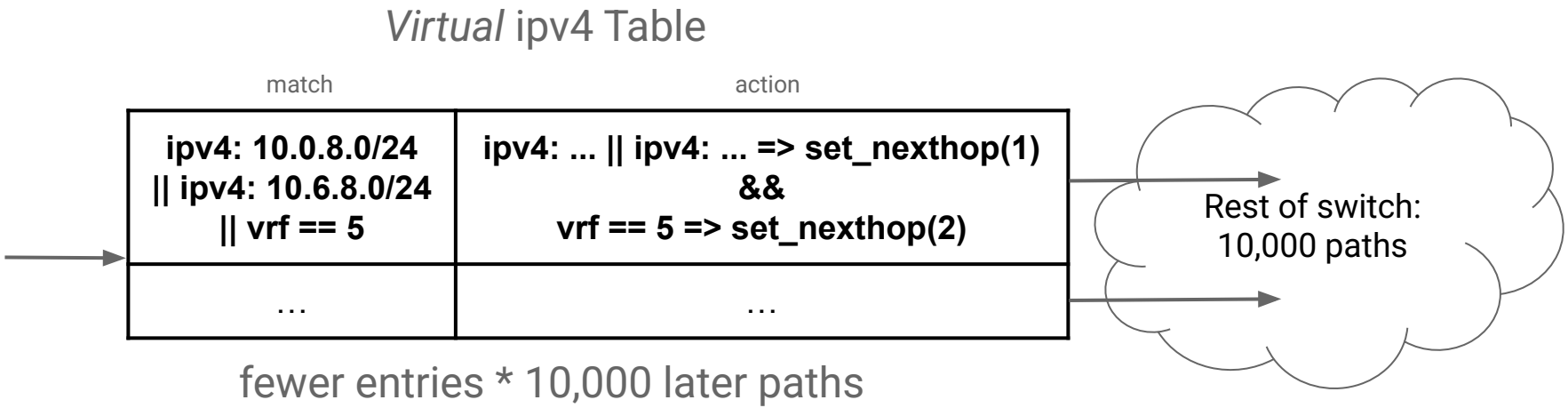
= 200K paths

Optimization: Merge “Similar” Table Entries



Optimization: Merge Similar Table Entries

Option 2: Merge table entries with same action name.

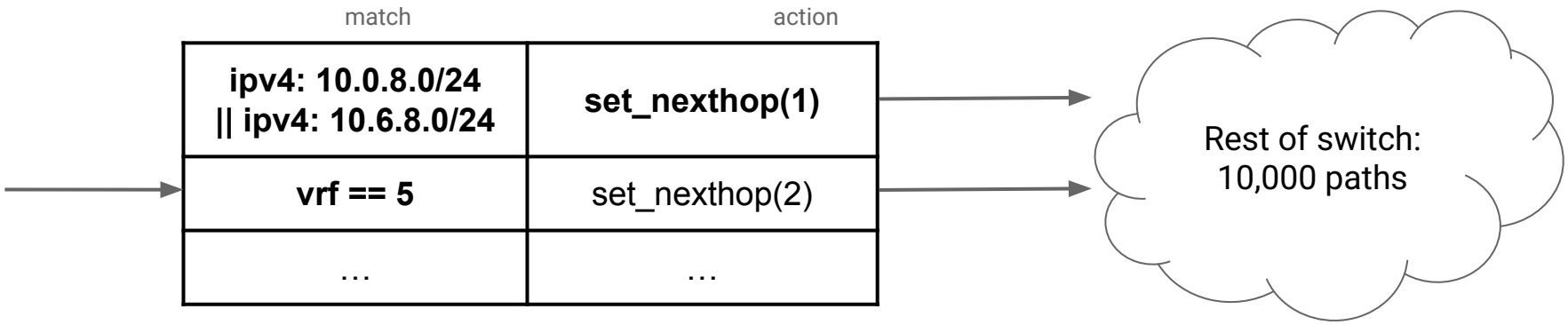


fewer entries * 10,000 later paths

Optimization: Merge Similar Table Entries

Option 3: Merge table entries with the same action name *and* identical non-wildcard match keys.

Virtual ipv4 Table



fewer entries * 10,000 later paths

Optimization: Merge Similar Table Entries

Path Condition Computation on Clos Stage 1 Snapshot

	Merge Strategy	Depth-First Traversal	Depth-First Traversal + Pipeline Summaries	# Test Packets
0.	Never Merge	7.1h	15h+	**1.8M**
1.	Action Names Equal + Match Keys Equal	29.5m	36.5m	37K
2.	Action Names Equal + Params Equal	25.3m	21m	34K
3.	Action Names Equal	10.2m	7m	**16K**

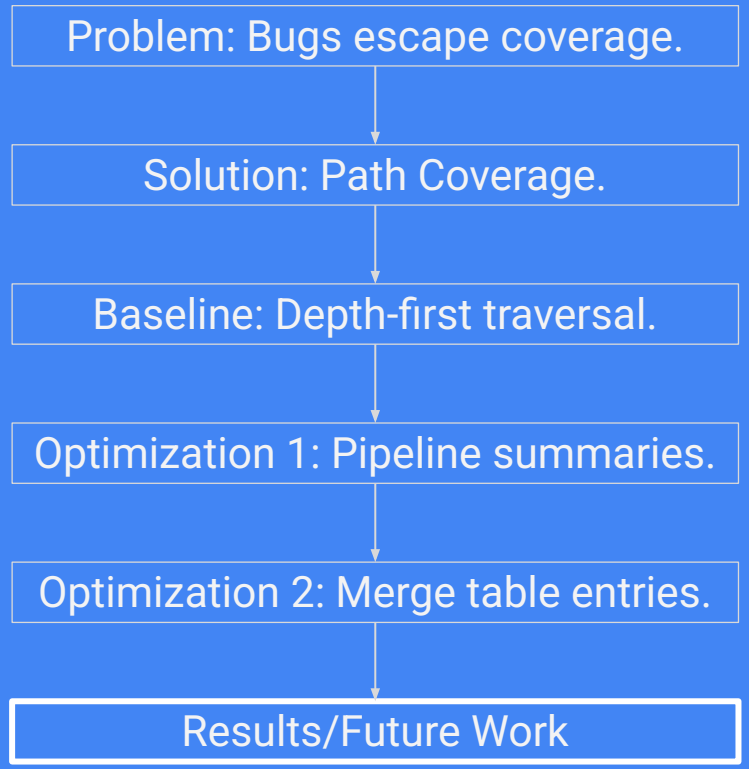
Optimization: Merge Similar Table Entries

Action Name Merging on All Snapshots

Snapshot	# Table Entries	Depth-First Traversal	Depth-First Traversal + Pipeline Summaries	# Test Packets
Clos Stage 2	~800	1.3h	**7m**	20K
Clos Stage 3	~800	26m	**3m**	14K
Border Clos Stage 2	~1100	49m	**14m**	33K
Border Clos Stage 3	~2400	7h	**35m**	39K
Clos Stage 1	~300	10m	**7m**	16K

Results and Future Work:

Cover more, better.



Preliminary Runs Already Find New Potential Bugs!

Discrepancy between SAI-P4 and switch behavior on inner packet with hop_by_hop_options header

(Root causing in progress…)

```
DVaaS Detective: Found the following pattern(s) that match FAILING TEST VECTORS:
- IF nexthop_table == HIT
  && IP protocol == IPv6
  && inner IP protocol == HOPOPT
  && At source location drop_martians.p4(100) == NOT MARKED TO DROP
  && input TTL >= 2
  THEN test vector FAILS
    * coverage: 100%, accounting for 252 out of 252 failing test vectors
    * accuracy: 100%, 252 out of 252 test vectors that match the conditions fail
- All failing test vectors accounted for
```

Future Work – Increased Coverage for Switches and More

- Integrate path coverage with DVaaS
- Reduce runtime with further algorithmic improvements
- Extend P4-based path coverage to network validation

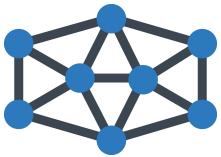
Network Validation – Path Coverage for Networks

Switch

Explore all paths of P4 model

Validate switch dataplane behavior

Network



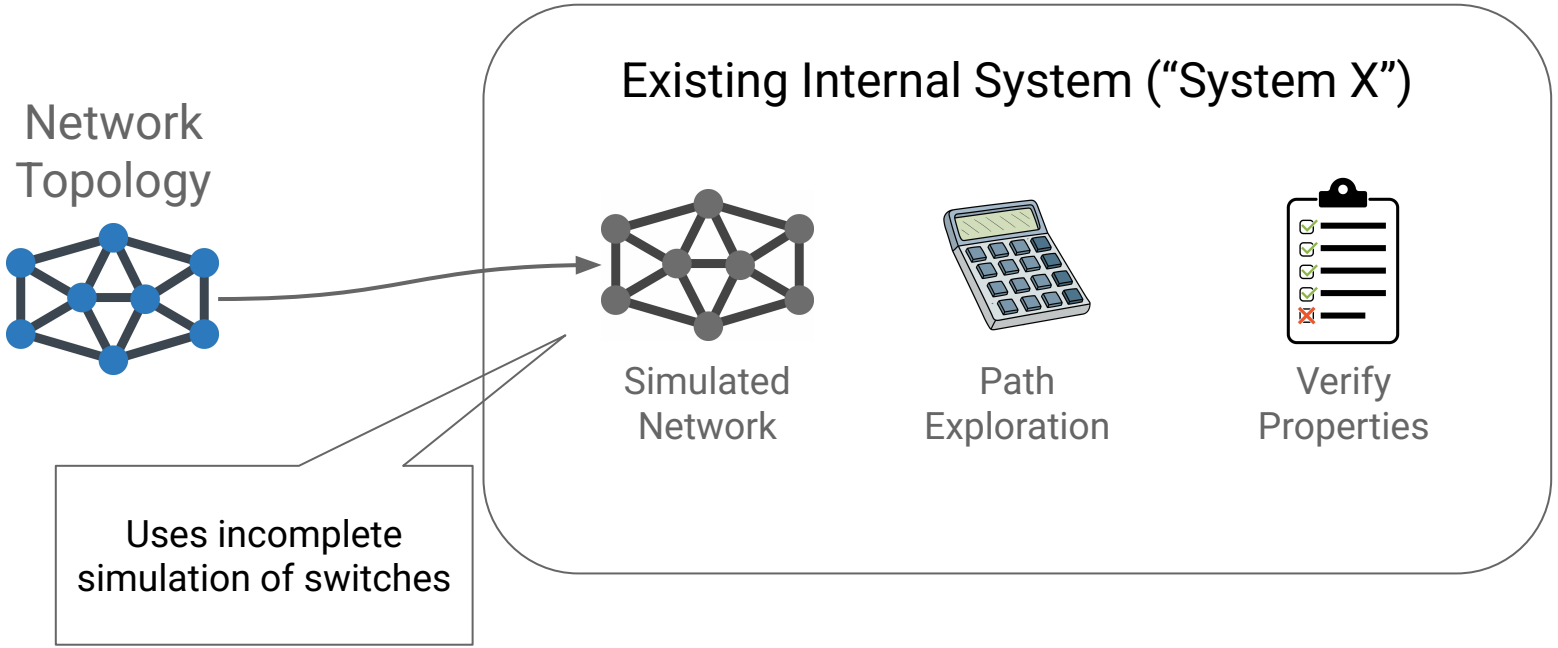
Explore all paths of P4 model

?

Network Validation – Path Coverage for Networks

Q: Does my network obey traffic routing policies?

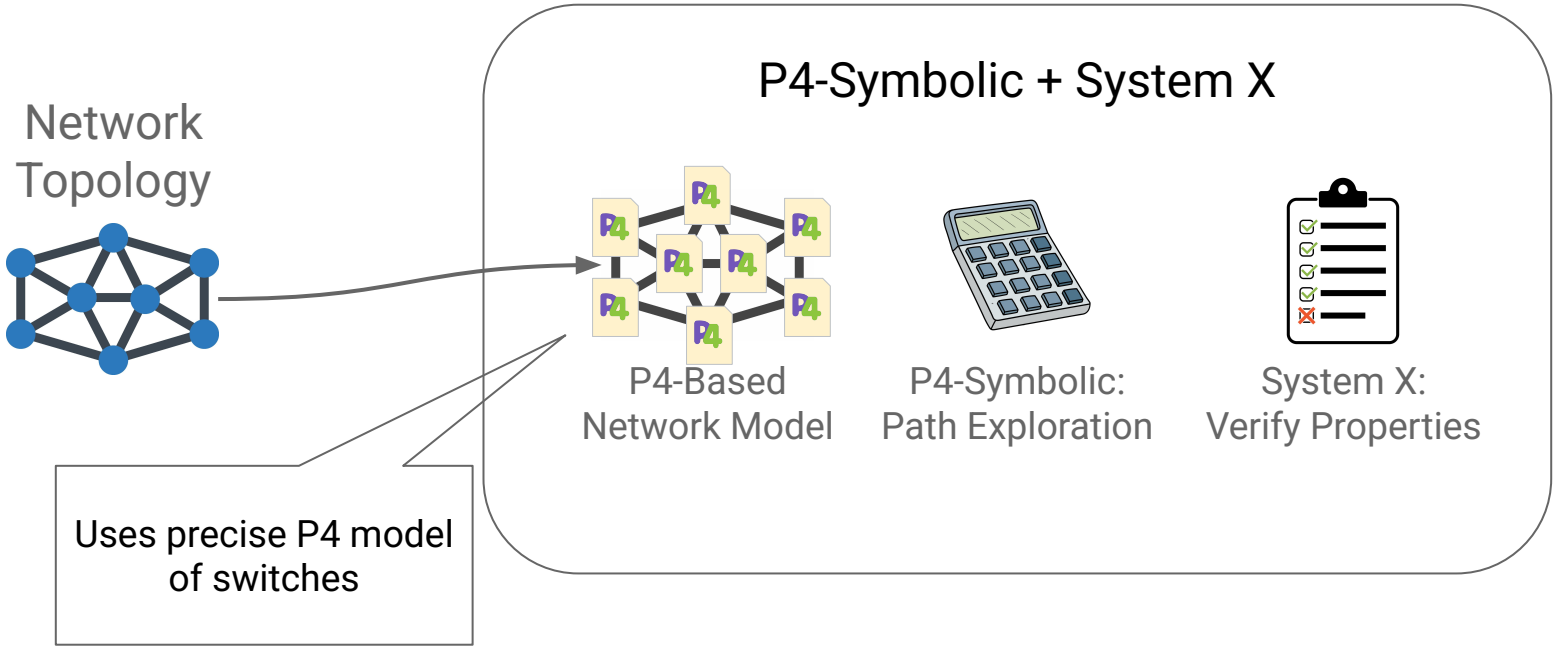
- 2 sources of truth for switch behavior
- Fine-grained switch behavior not modeled



Network Validation – Path Coverage for Networks

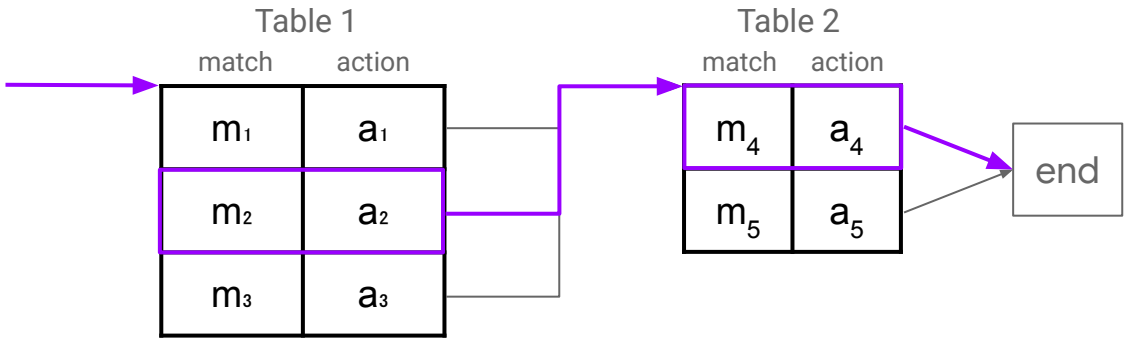
Q: Does my network obey traffic routing policies?

- 1 source of truth for switch behavior
- Models fine-grained switch behavior



Conclusion

- Path coverage for switch dataplane validation
  - Optimizations:
    - Divide and conquer pipelines => Faster test generation
    - Merge similar table entries => Concentrate distinct tests
- Some potential bugs already identified!
- DVaaS integration soon!



Vs Existing Test Generators

	Per-table Entry Coverage	Path Coverage	Synthesizes Table Entries	Speed
P4-Symbolic	✔	✔	✔	⚡
Meissa[1]	✔	✔	×	🚶
P4TestGen[2]	✔	×	✔	⚡
P4PktGen[3]	✔	×	✔	🐢

[1] Zheng, Naiqian, et al. "Meissa: scalable network testing for programmable data planes." Proceedings of the ACM SIGCOMM 2022 Conference. 2022
[2] Ruffy, Fabian, et al. "P4Testgen: An extensible test oracle for P4." Proceedings of the ACM SIGCOMM 2023 Conference. 2023.
[3] Nötzli, Andres, et al. "P4pktgen: Automated test case generation for p4 programs." Proceedings of the Symposium on SDN Research. 2018..

SAI-P4 Pipeline