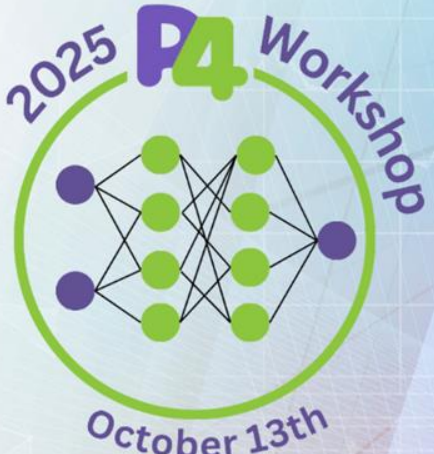


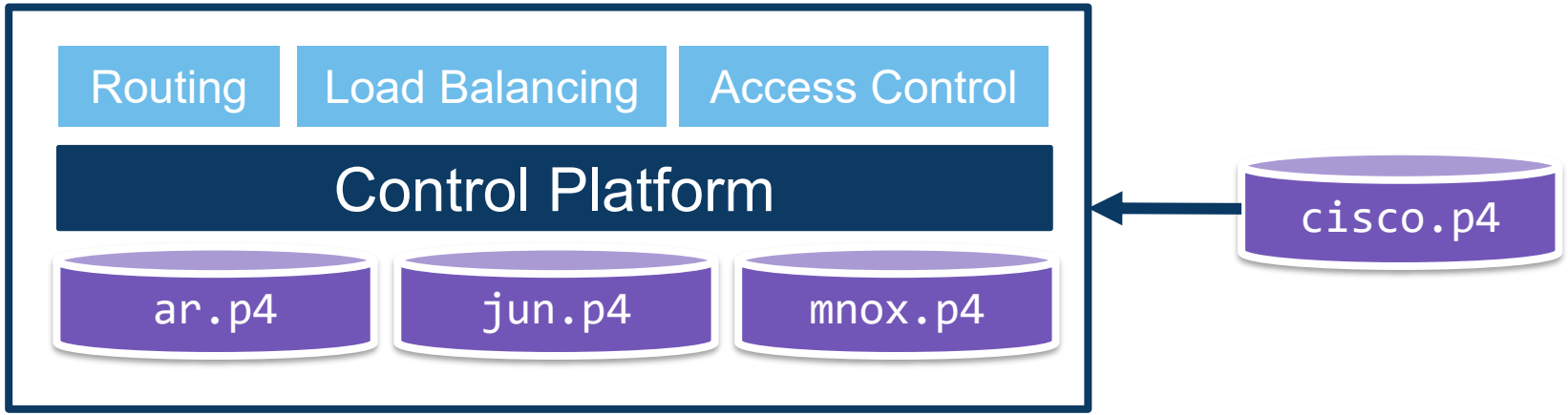
When P4 isn't Enough:
Specifying the Control Plane Interface

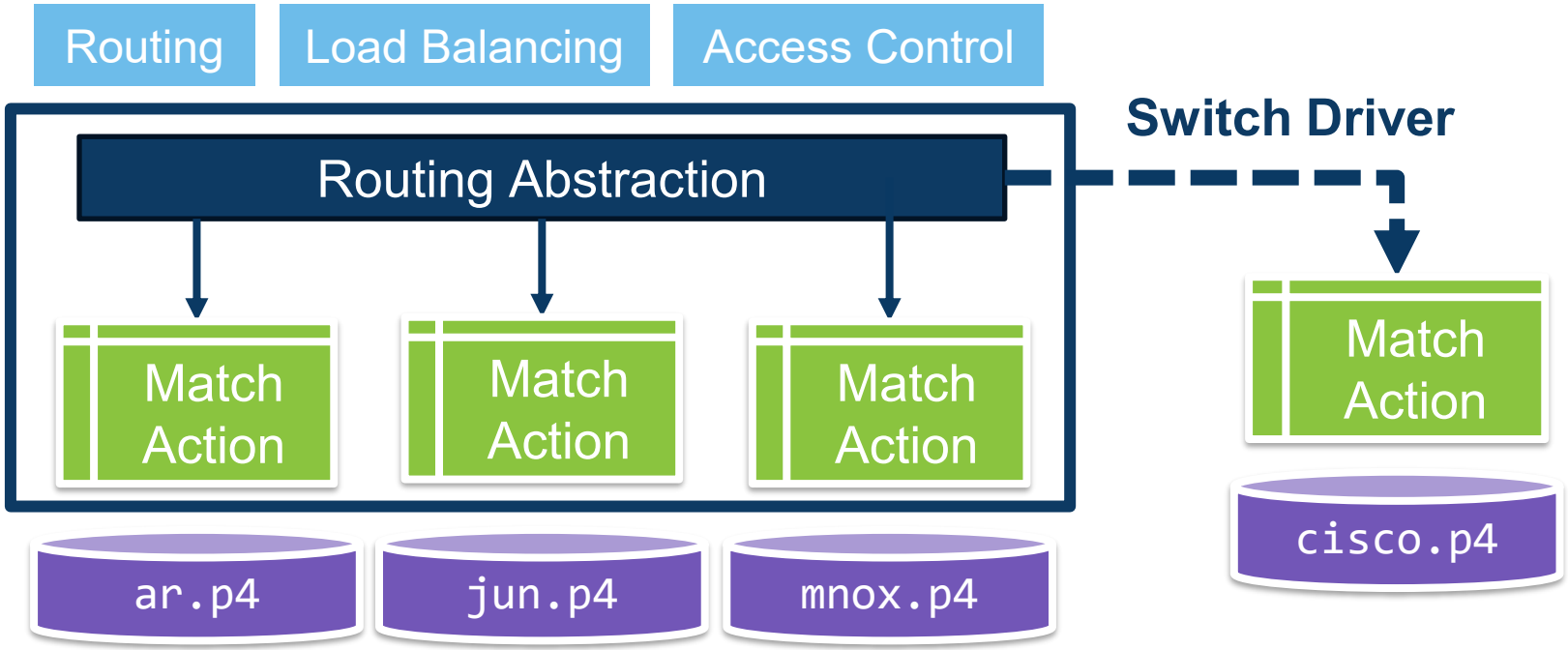
Eric Hayden Campbell
UT Austin

THE LINUX FOUNDATION

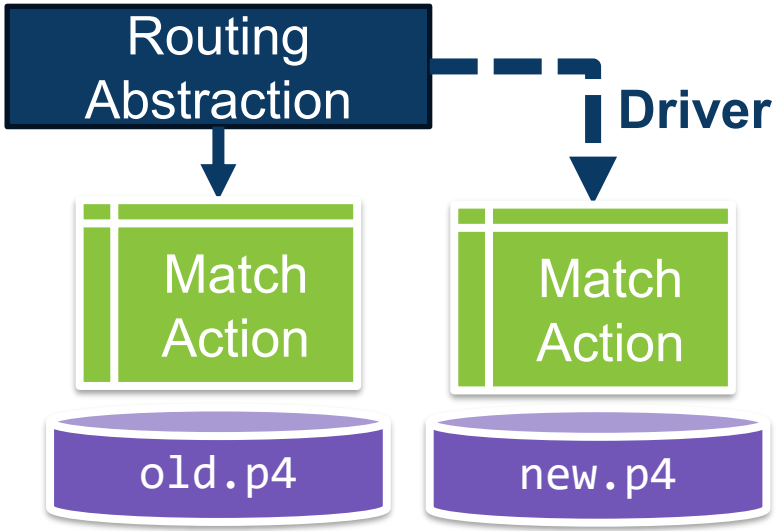
A “Simple” Problem

Software Defined Network




Routing
Load Balancing
Access Control
Routing Abstraction
Match Action
Match Action
Match Action
ar.p4
jun.p4
mnox.p4
Switch Driver
Match Action
cisco.p4

What can we know *in advance*?



Safety

Compatibility

Safety

```
table l2 {
  keys = {
      ether.type : exact;
      ipv4.dst : ternary;
      ipv6.dst : ternary; }
  actions =
    { route; drop; NoAction; }
}
```

Control Specs

```
table l2 {
  keys =
    { ether.type : exact;
      ipv4.dst : ternary;
      ipv6.dst : ternary; }
  actions =
    { route; drop; NoAction; }
}
```

```
ether.type = 0x86DD ⇒
ipv4.dst  = DONT_CARE
```

```
ether.type = 0x0800 ⇒
ipv6.dst  = DONT_CARE
```

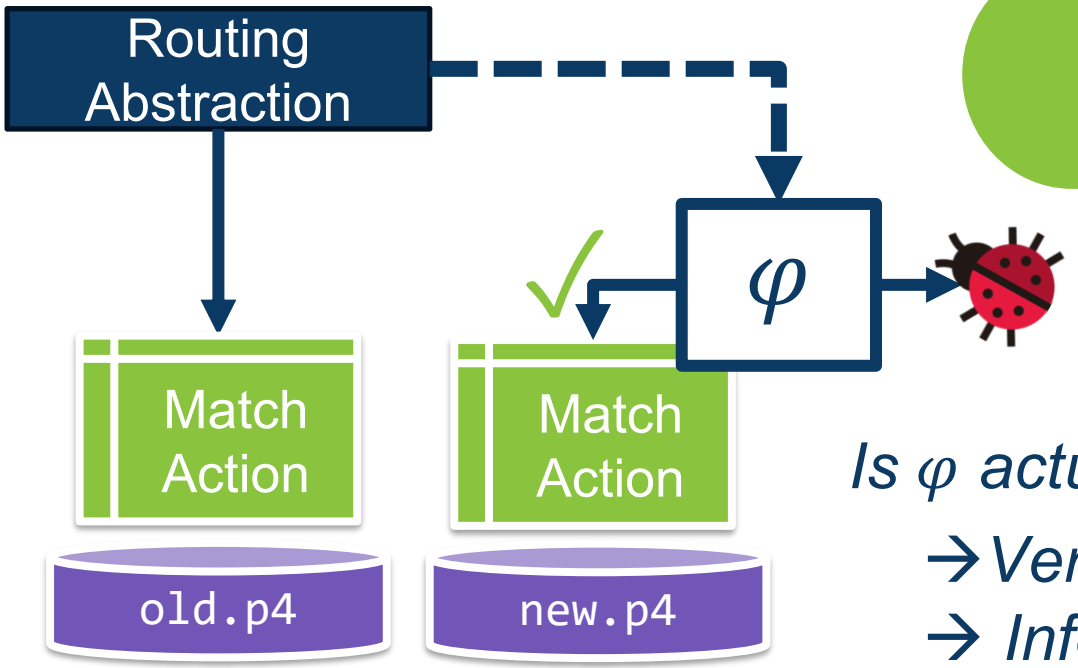
Checking Control Specs

```
ether.type = 0x86DD ⇒
ipv4.dst  = DONT_CARE
```

```
ether.type = 0x0800 ⇒
ipv6.dst  = DONT_CARE
```

type	ipv4.dst	ipv6.dst	action
0x0800	10.0.1.10	*	route(99)
0x86DD	*	DECA:FBAD::	route(32)
0x0800	*	00DD:BA11::	route(88)
*	*	*	drop

THE LINUX FOUNDATION



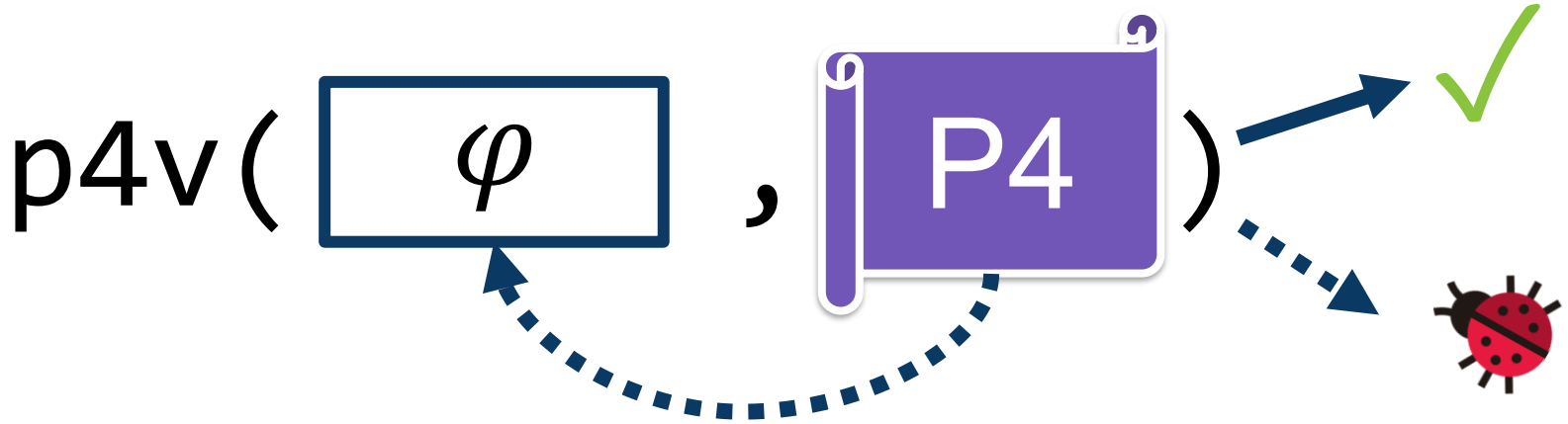
Safety

Is φ actually safe?

→ *Verification*

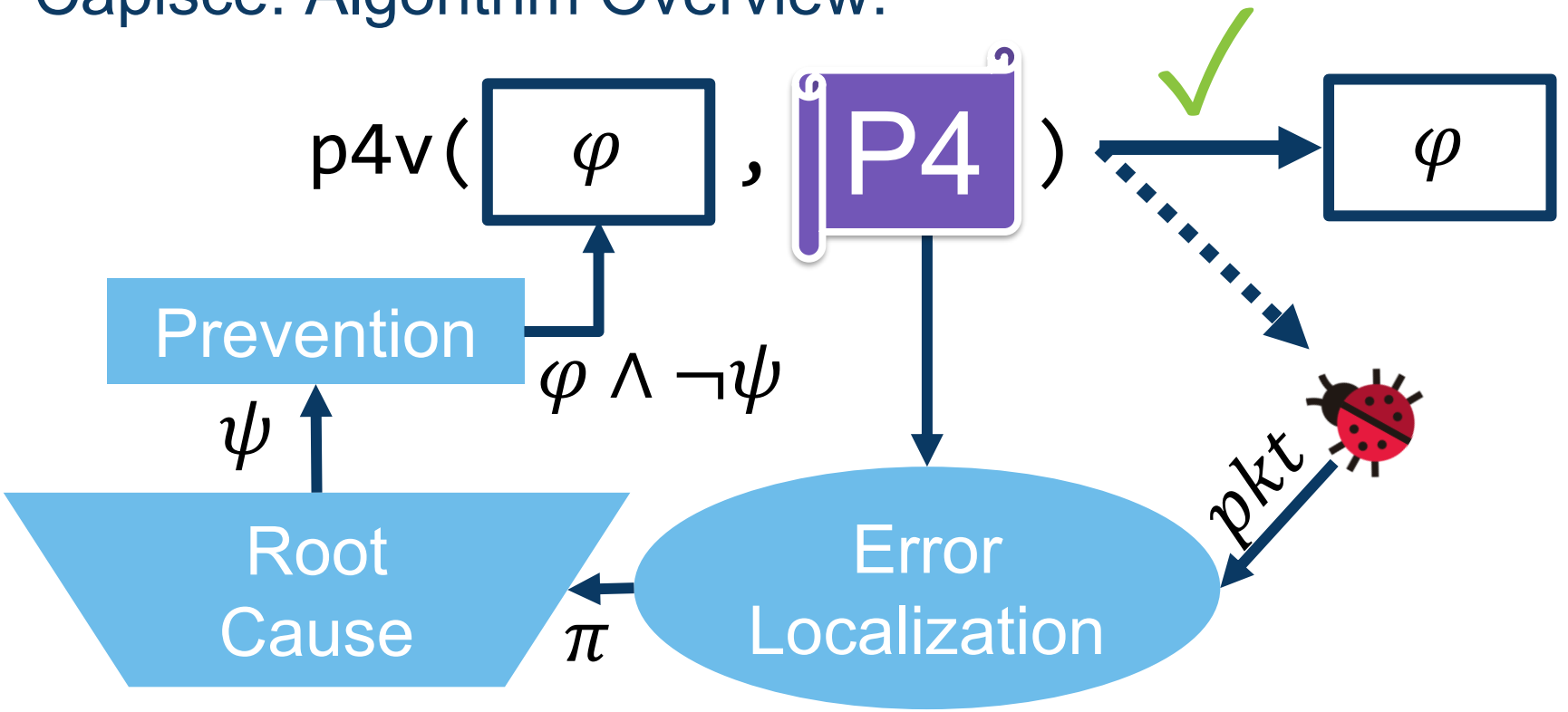
→ *Inference*

P4 Verification [Liu et al. SIGCOMM ‘18]



Inference of Control Specifications [OOPSLA' 24]

Capisce: Algorithm Overview:

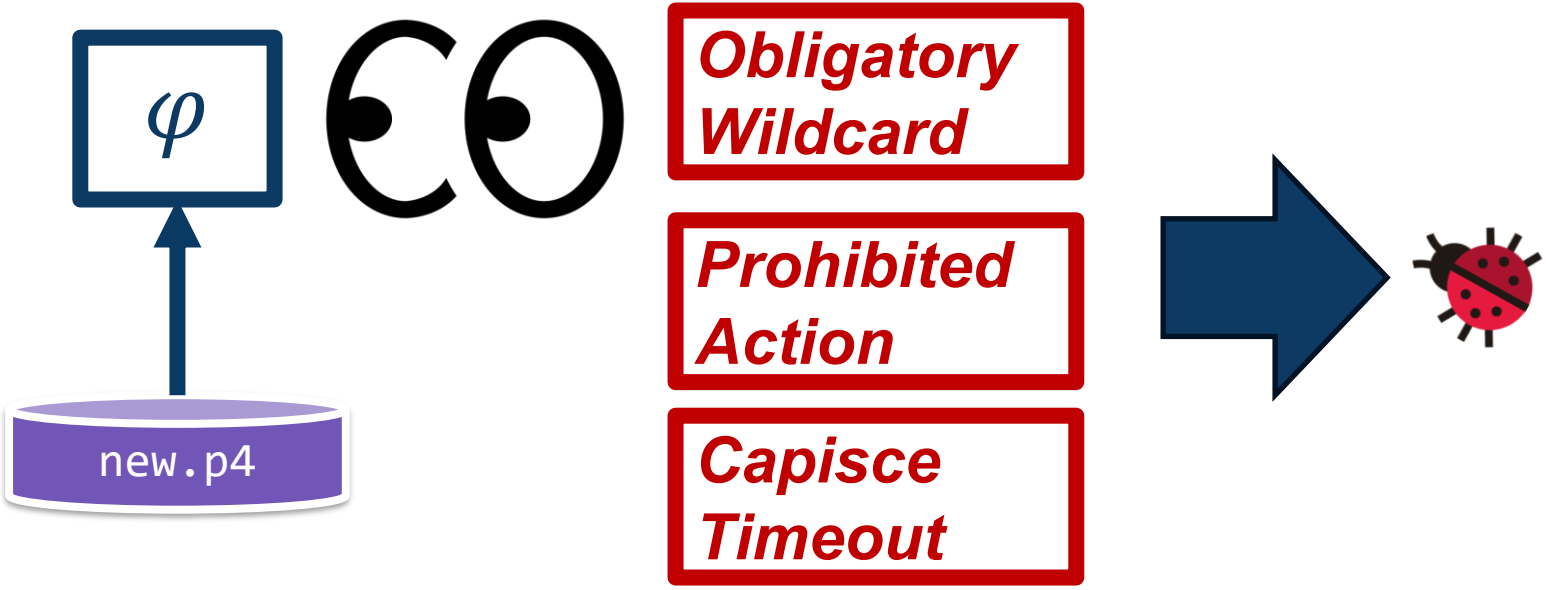


Evaluation on Real Programs

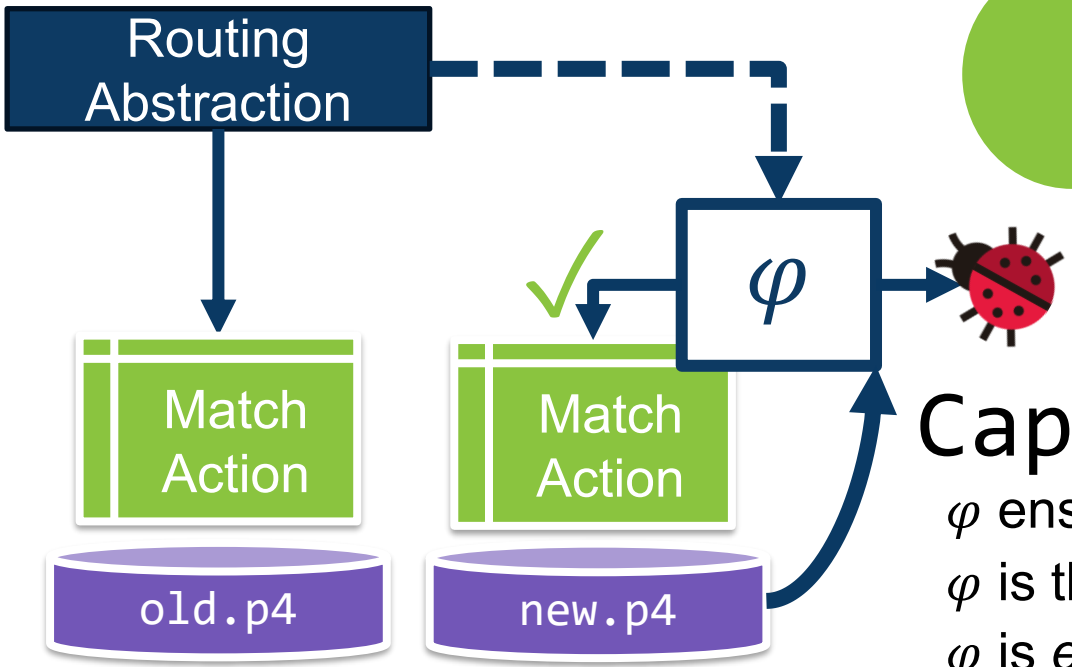
Program	Program Paths	Result	Time (s)	Explored Paths	Spec AST Size	Explored Ratio
			Absurd Programs			
ts-switching	21	$\bot$	0.160	2	1	0.095
mc-nat	39	$\bot$	0.089	1	1	0.026
			Fixes to Absurd Programs			
ts-switching-fixed	21	$\top$	0.030	0	1	0.0
mc-nat-fixed	39	$\top$	0.027	0	1	0.0
			Trivial Programs			
resubmit	9	$\top$	0.028	0	1	0.0
netpaxos-acceptor	0.116	$\top$	30.0	0	1	0.0
ecmp	102	$\top$	0.030	0	1	0.0
hula	3629	$\top$	0.068	0	1	0.0
ndp-router	3843	$\top$	2.9	0	1	0.0
			Nontrivial Programs			
arp	95	φ	5.0	0.016	349	0.17
heavy-hitter-2	267	φ	0.29	3	26	0.011
heavy-hitter-1	327	φ	0.60	7	90	0.021
flowlet	649	φ	1.8	9	127	0.014
simple_nat	66531	φ	5.2	54	1421	0.00081
07-multiprotocol	54459	φ	16	143	3138	0.0026
netchain	26726780	φ	2.9×10^3	264	11658	9.9×10^{-6}
linearroad	54477696		timeout			
fabric	133365047559893		timeout			
			Spec Smell Program Fixes			
heavy-hitter-1-fixed	327	φ	0.63	7	107	0.021
linearroad-fixed	54477696	φ	5.9×10^4	3236	179885	5.9×10^{-5}
fabric-fixed	133365047559893	φ	1.2×10^3	653	41140	4.9×10^{-12}

Program	Program Paths	Result	Time (s)	Explored Paths	ci-spec Size	Explored Ratio
			Absurd Programs			
ecmp	102	$\bot$	0.320	4	1	0.039
fabric	133365047559893	$\bot$	7.3	5	1	3.7×10^{-14}
netchain	26726780	$\bot$	27	7	1	2.6×10^{-7}
			Trivial Programs			
arp	95	$\top$	0.027	0	1	0.0
linearroad	54477696	$\top$	0.054	0	1	0.0
simple-nat	5548	$\top$	0.034	0	1	0.0
			Nontrivial Programs			
resubmit	9	φ	0.016	2	17	0.22
ts-switching	21	φ	0.10	1	4	0.048
mc-nat	39	φ	0.27	3	21	0.077
netpaxos-acceptor	116	φ	0.12	1	4	0.0086
heavy-hitter-2	267	φ	88	15	233	0.056
heavy-hitter-1	327	φ	0.10	11	187	0.034
flowlet	649	φ	79	15	490	0.023
hula	3629	φ	0.39	1	9	0.00028
ndp-router	3843	φ	40	36	824	0.0094
07-multiprotocol	54459	φ	30	232	5034	0.0043
			Spec Smells & Fixes			
ecmp-fixed	102	φ	0.28	3	34	0.029
mc-nat-fixed	27	$\top$	0.029	0	1	0.0

Manual Audit of Control Specifications



What can we know *in advance*?

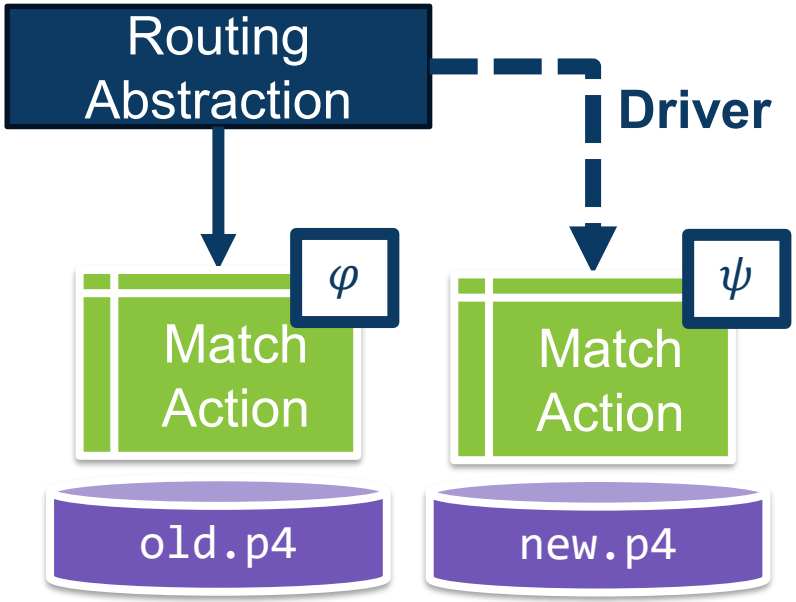


Safety

Capisce

- φ ensures no runtime bugs (safe)
- φ is the *most permissive safe spec*
- φ is *efficient to compute and check*

What can we know *in advance*?



Compatibility

Spectacle:

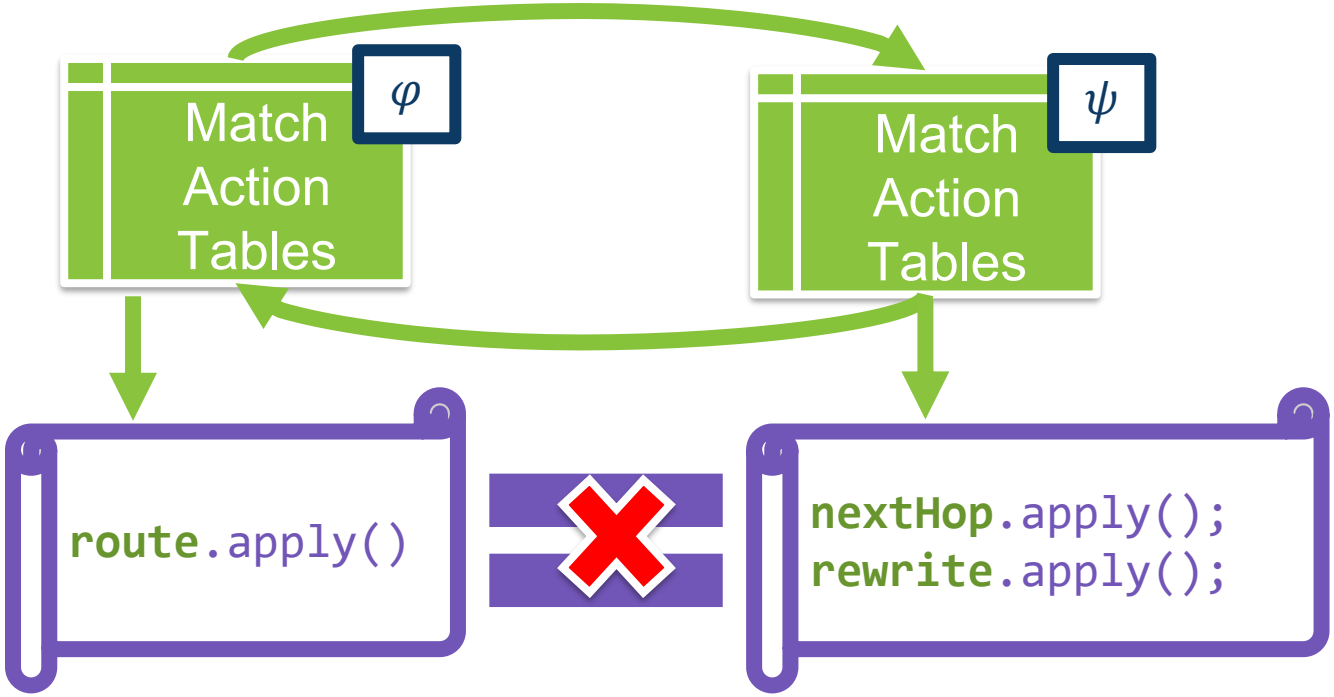
Can new.p4 modulo ψ realize *all required behaviors*?

Compatibility


```
route.apply();
```

```
nexthop.apply();
rewrite.apply();
```

Spectacle: Proving Compatibility



ipv4.dst	action
10.0.1.10	route(99, ca:f3)
*	drop()

ipv4.dst	action
~~10.0.1.10~~	next(99)
*	drop()

ipv4.dst	action
~~10.0.1.10~~	write(ca:f3)
*	drop()

```
route.apply()
```

=

```
nextHop.apply();
rewrite.apply();
```

ipv4.dst	action
10.0.1.10	route(99,ca:f3)
*	drop()

ipv4.dst	action
10.0.1.10	next(99)
*	drop

ipv4.dst	action
10.0.1.10	write(ca:f3)
*	drop

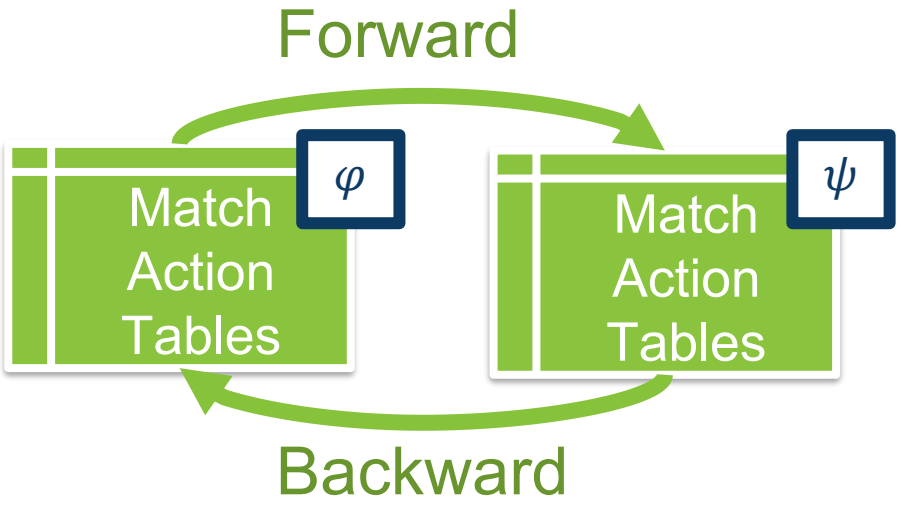
```
route.apply()
```

=

```
nextHop.apply();
rewrite.apply();
```

THE LINUX FOUNDATION

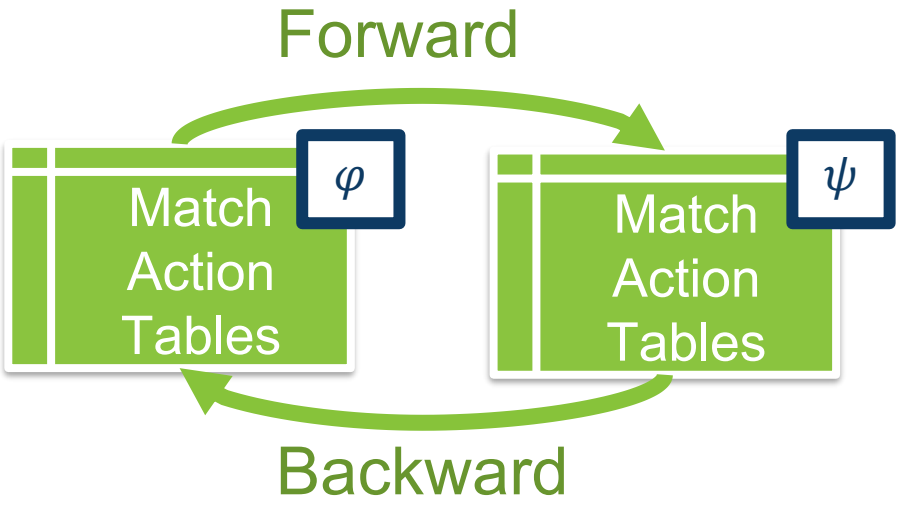
Well-Formedness Constraints




```
t = forward(s);
s' = backward(t);
assert s' = s

s = backward(t);
t' = forward(s);
assert t' = t
```

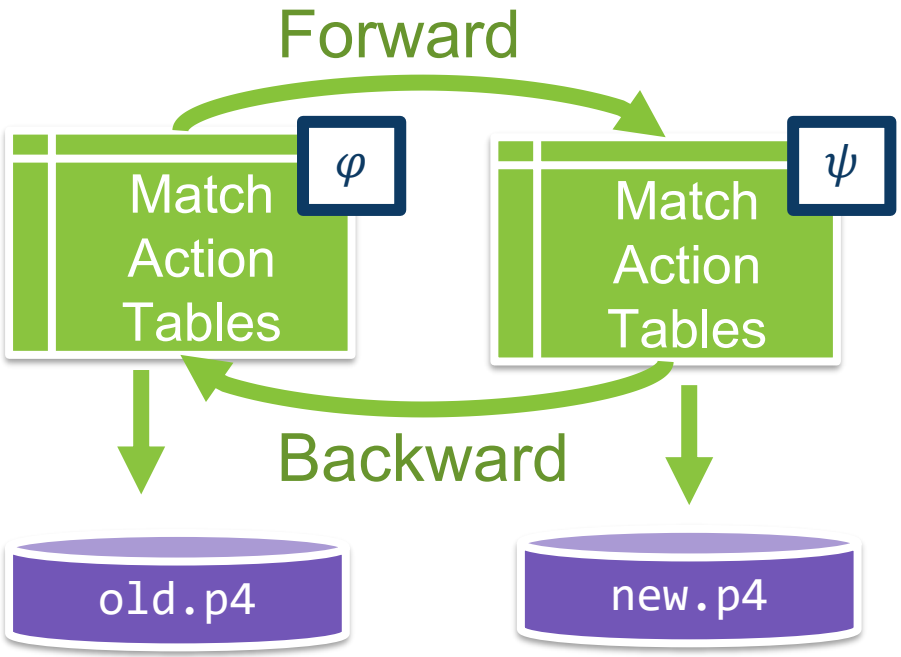
Well-Formedness Constraints (with state)




```
(t, y) = forward(s, x);
(s', z) = backward(t, y);
assert s' = s && y = z

(s, y) = backward(t, x);
(t', z) = forward(s, y);
assert s' = s && y = z
```

Symmetric Lenses [Hoffman, Pierce, Wagner. POPL'11]



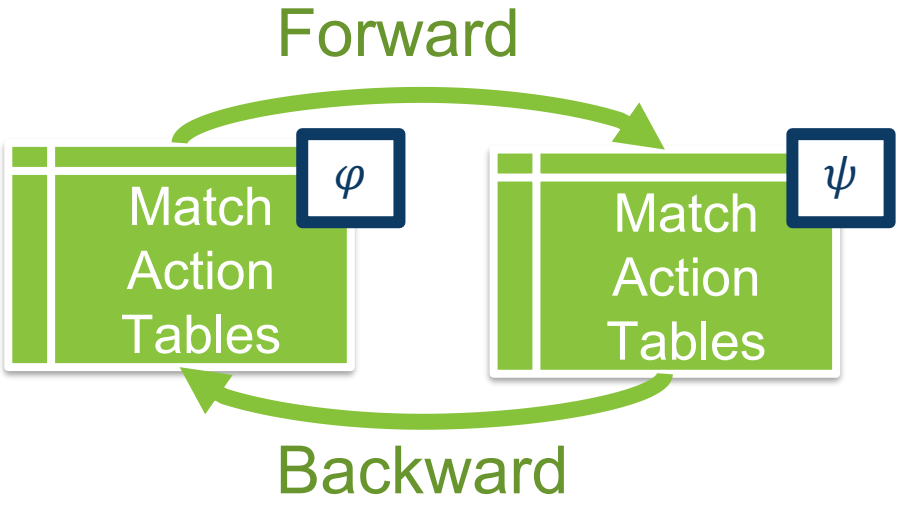
Lots to verify…..

$$\forall s, t.\ s \vDash \varphi \wedge \boldsymbol{forward}(s) = t \wedge t \vDash \psi \Rightarrow \forall\, pkt.\, \boldsymbol{old}(s, pkt) = \boldsymbol{new}(t, pkt)$$

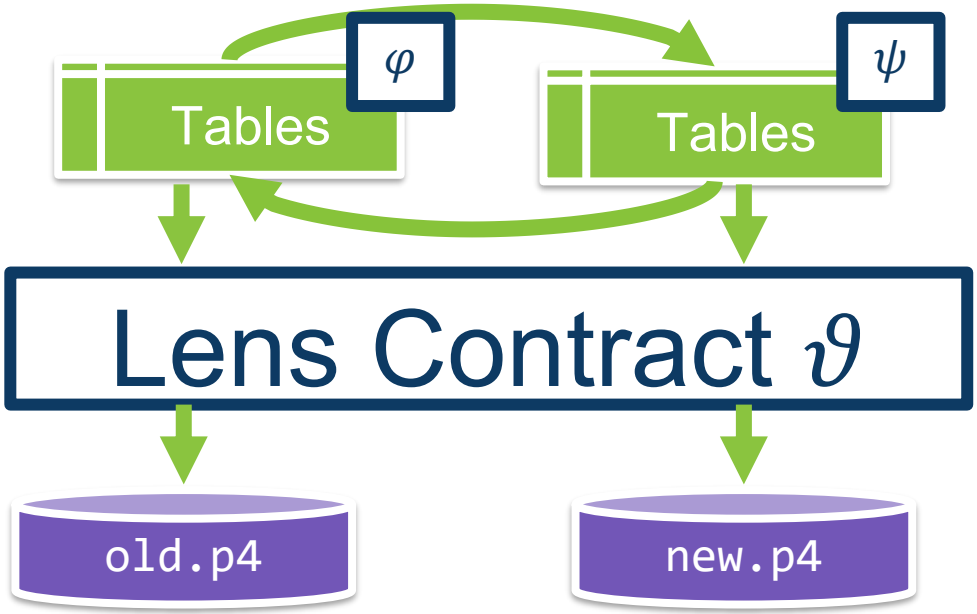
$$\forall t, s.\ t \vDash \psi \wedge \boldsymbol{backward}(t) = s \wedge s \vDash \varphi \Rightarrow \forall\, pkt.\, \boldsymbol{old}(s, pkt) = \boldsymbol{new}(t, pkt)$$

A New (and really old) Idea: Lens Contracts

Constraint Maintainers [Meertens '98]



$$\vartheta$$

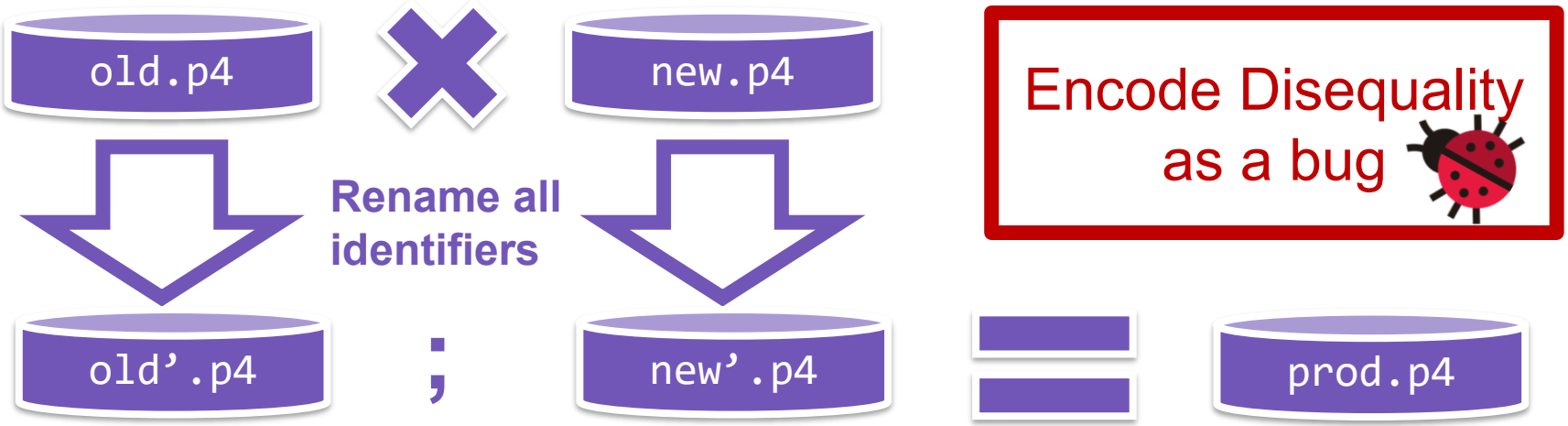


$$forward(s) = t \Rightarrow (s,t) \vDash \vartheta$$
$$backward(t) = s \Rightarrow (s,t) \vDash \vartheta$$

$$(s,t) \vDash \vartheta \Rightarrow$$
$$old(s,pkt) = new(t,pkt)$$

Verifying Equivalence via Product Programs

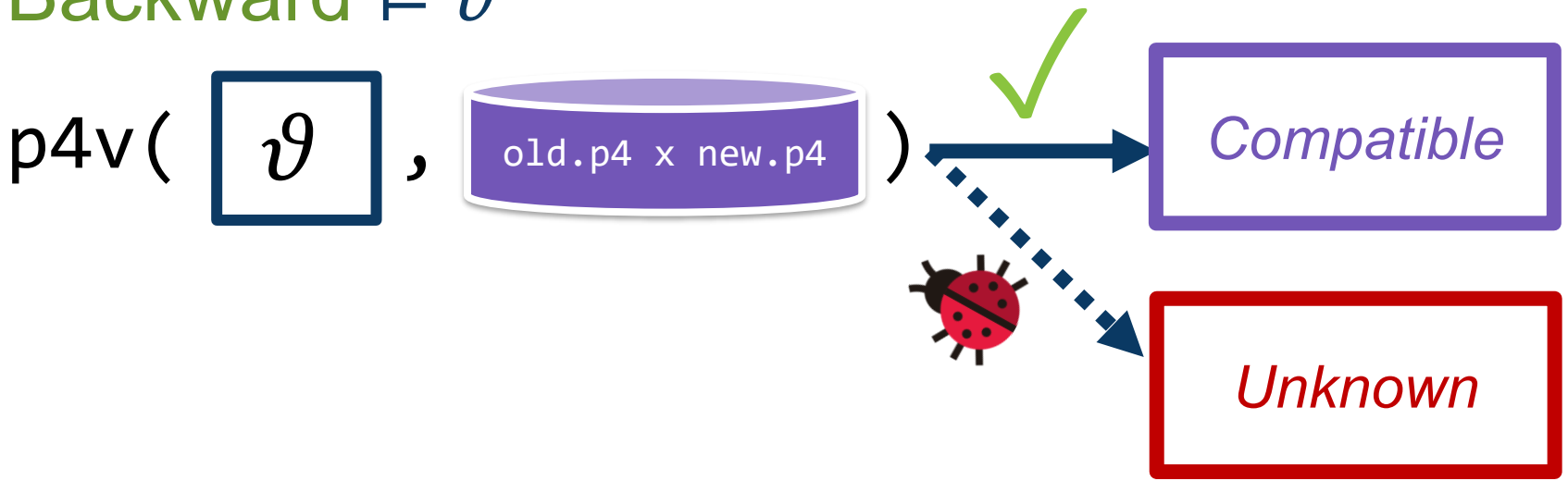
$$(s,t) \vDash \vartheta \Rightarrow \boldsymbol{old}(s,pkt) = \boldsymbol{new}(t,pkt)$$



A Lens Contract is a (Product) Control Spec

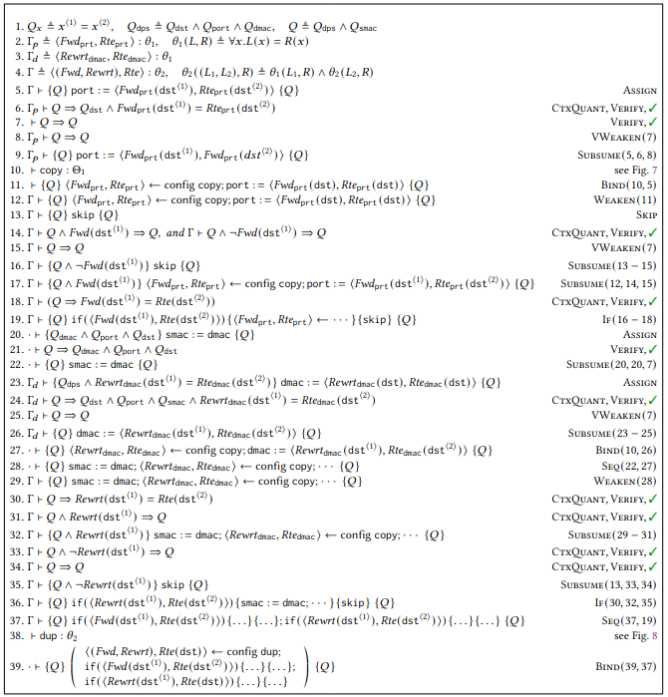
Forward $\vDash \vartheta$

Backward $\vDash \vartheta$



Proofs in Spectacle:

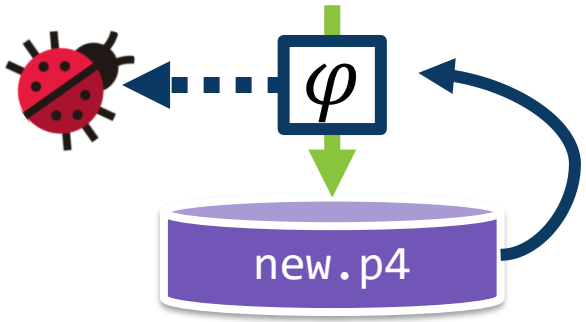
```
assume_ eq@@lens (copy Nxt Rte) @@
annotate eq@@ifthen2 (Nxt dst, Rte dst)(
 lens (compose Nxt_port Grp Rte_prt) @@
 assume_ eq @@
 assign2 port (Nxt_port dst, Rte_prt dst) @@
 assign1 smac dmac @@
 lens (copy Nxt_dmac Rte_dmac) @@
  assume_ eq @@
  assign2 dmac (Nxt_dmac dst, Rte_dmac dst)
  skip) @@ skip
assert_ eq
```



1. $Q_x \triangleq x^{(1)} = x^{(2)}, \quad Q_{dps} \triangleq Q_{dst} \wedge Q_{port} \wedge Q_{dmac}, \quad Q \triangleq Q_{dps} \wedge Q_{snac}$
2. $\Gamma_p \triangleq \langle Fwd_{prt}, Rte_{prt} \rangle : \theta_1, \quad \theta_1(L,R) \triangleq \forall x.L(x) = R(x)$
3. $\Gamma_d \triangleq \langle Rewrt_{dmac}, Rte_{dmac} \rangle : \theta_1$
4. $\Gamma \triangleq \langle (Fwd, Rewrt), Rte \rangle : \theta_2, \quad \theta_2((L_1,L_2),R) \triangleq \theta_1(L_1,R) \wedge \theta_2(L_2,R)$
5. $\Gamma \vdash \{Q\}\ \mathsf{port} := \langle Fwd_{prt}(\mathsf{dst}^{(1)}), Rte_{prt}(\mathsf{dst}^{(2)}) \rangle\ \{Q\}$ — Assign
6. $\Gamma_p \vdash Q \Rightarrow Q_{dst} \wedge Fwd_{prt}(\mathsf{dst}^{(1)}) = Rte_{prt}(\mathsf{dst}^{(2)})$ — CtxQuant, Verify, ✓
7. $\vdash Q \Rightarrow Q$ — Verify, ✓
8. $\Gamma_p \vdash Q \Rightarrow Q$ — VWeaken(7)
9. $\Gamma_p \vdash \{Q\}\ \mathsf{port} := \langle Fwd_{prt}(\mathsf{dst}^{(1)}), Fwd_{prt}(dst^{(2)}) \rangle\ \{Q\}$ — Subsume(5, 6, 8)
10. $\vdash \mathsf{copy} : \Theta_1$ — see Fig. 7
11. $\vdash \{Q\}\ \langle Fwd_{prt}, Rte_{prt} \rangle \leftarrow \mathsf{config\ copy}; \mathsf{port} := \langle Fwd_{prt}(\mathsf{dst}), Rte_{prt}(\mathsf{dst}) \rangle\ \{Q\}$ — Bind(10, 5)
12. $\Gamma \vdash \{Q\}\ \langle Fwd_{prt}, Rte_{prt} \rangle \leftarrow \mathsf{config\ copy}; \mathsf{port} := \langle Fwd_{prt}(\mathsf{dst}), Rte_{prt}(\mathsf{dst}) \rangle\ \{Q\}$ — Weaken(11)
13. $\Gamma \vdash \{Q\}\ \mathsf{skip}\ \{Q\}$ — Skip
14. $\Gamma \vdash Q \wedge Fwd(\mathsf{dst}^{(1)}) \Rightarrow Q$, *and* $\Gamma \vdash Q \wedge \neg Fwd(\mathsf{dst}^{(1)}) \Rightarrow Q$ — CtxQuant, Verify, ✓
15. $\Gamma \vdash Q \Rightarrow Q$ — VWeaken(7)
16. $\Gamma \vdash \{Q \wedge \neg Fwd(\mathsf{dst}^{(1)})\}\ \mathsf{skip}\ \{Q\}$ — Subsume(13 – 15)
17. $\Gamma \vdash \{Q \wedge Fwd(\mathsf{dst}^{(1)})\}\ \langle Fwd_{prt}, Rte_{prt} \rangle \leftarrow \mathsf{config\ copy}; \mathsf{port} := \langle Fwd_{prt}(\mathsf{dst}^{(1)}), Rte_{prt}(\mathsf{dst}^{(2)}) \rangle\ \{Q\}$ — Subsume(12, 14, 15)
18. $\Gamma \vdash (Q \Rightarrow Fwd(\mathsf{dst}^{(1)}) = Rte(\mathsf{dst}^{(2)}))$ — CtxQuant, Verify, ✓
19. $\Gamma \vdash \{Q\}\ \mathsf{if}(\langle Fwd(\mathsf{dst}^{(1)}), Rte(\mathsf{dst}^{(2)}) \rangle)\{\langle Fwd_{prt}, Rte_{prt} \rangle \leftarrow \cdots\}\{\mathsf{skip}\}\ \{Q\}$ — If(16 – 18)
20. $\cdot \vdash \{Q_{dmac} \wedge Q_{port} \wedge Q_{dst}\}\ \mathsf{smac} := \mathsf{dmac}\ \{Q\}$ — Assign
21. $\cdot \vdash Q \Rightarrow Q_{dmac} \wedge Q_{port} \wedge Q_{dst}$ — Verify, ✓
22. $\cdot \vdash \{Q\}\ \mathsf{smac} := \mathsf{dmac}\ \{Q\}$ — Subsume(20, 20, 7)
23. $\Gamma_d \vdash \{Q_{dps} \wedge Rewrt_{dmac}(\mathsf{dst}^{(1)}) = Rte_{dmac}(\mathsf{dst}^{(2)})\}\ \mathsf{dmac} := \langle Rewrt_{dmac}(\mathsf{dst}), Rte_{dmac}(\mathsf{dst}) \rangle\ \{Q\}$ — Assign
24. $\Gamma_d \vdash Q \Rightarrow Q_{dst} \wedge Q_{port} \wedge Q_{smac} \wedge Rewrt_{dmac}(\mathsf{dst}^{(1)}) = Rte_{dmac}(\mathsf{dst}^{(2)})$ — CtxQuant, Verify, ✓
25. $\Gamma_d \vdash Q \Rightarrow Q$ — VWeaken(7)
26. $\Gamma_d \vdash \{Q\}\ \mathsf{dmac} := \langle Rewrt_{dmac}(\mathsf{dst}^{(1)}), Rte_{dmac}(\mathsf{dst}^{(2)}) \rangle\ \{Q\}$ — Subsume(23 – 25)
27. $\cdot \vdash \{Q\}\ \langle Rewrt_{dmac}, Rte_{dmac} \rangle \leftarrow \mathsf{config\ copy}; \mathsf{dmac} := \langle Rewrt_{dmac}(\mathsf{dst}^{(1)}), Rte_{dmac}(\mathsf{dst}^{(2)}) \rangle\ \{Q\}$ — Bind(10, 26)
28. $\cdot \vdash \{Q\}\ \mathsf{smac} := \mathsf{dmac}; \langle Rewrt_{dmac}, Rte_{dmac} \rangle \leftarrow \mathsf{config\ copy}; \cdots\ \{Q\}$ — Seq(22, 27)
29. $\Gamma \vdash \{Q\}\ \mathsf{smac} := \mathsf{dmac}; \langle Rewrt_{dmac}, Rte_{dmac} \rangle \leftarrow \mathsf{config\ copy}; \cdots\ \{Q\}$ — Weaken(28)
30. $\Gamma \vdash Q \Rightarrow Rewrt(\mathsf{dst}^{(1)}) = Rte(\mathsf{dst}^{(2)})$ — CtxQuant, Verify, ✓
31. $\Gamma \vdash Q \wedge Rewrt(\mathsf{dst}^{(1)}) \Rightarrow Q$ — CtxQuant, Verify, ✓
32. $\Gamma \vdash \{Q \wedge Rewrt(\mathsf{dst}^{(1)})\}\ \mathsf{smac} := \mathsf{dmac}; \langle Rewrt_{dmac}, Rte_{dmac} \rangle \leftarrow \mathsf{config\ copy}; \cdots\ \{Q\}$ — Subsume(29 – 31)
33. $\Gamma \vdash Q \wedge \neg Rewrt(\mathsf{dst}^{(1)}) \Rightarrow Q$ — CtxQuant, Verify, ✓
34. $\Gamma \vdash Q \Rightarrow Q$ — CtxQuant, Verify, ✓
35. $\Gamma \vdash \{Q \wedge \neg Rewrt(\mathsf{dst}^{(1)})\}\ \mathsf{skip}\ \{Q\}$ — Subsume(13, 33, 34)
36. $\Gamma \vdash \{Q\}\ \mathsf{if}(\langle Rewrt(\mathsf{dst}^{(1)}), Rte(\mathsf{dst}^{(2)}) \rangle)\{\mathsf{smac} := \mathsf{dmac}; \cdots\}\{\mathsf{skip}\}\ \{Q\}$ — If(30, 32, 35)
37. $\Gamma \vdash \{Q\}\ \mathsf{if}(\langle Fwd(\mathsf{dst}^{(1)}), Rte(\mathsf{dst}^{(2)}) \rangle)\{\ldots\}\{\ldots\}; \mathsf{if}(\langle Rewrt(\mathsf{dst}^{(1)}), Rte(\mathsf{dst}^{(2)}) \rangle)\{\ldots\}\{\ldots\}\ \{Q\}$ — Seq(37, 19)
38. $\vdash \mathsf{dup} : \theta_2$ — see Fig. 8
39. $\cdot \vdash \{Q\}\ \left(\begin{array}{l} \langle (Fwd, Rewrt), Rte(\mathsf{dst}) \rangle \leftarrow \mathsf{config\ dup}; \\ \mathsf{if}(\langle Fwd(\mathsf{dst}^{(1)}), Rte(\mathsf{dst}^{(2)}) \rangle)\{\ldots\}\{\ldots\}; \\ \mathsf{if}(\langle Rewrt(\mathsf{dst}^{(1)}), Rte(\mathsf{dst}) \rangle)\{\ldots\}\{\ldots\} \end{array} \right)\ \{Q\}$ — Bind(39, 37)

Control Specifications

Capisce



Spectacle