# CAMPION: DEBUGGING ROUTER CONFIGURATION DIFFERENCES

Alan Tang[1]

Siva Kesava Reddy Kakarla[1] Ryan Beckett[2] Ennan Zhai[3]

Matt Brown[4] Todd Millstein[1,4] Yuval Tamir[1] George Varghese[1]

[1]University of California, Los Angeles [2]Microsoft Research

[3]Alibaba Group [4]Intentionet

# Network Misconfigurations Cause Outages

**Google accidentally broke the internet throughout Japan**

A mistake led to internet outages for about half of the country.

**What was wrong with United's router?**

Airline isn't saying, but ideas abound

**BGP errors are to blame for Monday's Twitter outage, not DDoS attacks**

No, your toaster didn't kill Twitter, an engineer did

**Xbox Live outage caused by network configuration problem**

BY TODD BISHOP on April 15, 2013 at 9:27 am

# Router Configuration is Hard

## Multiple Protocols

- BGP, OSPF, etc.

## Lots of configurable properties

- Link costs, IP addresses, etc.

## Various filters for header fields

- ACLs, prefix lists, community lists, etc.

```
interface GigabitEthernet2/0
 ip address 2.12.12.1 255.255.255.0
!
router ospf 1
 router-id 2.1.1.1
 redistribute connected subnets
 network 2.0.0.0 0.255.255.255 area 1
!
router bgp 2
 bgp router-id 2.1.1.1
 neighbor as1 peer-group
 neighbor as1 remote-as 1
 neighbor 10.12.11.1 peer-group as1
 !
 address-family ipv4
  network 1.0.0.0 mask 0.255.255.255
  neighbor as1 send-community
  neighbor as1 route-map POL in
  neighbor 10.12.11.1 activate
 exit-address-family
```

# Network Configuration is Hard

**Multiple routers**

- border, core, etc.

**Different vendor formats**

- Cisco, Juniper, etc.

**Need to be updated**

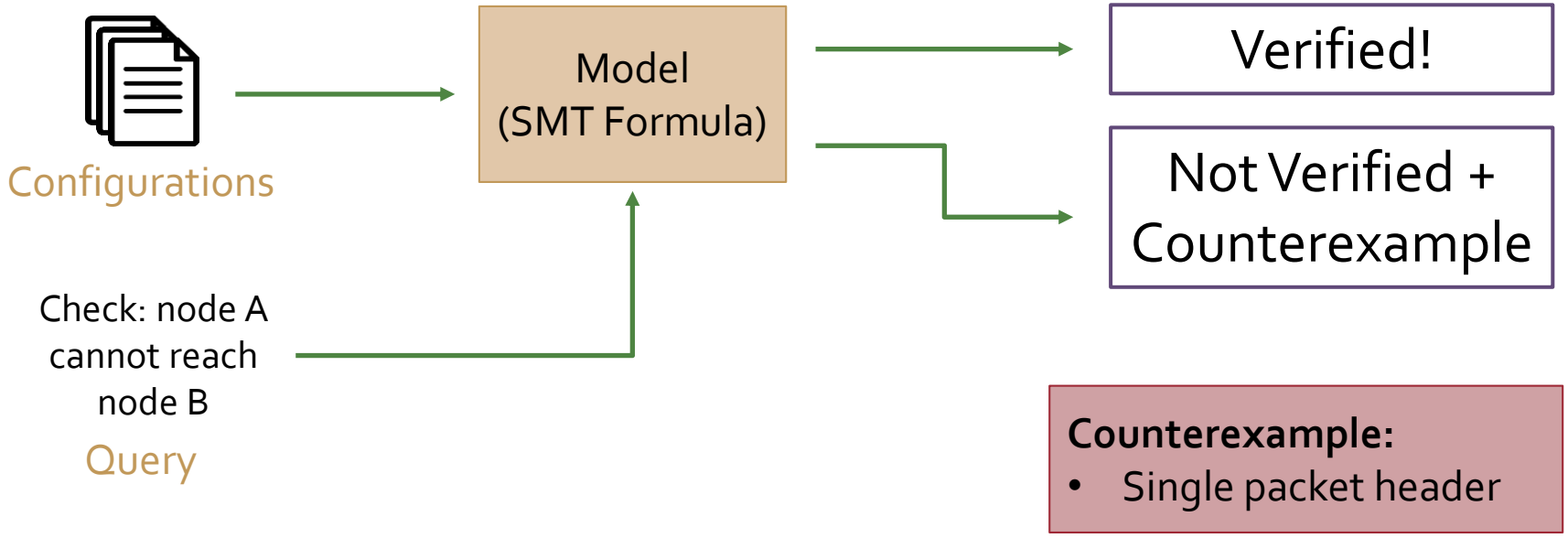
```
interface GigabitEthernet2/0
 ip address 2.12.12.1 255.255.255.0
!
router ospf 1
 router-id 2.1.1.1
 redistribute connected subnets
 network 2.0.0.0 0.255.255.255 area 1
!
router bgp 2
 bgp router-id 2.1.1.1
 neighbor as1 peer-group
 neighbor as1 remote-as 1
 neighbor 10.12.11.1 peer-group as1
 !
 address-family ipv4
  network 1.0.0.0 mask 0.255.255.255
  neighbor as1 send-community
  neighbor as1 route-map POL in
  neighbor 10.12.11.1 activate
 exit-address-family
```

```
protocols {
    bgp {
        traceoptions {
            file nsr_bgp_trace_file;
            flag nsr-synchronization detail;
        }
        log-updown;
        damping;
        group GROUP1 {
            type internal;
            local-address 2.1.1.1;
            family inet {
                unicast;
            }
            import POL;
            local-as 2;
            neighbor 10.12.11.1 {
                description R1;
            }
        }
    }
    ospf {
        area 0.0.0.1 {
            interface ge-1/0/0;
            interface ge-2/0/0;
        }
    }
}
```

# Existing Tools Provide Single Counterexample

Existing tools for control plane can model and verify network behavior

- e.g. Minesweeper [Beckett et al. 2017]



# Problem

**What a single counterexample cannot tell you:**

| What is causing the error? | What is the scope of the error? |
| --- | --- |
| • BGP?<br>• OSPF?<br>• Static routes?<br>• ACL?<br>• Bug in the modeling? | • One error or many errors?<br>• Few IP addresses or many?<br>• Real error or exceptional case? |

# Our Goal: Error Localization

1. Find all errors

2. Find the part of the configuration causing the errors

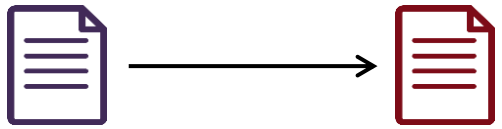
3. Find the input sets affected

# Our Setting: Equivalence of Router Pair

Task: Check that a pair of routers are equivalent / Find their differences

Upgrading / updating configuration

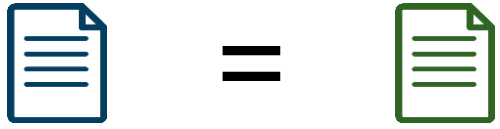
- Rewrite policy for different router
- Small update without major change



Old New

Backups

- Intended to have same behavior
- May contain minor differences



R1 R2

# Campion

Finds the difference between two router configurations using modular comparisons

- **Finds all differences**
- **Text localization:** configuration block or lines causing a difference
- **Header localization:** input headers are affected by the difference

Found bugs in major cloud datacenter and large university network configurations:

- Errors in datacenter routers could have caused service disruption
- Found unintentional policy in university core and border routers
  - Operators said was “highly unlikely” to find by manual inspection

# Example

## Cisco Excerpt

```
ip prefix-list NETS permit 10.9.0.0/16 le 32
ip prefix-list NETS permit 10.100.0.0/16 le 32
!
ip community-list standard COMM permit 10:10
ip community-list standard COMM permit 10:11
!
route-map POL deny 10
 match ip address NETS
route-map POL deny 20
 match community COMM
route-map POL permit 30
 set local-preference 30
```

Prefixes

Communities

Routing Policy

## Juniper Excerpt

```
prefix-list NETS {
    10.9.0.0/16;
    10.100.0.0/16;
}
community COMM members [10:10 10:11];
policy-statement POL {
    term rule1 {
        from prefix-list NETS;
        then reject;
    }
    term rule2 {
        from community COMM;
        then reject;
    }
    term rule3 {
        then {
            local-preference 30;
            accept;
        }
    }
}
```

# Output Comparison

Minesweeper:
(after getting relevant model variables)

| | |
|---|---|
| Route received (Cisco) | Prefix: 10.9.0.0/17 |
| Route received (Juniper) | Prefix: 10.9.0.0/17 |
| Packet | dstIp: 10.9.0.0 |
| Forwarding | Juniper router forwards (BGP)<br>Cisco router does not forward |

Campion:

| | cisco_router | juniper_router |
|---|---|---|
| Included Prefixes | 10.9.0.0/16 : 16-32<br>10.100.0.0/16 : 16-32 | |
| Excluded Prefixes | 10.9.0.0/16 : 16-16<br>10.100.0.0/16 : 16-16 | |
| Policy Name | POL | POL |
| Action | REJECT | SET LOCAL PREF 30<br>ACCEPT |
| Text | route-map POL deny 10<br>match ip address NETS | rule3 {<br>then {<br>local-preference 30;<br>accept;<br>}<br>} |

| | cisco_router | juniper_router |
|---|---|---|
| Included Prefixes | 0.0.0.0/0 : 0-32 | |
| Excluded Prefixes | 10.9.0.0/16 : 16-32<br>10.100.0.0/16 : 16-32 | |
| Community | 10:10 | |
| Policy Name | POL | POL |
| Action | REJECT | SET LOCAL PREF 30<br>ACCEPT |
| Text | route-map POL deny 20<br>match community COMM | rule3 {<br>then {<br>local-preference 30;<br>accept;<br>}<br>} |

# Example Difference

## Cisco Excerpt

```
ip prefix-list NETS permit 10.9.0.0/16 le 32
ip prefix-list NETS permit 10.100.0.0/16 le 32
!
ip community-list standard COMM permit 10:10
ip community-list standard COMM permit 10:11
!
route-map POL deny 10
 match ip address NETS
route-map POL deny 20
 match community COMM
route-map POL permit 30
 set local-preference 30
```

## Juniper Excerpt

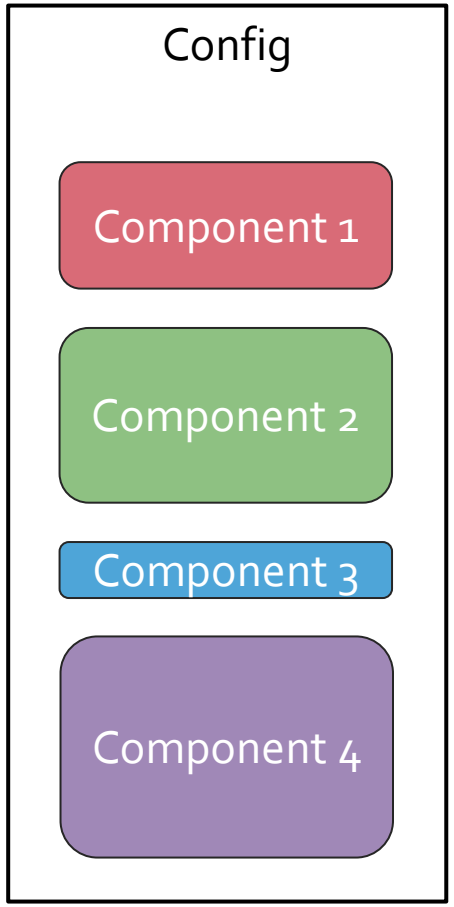
```
prefix-list NETS {
    10.9.0.0/16;
    10.100.0.0/16;
}
community COMM members [10:10 10:11];
policy-statement POL {
    term rule1 {
        from prefix-list NETS;
        then reject;
    }
    term rule2 {
        from community COMM;
        then reject;
    }
    term rule3 {
        then {
            local-preference 30;
            accept;
```

**Difference 1:** match /16 or longer vs. /16 exact
**Difference 2:** match EITHER community vs. BOTH communities

# Key Idea: Modularity

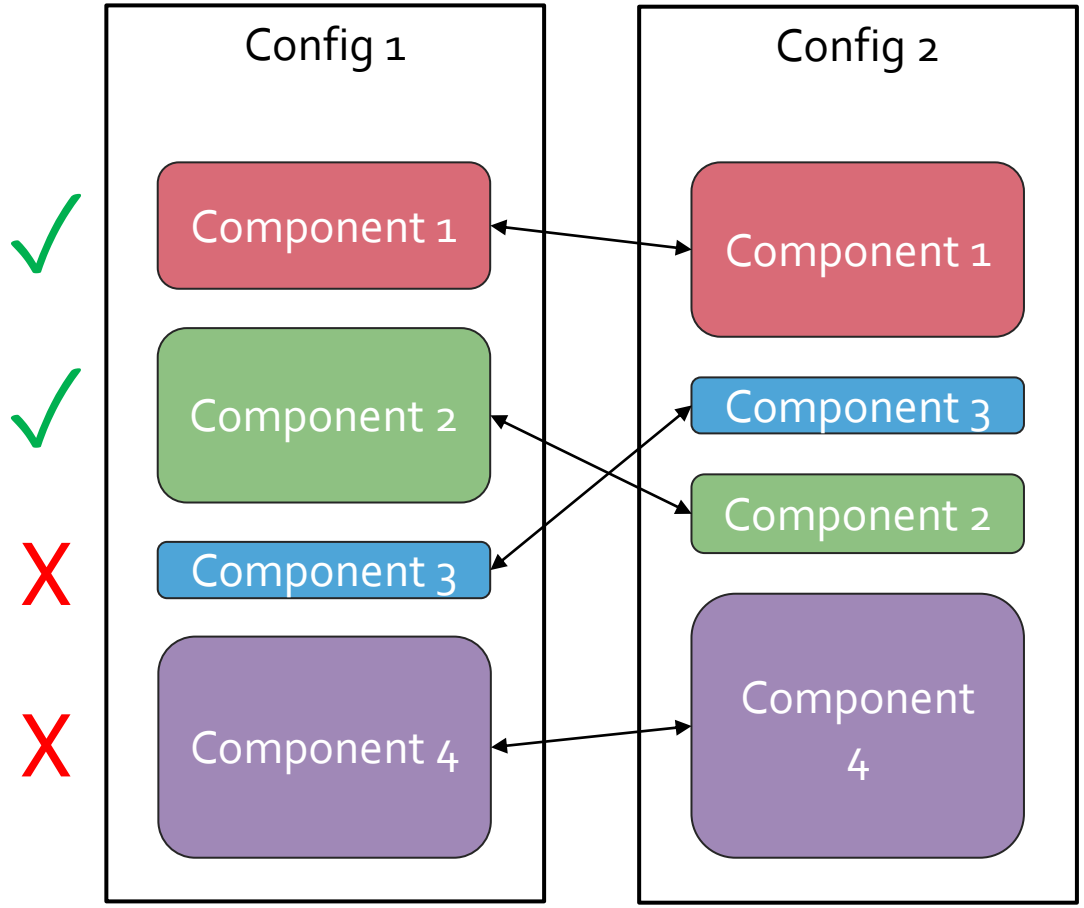
Perform checks on individual components to take advantage of config structure

1. Immediately localizes to component
2. Does not require modeling protocols
3. Simplifies checks for individual components
4. Allows tracing back to lines and getting multiple results



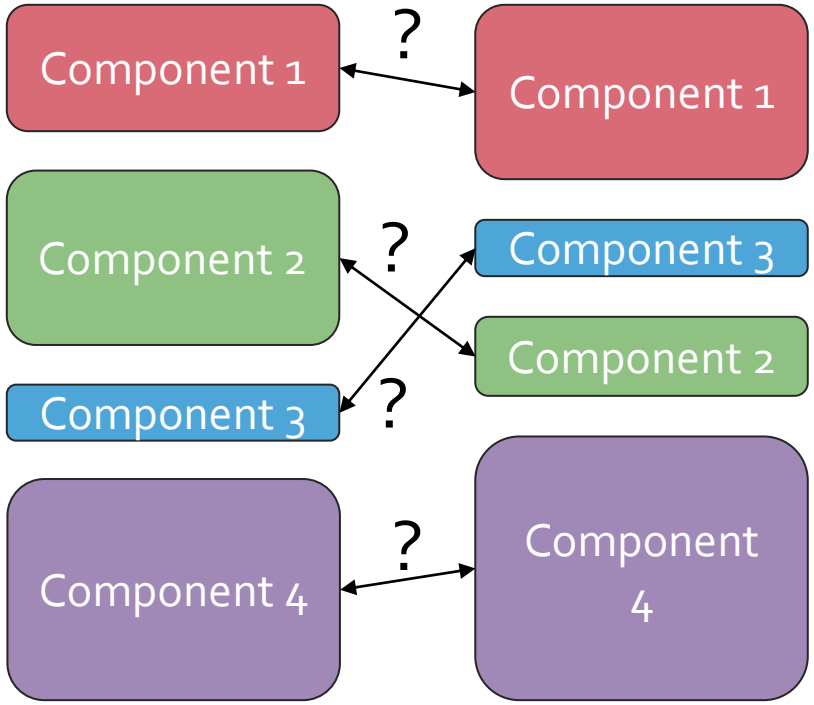
# Overview

1. Match corresponding components:
   - Use heuristics for matching edges
2. Compare corresponding components
   - Use structure of each component
3. Provide the text and headers (where applicable)
   - Use info from configuration

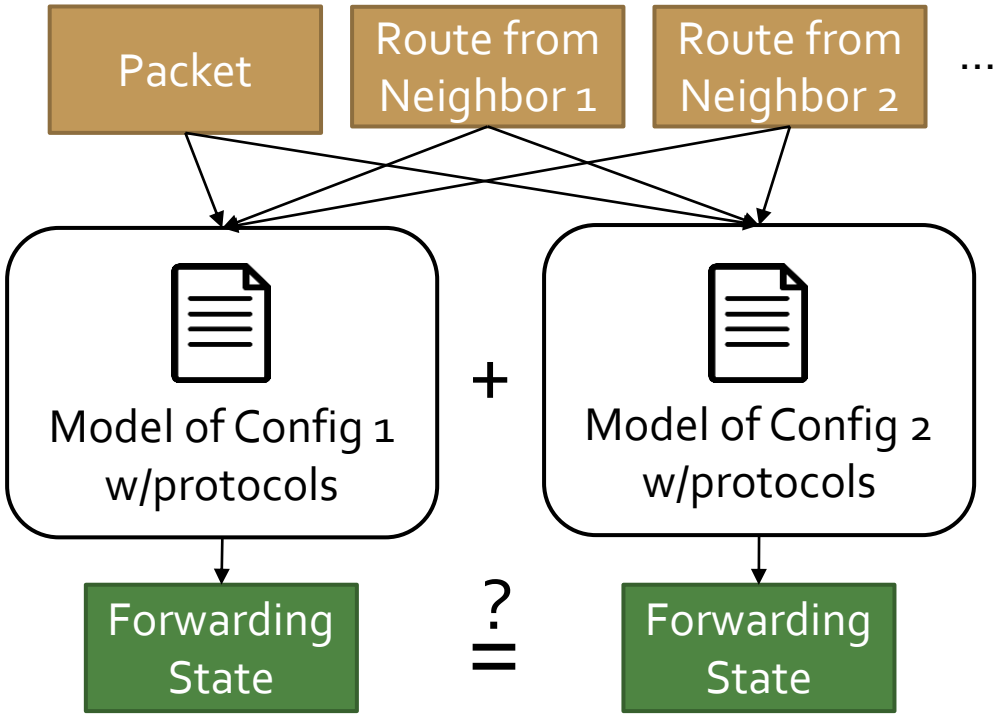


# Modular vs. Monolithic

## Modular (Campion)




## Monolithic (Minesweeper)

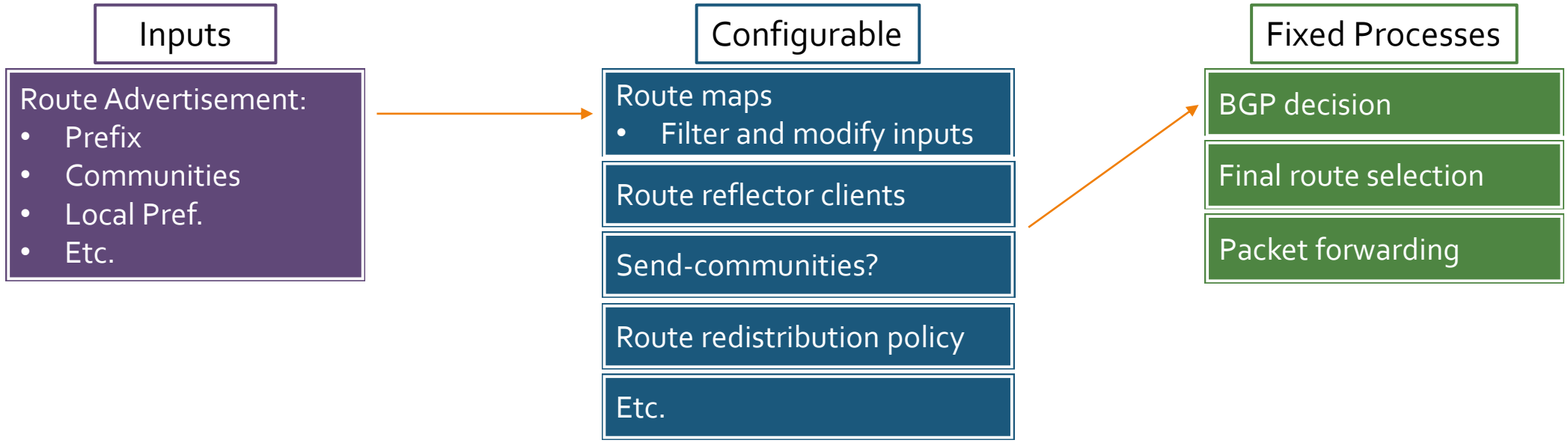


# Avoids Modeling Protocols

**Does NOT need to model or simulate protocols!**

- Protocols are fixed
- Only needs to check configurable properties for all inputs

**For BGP:**



# Simplifying Checks

## Behavioral / Semantic Comparison

ACLs and route maps configure function on packet headers

→ Need to model behavior

```
route-map POL deny 10
 match ip address NETS
route-map POL deny 20
 match community COMM
route-map POL permit 30
 set local-preference 30
```

```
policy-statement POL {
    term rule1 {
        from prefix-list NETS;
        then reject;
    }
    term rule2 {
        from community COMM;
        then reject;
    }
    term rule3 {
        then {
            local-preference 30;
            accept;
        }
    }
}
```

## Structural Comparison

Most other component behaviors can only be expressed in a one way

→ Can be compared structurally

```
interface ethernet ½
  ip ospf cost 65
```

```
interface fe-1/0/1.0 {
    metric 65;
}
```

# Comparing Filter Behavior

1. Associate clauses with input set, action, and text

```
route-map POL deny 10
 match ip address NETS
route-map POL deny 20
 match community COMM
route-map POL permit 30
 set local-preference 30
```

| Inputs: | [NETS] |
|---|---|
| Action: | Reject |
| Text: | route-map POL deny 10<br>match ip address NETS |

| Inputs: | ¬[NETS] ∩ [COMM] |
|---|---|
| Action: | Reject |
| Text: | route-map POL deny 20<br>match community COMM |

| Inputs: | ¬[NETS] ∩ ¬[COMM] |
|---|---|
| Action: | Permit, local-pref = 30 |
| Text: | route-map POL permit 30<br>set local-preference 30 |

2. Compare clauses with different actions

| Inputs: | [NETS2] |
|---|---|
| Action: | Reject |
| Text: | route-map POL2 deny 10<br>match ip address NETS2 |

| Inputs: | ¬[NETS2] |
|---|---|
| Action: | Permit, local-pref = 30 |
| Text: | route-map POL2 permit 30<br>set local-preference 30 |

3. Return cases where intersections are non-empty

**R1 rejects but R2 permits**

| Routes: | ¬[NETS] ∩ [COMM] ∩ ¬[NETS2] |
|---|---|
| Text 1: | route-map POL deny 20<br>match community COMM |
| Text 2: | route-map POL2 permit 30<br>set local-preference 30 |

# Data Center Results

- Ran Campion on configurations from a major cloud datacenter network
- Tried on three scenarios
  - Scenario 1: Backup routers
  - Scenario 2: Router replacement to different vendor
  - Scenario 3: Gateway ACLs
- They ran Campion and interpreted results without feedback from us
- Some differences could have caused major issues if left undetected
- No false positives

| Scenario | Component | Structural or Semantic | Differences |
|---|---|---|---|
| Scenario 1 | BGP | Semantic | 5 |
| | Static Routes | Structural | 2 |
| Scenario 2 | BGP | Semantic | 4 |
| Scenario 3 | ACLs | Semantic | 3 |

**Table 6: Data Center Network Results**

# University Results

- Core router and border router backup pairs
  - Different vendors
- We ran and interpreted results without knowing detailed intent
  - Confirmed errors with operators
- Found differences in multiple policies
  - Some present for nearly three years
  - Operators claimed they were unlikely to discover by manually inspecting configs
- Some false positives
  - Intentional static route differences
  - One difference that should not affect behavior

| Router Pair | Route Map | Outputted Differences | Differences Reported | Confirmed | Pending |
|---|---|---|---|---|---|
| Core Routers | Export 1 | 5 | 5 | 4 | 1 |
| | Export 2 | 1 | 1 | 1 | 0 |
| Border Routers | Export 3 | 1 | 1 | 1 | 0 |
| | Export 4 | 1 | 1 | 1 | 0 |
| | Export 5 | 2 | 1 | 1 | 0 |
| | Import | 0 | - | - | - |

**(a) SemanticDiff results on route maps**

| Router Pair | Component | Classes of Errors | Differences Reported | Confirmed | Pending |
|---|---|---|---|---|---|
| Core Routers | Static Routes | 2 | 1 | 0 | 0 |
| | BGP Properties | 1 | 1 | 0 | 0 |

**(b) StructuralDiff results**

# Conclusion

- Campion can find and localize differences between two configurations
  - Gets components and lines
  - Gets input space
- Use modularity:
  - Reflects structure of the configuration
  - Avoid modeling protocols
  - Simplifies checks
  - Fine-grained localization
- We found many differences in data center and university network configurations