AALBORG UNIVERSITY

universität wien

AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks

Peter G. Jensen[1], Dan Kristiansen[1], Stefan Schmid[2],

Morten K. Schou[1], Bernhard C. Schrenk[2], Jiří Srba[1]

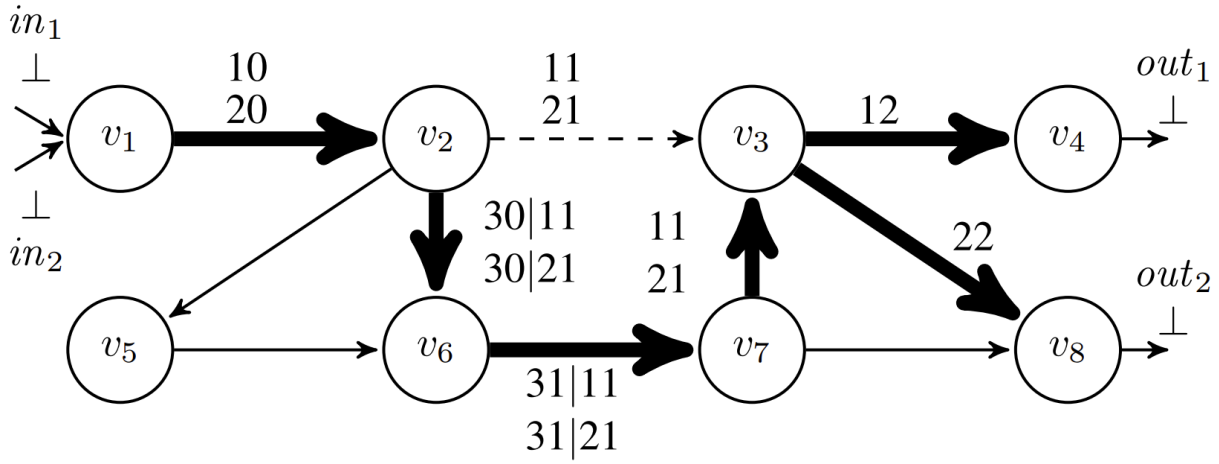
[1]Aalborg University
Denmark

[2]University of Vienna
Austria

AalWiNes = **Aal**borg **Wi**en **Ne**twork verification **s**uite

Motivation

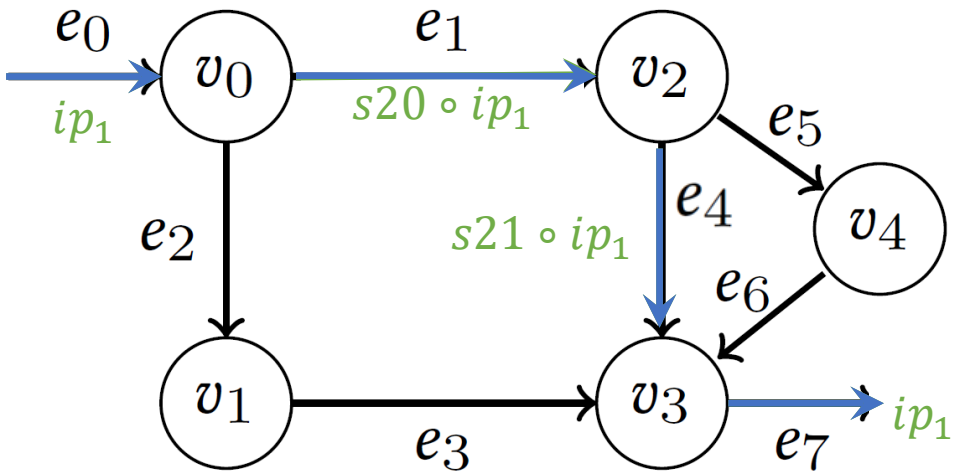
- Operation of MPLS networks is an error prone task
- Hard to reason about behaviour in case of failures (what-if)
- Past work focus on logical properties P-Rex [CoNEXT ’18]
- Quantitative aspects are also important



Our Contribution

- Tool for **dataplane verification** of **MPLS** networks
- Behaviour of **fast reroute** in case of multiple **failures**
- Over-approximation with **polynomial-time** algorithm
- Analyse **quantitative** aspects like latency and hop-count
- Visualize results in a cross-platform GUI

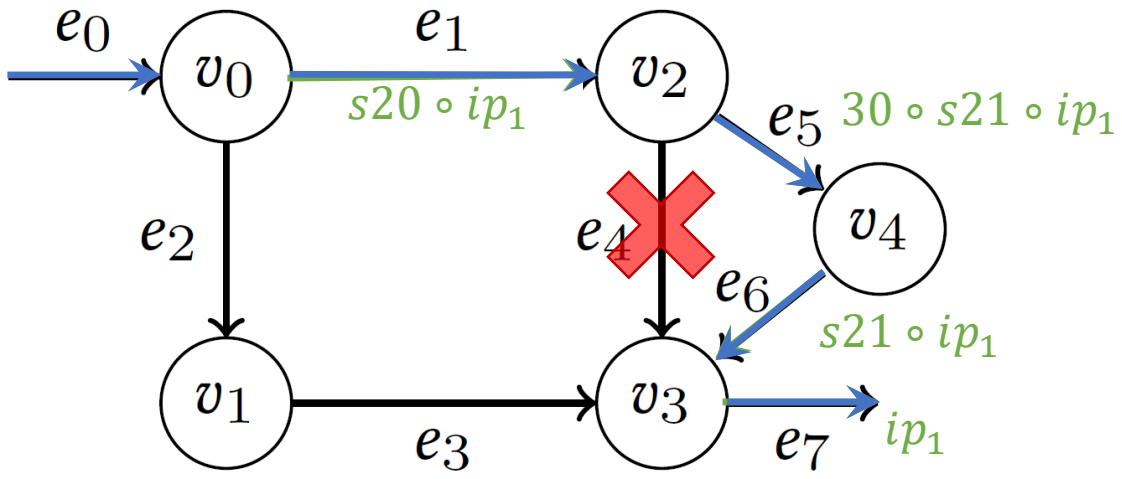
MPLS Network Model



Router	e_{in}	Label	Priority	e_{out}	Operation
v_0	e_0	ip_1	1	e_1	push($s20$)
	e_0	ip_1	1	e_2	push($s10$)
	e_0	$s40$	1	e_1	swap($s41$)
v_1	e_2	$s10$	1	e_3	swap($s11$)
v_2	e_1	$s20$	1	e_4	swap($s21$)
	e_1	$s41$	1	e_5	swap($s42$)
	e_1	$s20$	2	e_5	swap($s21$) $\circ$ push(30)
v_3	e_3	$s11$	1	e_7	pop
	e_4	$s21$	1	e_7	pop
	e_6	$s43$	1	e_7	swap($s44$)
	e_6	$s21$	1	e_7	pop
v_4	e_5	30	1	e_6	pop
	e_5	$s42$	1	e_6	swap($s43$)

$$\sigma_0 = (e_0, ip_1)\ (e_1, s20 \circ ip_1)\ (e_4, s21 \circ ip_1)\ (e_7, ip_1)$$

MPLS Network Model – Link Failure



Router	e_{in}	Label	Priority	e_{out}	Operation
v_0	e_0	ip_1	1	e_1	push($s20$)
	e_0	ip_1	1	e_2	push($s10$)
	e_0	$s40$	1	e_1	swap($s41$)
v_1	e_2	$s10$	1	e_3	swap($s11$)
v_2	e_1	$s20$	1	e_4	swap($s21$)
	e_1	$s41$	1	e_5	swap($s42$)
	e_1	$s20$	2	e_5	swap($s21$) ∘ push(30)
v_3	e_3	$s11$	1	e_7	pop
	e_4	$s21$	1	e_7	pop
	e_6	$s43$	1	e_7	swap($s44$)
	e_6	$s21$	1	e_7	pop
v_4	e_5	30	1	e_6	pop
	e_5	$s42$	1	e_6	swap($s43$)

$$\sigma_0 = (e_0, ip_1)\ (e_1, s20 \circ ip_1)\ (e_4, s21 \circ ip_1)\ (e_7, ip_1)$$
$$\sigma_1 = (e_0, ip_1)\ (e_1, s20 \circ ip_1)\ (e_5, 30 \circ s21 \circ ip_1)(e_6, s21 \circ ip_1)\ (e_7, ip_1)$$
$$\sigma_2 = (e_0, ip_1)\ (e_2, s10 \circ ip_1)\ (e_3, s11 \circ ip_1)\ (e_7, ip_1)$$
$$\sigma_3 = (e_0, s40 \circ ip_1)\ (e_1, s41 \circ ip_1)\ (e_5, s42 \circ ip_1)\ (e_6, s43 \circ ip_1)\ (e_7, s44 \circ ip_1)$$

Query Language

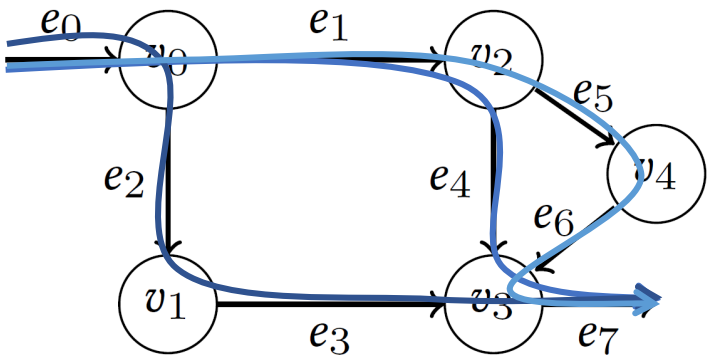
- Query language: $\langle a \rangle\ b\ \langle c \rangle\ k$
  - Regular expressions
    - a: Initial header
    - b: Path of links
    - c: Final header

- Does there exist a set of failed links F s.t. $|F| \leq k$, where a trace satisfies $\langle a \rangle\ b\ \langle c \rangle$?

Query Language

- Examples of properties

				Witness traces
• Reachability:	$\langle \text{ip} \rangle$	$[.\#v_0] \; .^* \; [v_3\#.]$	$\langle \text{ip} \rangle$ 0	σ_0, σ_2
• Waypointing:	$\langle \text{ip} \rangle$	$[.\#v_0] \; .^* \; [.\#v_4] \; .^* \; [v_3\#.]$	$\langle \text{ip} \rangle$ 1	σ_1
• Avoid edge:	$\langle \text{ip} \rangle$	$[.\#v_0] \; [\hat{} \; v_2\#v_3]^* \; [v_3\#.]$	$\langle \text{ip} \rangle$ 2	σ_1, σ_2
• Transparency:	$\langle s40 \; \text{ip} \rangle$	$[.\#v_0] \; .^* \; [v_3\#.]$	$\langle \text{mpls}^+ \; \text{smpls} \; \text{ip} \rangle$ 0	no



$$\sigma_0 = (e_0, ip_1)\ (e_1, s20 \circ ip_1)\ (e_4, s21 \circ ip_1)\ (e_7, ip_1)$$
$$\sigma_1 = (e_0, ip_1)\ (e_1, s20 \circ ip_1)\ (e_5, 30 \circ s21 \circ ip_1)(e_6, s21 \circ ip_1)\ (e_7, ip_1)$$
$$\sigma_2 = (e_0, ip_1)\ (e_2, s10 \circ ip_1)\ (e_3, s11 \circ ip_1)\ (e_7, ip_1)$$
$$\sigma_3 = (e_0, s40 \circ ip_1)\ (e_1, s41 \circ ip_1)\ (e_5, s42 \circ ip_1)\ (e_6, s43 \circ ip_1)\ (e_7, s44 \circ ip_1)$$

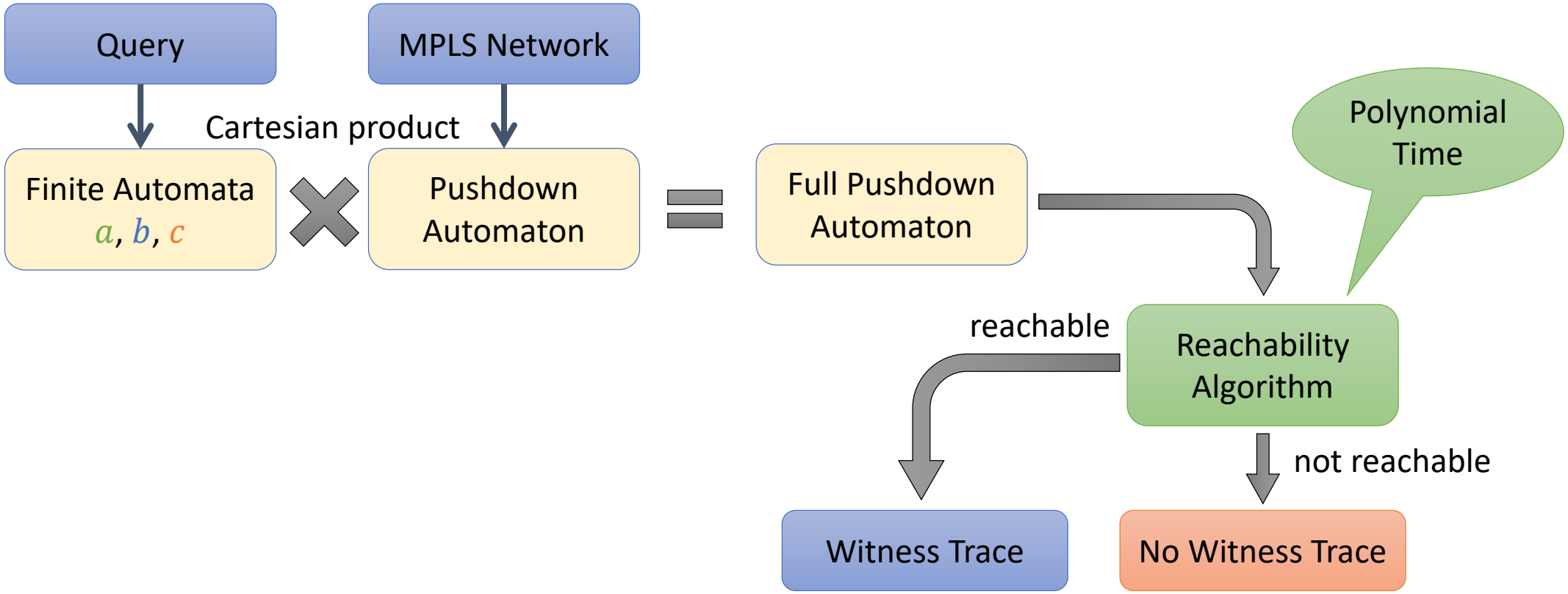
Quantitative Extension

- If query is satisfied, find trace that minimizes:
  - Hops
  - Latency (based on a latency value per link)
  - Failures
  - Tunnels

- Combinations: $expr ::= p \mid a * expr \mid expr_1 + expr_2$
  $(expr_1, expr_2, \ldots, expr_n)$

- Example: $(Hops, Failures + 3 \cdot Tunnels)$

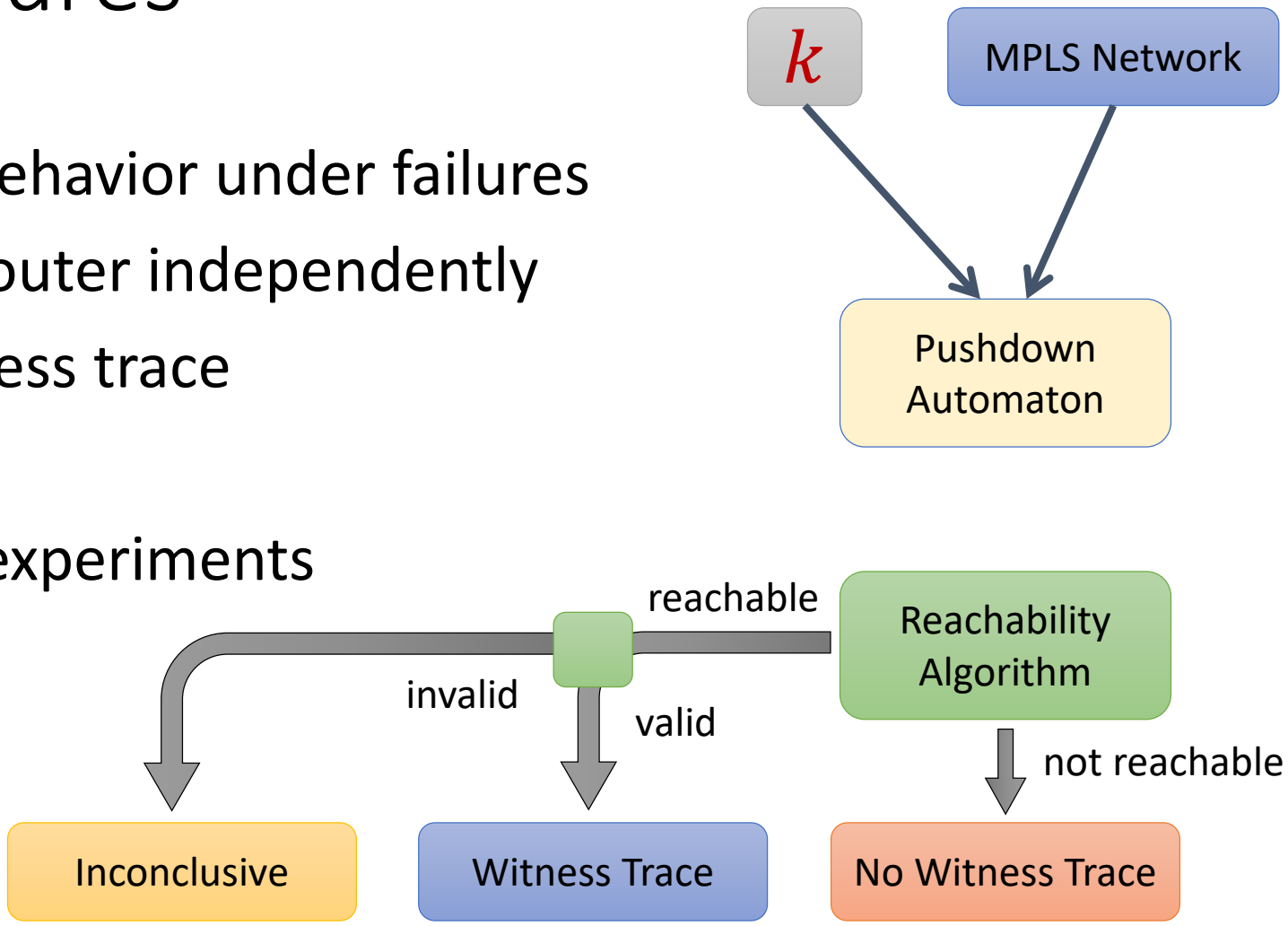
Pushdown Automata Reachability



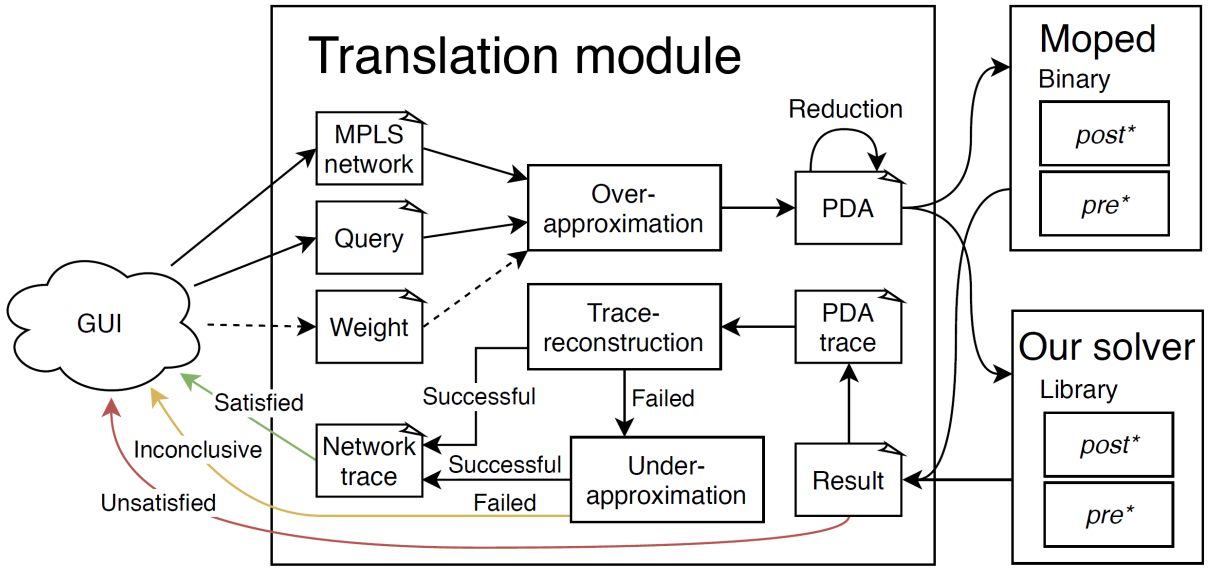
Handling Link Failures

- **Over-approximation** of behavior under failures
- Allow k failures at each router independently
- **Check soundness** of witness trace

- Inconclusive in $< 1\%$ of experiments



Implementation + Demo



Online demo: https://demo.aalwines.cs.aau.dk/

Source code: https://github.com/DEIS-Tools/AalWiNes

Performance Evaluation

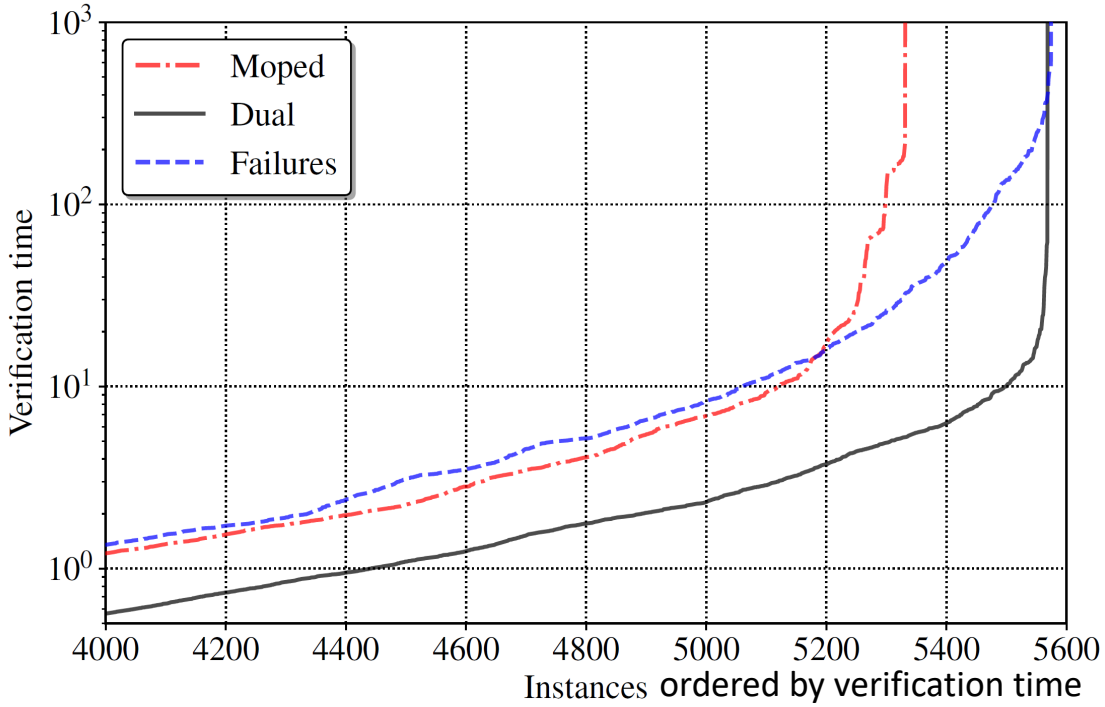
- Real-world network with 31 routers and $> 250\,000$ forwarding rules
- Specific queries requested by network operator

Query	Moped	Dual	Failures
$\langle \texttt{smpls ip} \rangle\ [\cdot\ \#R6]\ \cdot^*\ [\cdot\ \#R4]\ \langle \texttt{smpls ip} \rangle\ 1$	9.57	0.82	41.23
$\langle \texttt{smpls ip} \rangle\ [\cdot\ \#R2]\ \cdot^*\ [\cdot\ \#R18]\ \langle (\texttt{mpls}^*\ \texttt{smpls})?\ \texttt{ip} \rangle\ 1$	9.29	0.86	31.76
$\langle \texttt{ip} \rangle\ [\cdot\ \#R0]\ \cdot^*\ [\cdot\ \#R4]\ \langle \texttt{ip} \rangle\ 0$	0.88	0.01	0.02
$\langle [\$449550]\ \texttt{ip} \rangle\ [\cdot\ \#R0]\ \cdot^*\ [\cdot\ \#R5]\ \cdot^*\ [\cdot\ \#R1]\ \langle \texttt{ip} \rangle\ 0$	1.66	0.02	0.03
$\langle [\$449550]\ \texttt{ip} \rangle\ [\cdot\ \#R0]\ \cdot^*\ [\cdot\ \#R5]\ \cdot^*\ [\cdot\ \#R1]\ \langle \texttt{ip} \rangle\ 1$	6.08	0.05	0.06
$\langle \texttt{smpls? ip} \rangle\ \cdot^*\ \langle \cdot\ \texttt{smpls ip} \rangle\ 0$	89.37	14.73	432.66

- Our solver (‘Dual’) outperforms state-of-the-art modelchecker Moped
- Acceptable overhead of quantitative analysis (‘Failures’)

Performance Evaluation

- Synthetic networks (based on Topology Zoo)
- More than 5600 experiments



- Quantitative analysis ('Failures') has fewer inconclusive cases