Practical Verification Techniques for Wide-Area Routing

Nick Feamster
M.I.T. Computer Science and Artificial Intelligence Laboratory
feamster@lcs.mit.edu

http://nms.lcs.mit.edu/bgp/

(Thanks to Hari Balakrishnan and Jennifer Rexford)

BGP is Flexible

- Many options for implementing a variety of policies
  - Route injection, redistribution, aggregation
  - Import and export policies
  - Access control lists, filtering
  - AS Path prepending
  - Communities
- Flexibility for various network environments
  - Next-hop settings
  - Route flap damping
  - Timer settings

Wonderful!
But there’s a catch...

BGP Configuration Affects Correctness

- BGP has serious problems
  - Frequently misconfigured [Mahajan2002]
  - Forwarding loops [Dube1999]
  - Persistent route oscillation [Griffin1999, Varadhan2000]
  - Slow convergence/suppressed routes [Labovitz2001, Mao2002]
  - Useless routing messages [Labovitz1999, Wang2002]
  - Security weaknesses [Beard2002, Kent2000]

BGP's configuration determines whether the protocol behaves correctly or not.

BGP configuration is a distributed program.

We need practical verification techniques.

Today: Stimulus-response Reasoning

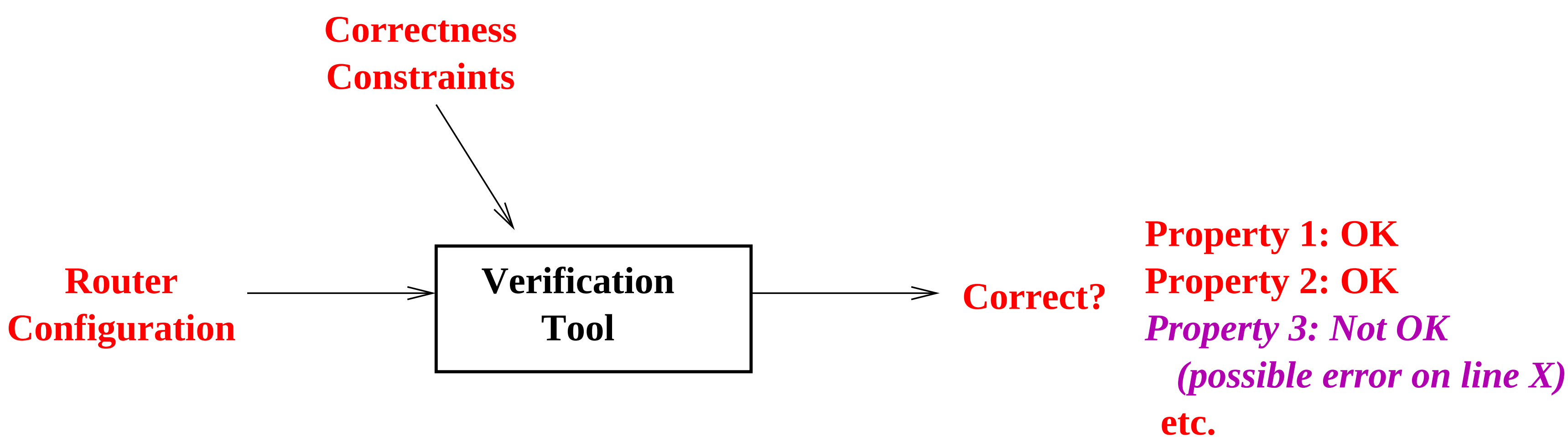
"What happens if I tweak this import policy?"
"Let’s just readjust this IGP weight..."
"New customer attachment point? Some cut-and-paste will fix that!"

Some time later, some "strange behavior" appears.
(OOPS! Revert.)

- Operators have a terrible "programming environment".
  - Configuration is ad hoc and painful.
  - Wastes operator time.
  - Suboptimal performance, angry customers.

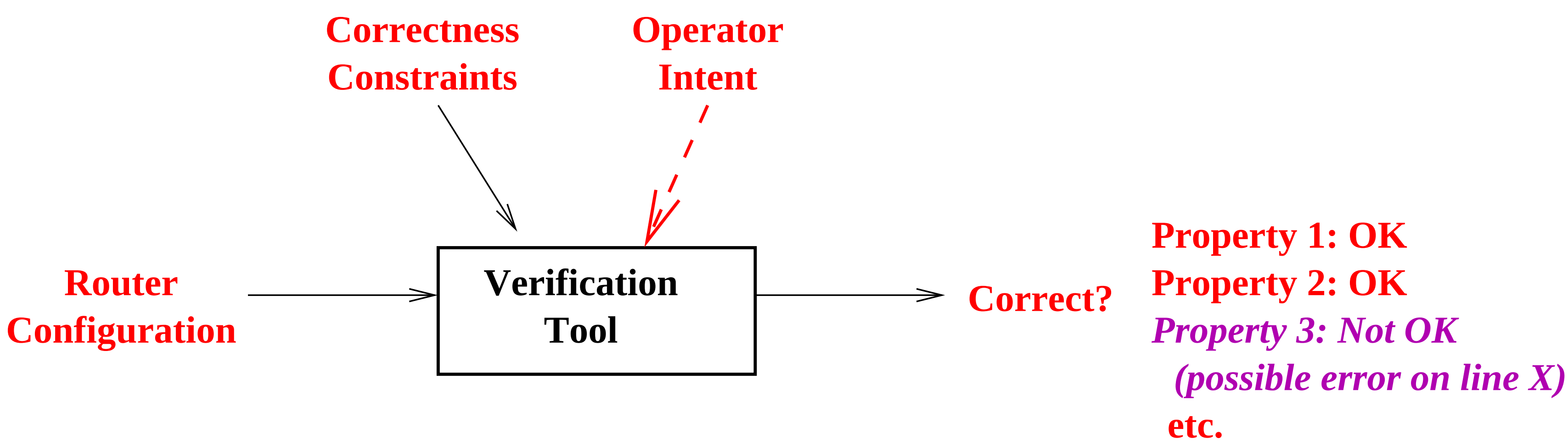
- Can’t check for errors by "seeing what happens".
  - Won’t catch misconfigured filters, redundant route reflectors, etc.

The Ideal Situation: Higher-level Reasoning



- **Verify** the behavior of a particular configuration.
  - Check "correctness properties".
    - (e.g., forwarding loops in iBGP configuration?)

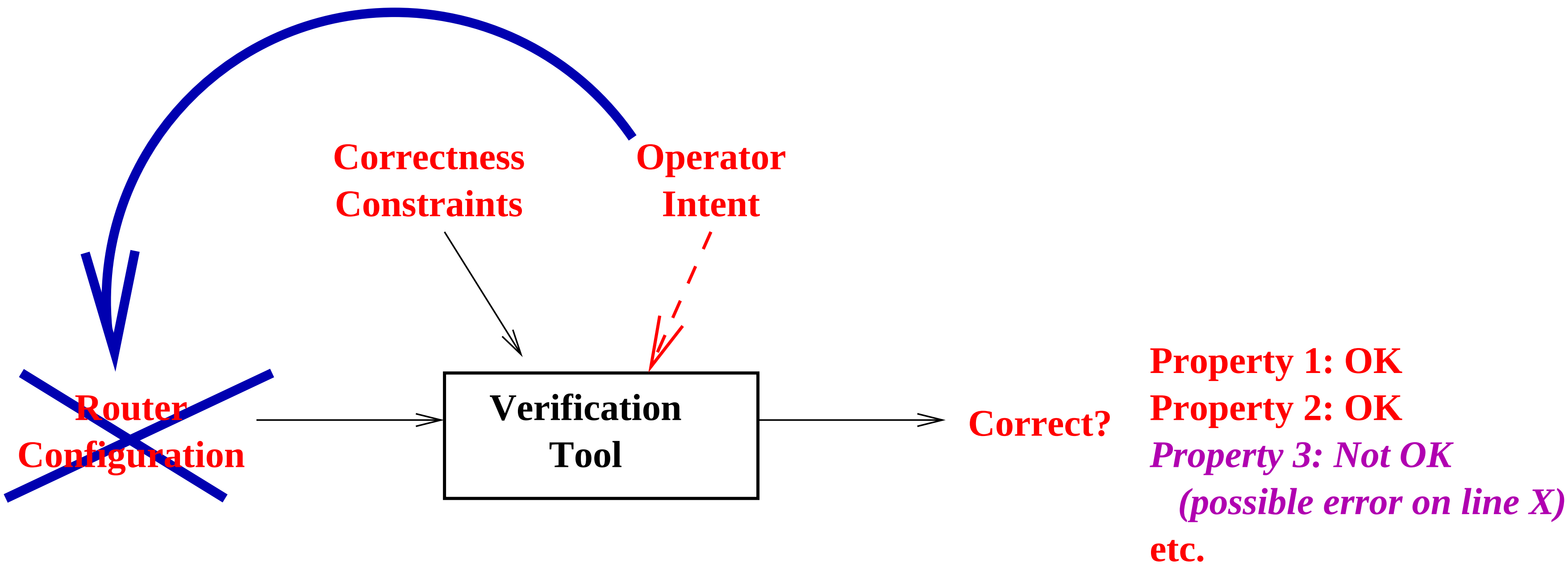
The Ideal Situation: Higher-level Reasoning



- **Verify** the behavior of a particular configuration.
  - Check "correctness properties".
    - (e.g., forwarding loops in iBGP configuration?)
  - Check that the configuration conforms to intended behavior.
    - (e.g., is aggregation appropriate? readvertising according to policy?)

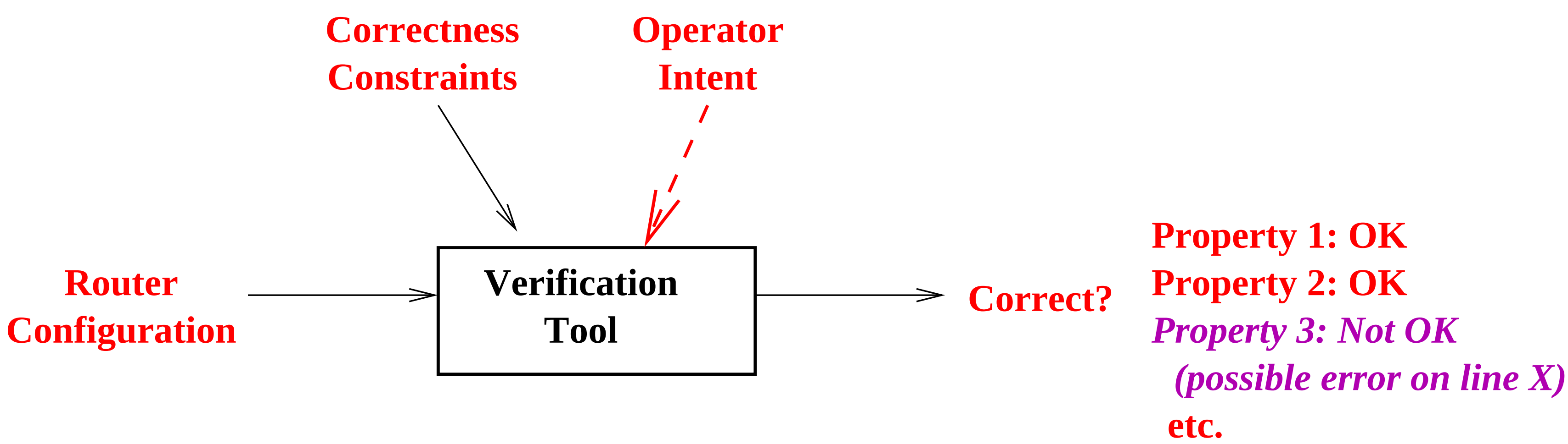
More than a band-aid fix.
Useful for any router configuration language.

Eventually: Higher-level Configuration



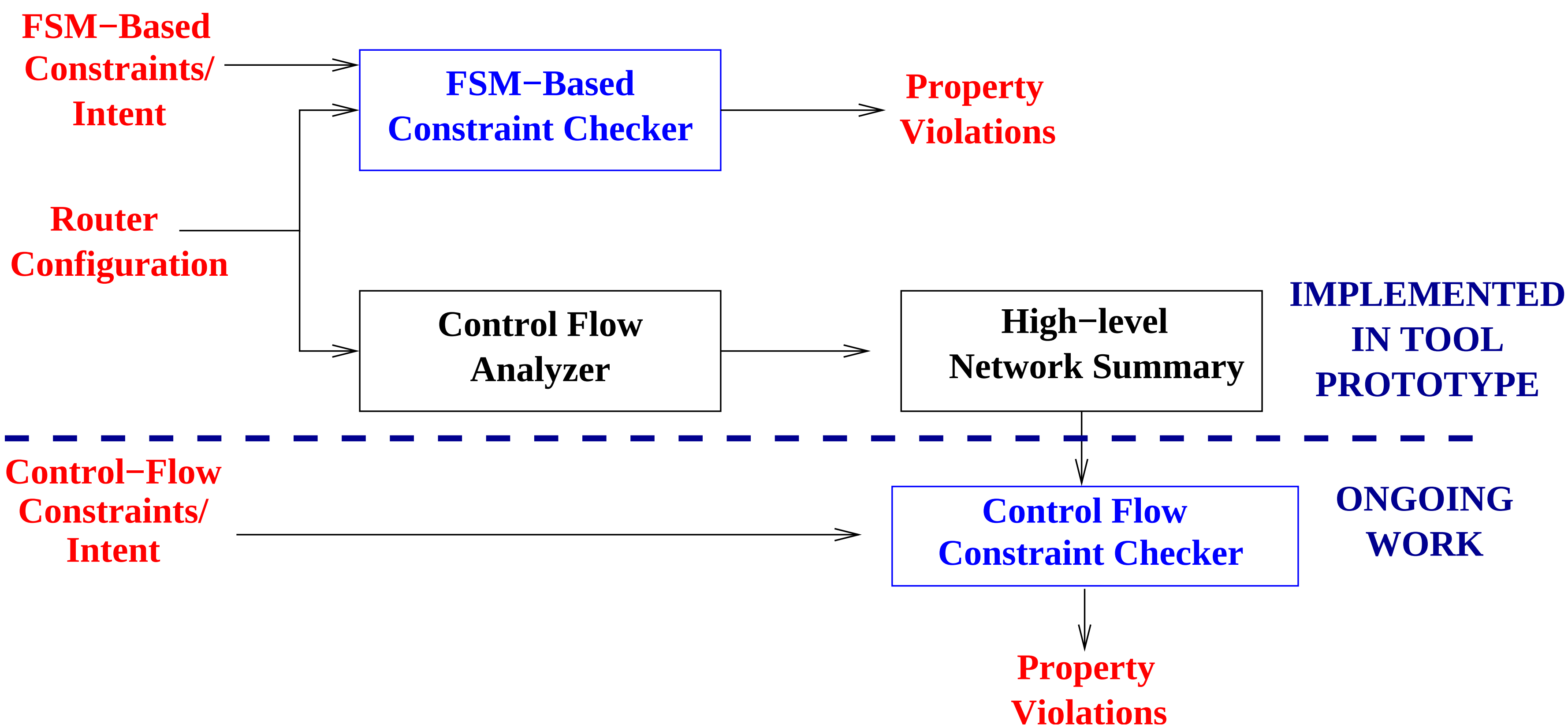
- **Specify** configuration based on intended behavior.
  - Configuring low-level mechanisms is error-prone.
  - Specifying high-level intended behavior makes sense.

Three Challenges



- **How to design the verification tool?**
- **How to express correctness constraints?**
- How to express operator intent?

Verification Tool Design



- How to express correctness constraints?
- How to express operator intent?

Correctness: The Routing Logic [FDNA 2003]

- **Validity:** Does it advertise invalid routes?
  - Bogus route injection, persistent forwarding loops, etc.
- **Visibility:** Does every valid path have a route?
  - Session resets, missing sessions, damped routes, etc.
- **Safety:** Will it converge to a unique, stable answer?
  - Policy-induced oscillation
- **Determinism:** Answer depend on orderings, etc.?
  - Irrelevant route alternatives can affect outcomes.
- **Information-flow control:** Expose information?
  - Accidental route leaks to neighbors, etc.

Correctness Constraints: Validity

Use the routing logic to express correctness constraints.

A **puts route to dest for** B

Reachability:
A **can reach dest via route**

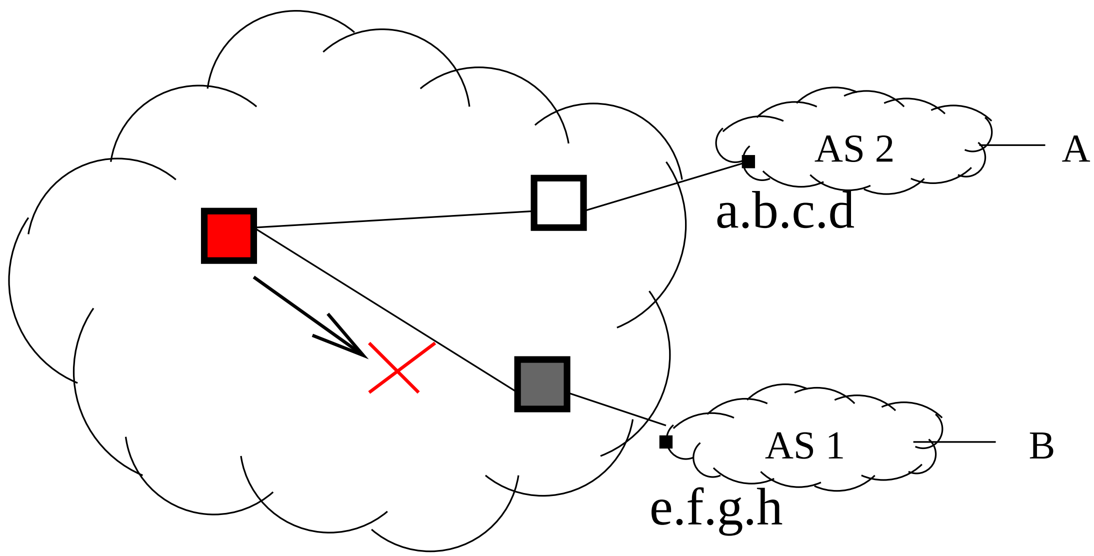
Policy conformance:
A **carries traffic to dest for** B

Progress:
route.next-hop makes progress along route to dest

Example: Validity

- One necessary, commonly violated condition:

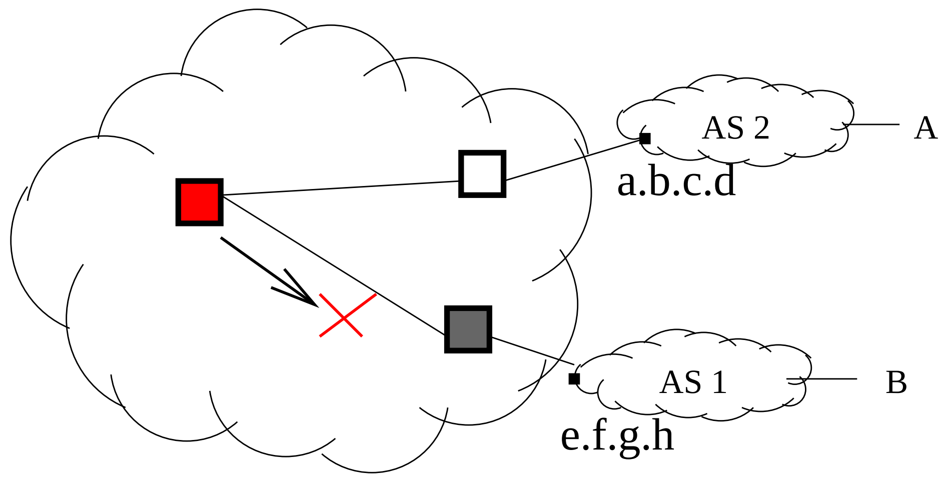
  next-hop reachability



Routes from AS 1 have next-hop e.f.g.h

If e.f.g.h not injected into IGP, some routes from within AS will fail.

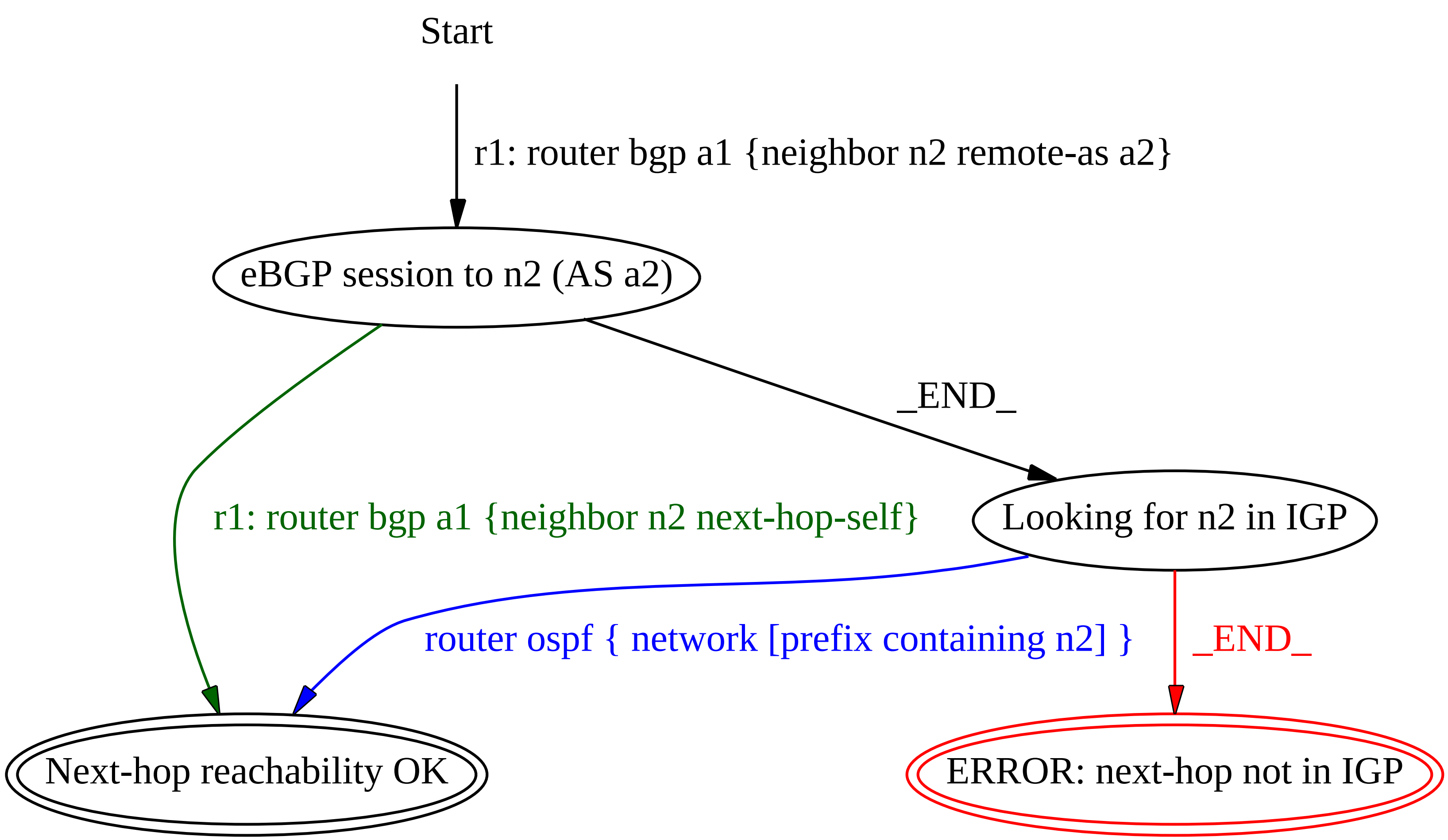
Validity: Checking Next-hop Reachability



- *Bad:* Copy/paste configurations and hope for the best. Traceroute-based debugging.
- *Better:* Apply the theory of the routing logic rules.

Next-Hop Reachability: An FSM-Based Rule

- The next-hop *refers to some router in the AS*, or
- The next-hop is *"injected" into the IGP*

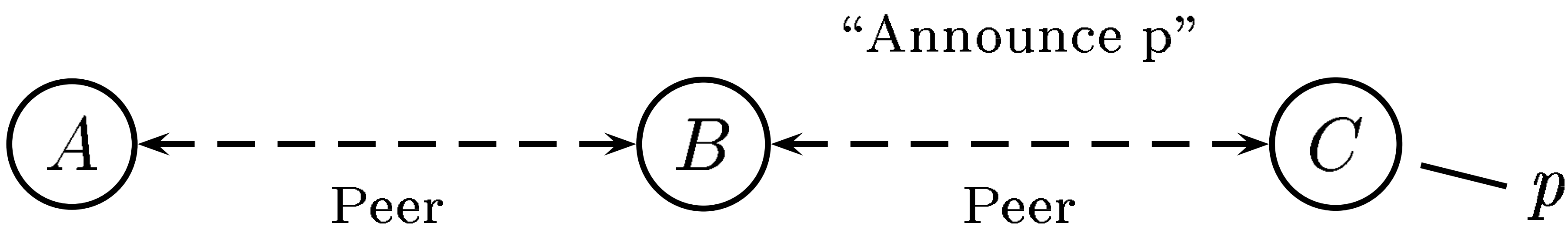


More on FSM-Based Rules

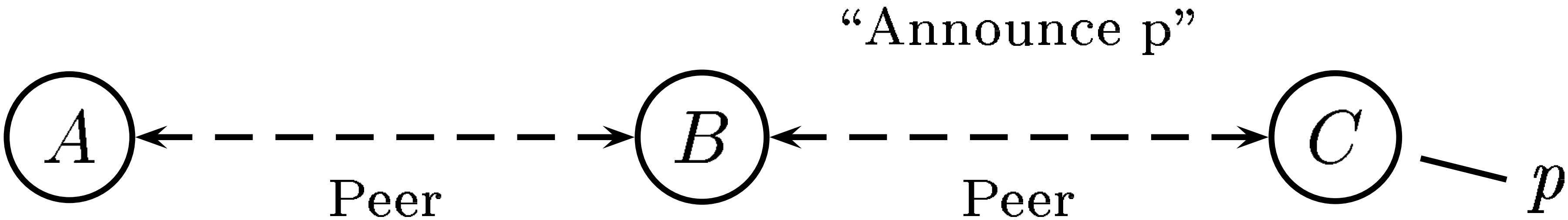
- Each correctness constraint: an FSM
  - specifies the verification procedure
  - gives useful information about the error
- Tool provides finite-state machinery and some rules
  - Rules are simple: 41 lines of code for next-hop test
- Figuring out "boundary" between users, developers.
  - Ruleset is part of the tool and is designed for extensibility.
  - Each rule is an FSM specification.

Example: Information-flow Control

Simple rule: don't advertise routes from one peer to other peers.



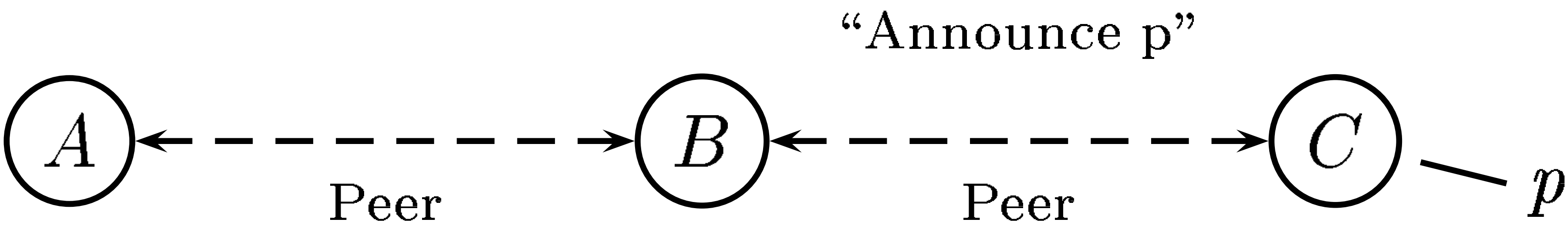
Today: Specifying Policy with Mechanism



Bad: Import/export route maps, ACLs, communities, etc.

```
neighbor 10.0.0.1 route-map EXPORT-A out
neighbor 192.168.0.1 route-map IMPORT-C in
...
ip community-list 1 permit 0:1000
...
route-map IMPORT-C permit 10
  set community 0:1000
!
...
route-map EXPORT-A permit 10
  match community 1
!
```

Tomorrow: High-level Policy Specification



Better: Use information-flow control principles.

Operator specifies intended flow.
Check against a control graph.

Key Challenge: Specification

(ongoing work)

Limitations and Ongoing Work

- Static analysis can’t catch everything.
  - *Idea:* "sandbox" to test configurations
- Constraint specification is not easy (yet).
  - *Idea:* statistical beliefs of "correctness"
- Verifying constraints across multiple ASes.
- Towards *intent-based* configuration languages.
  - Figuring out how to express operator intent.
  - Operator should specify intended goals, not the mechanism.

Conclusion

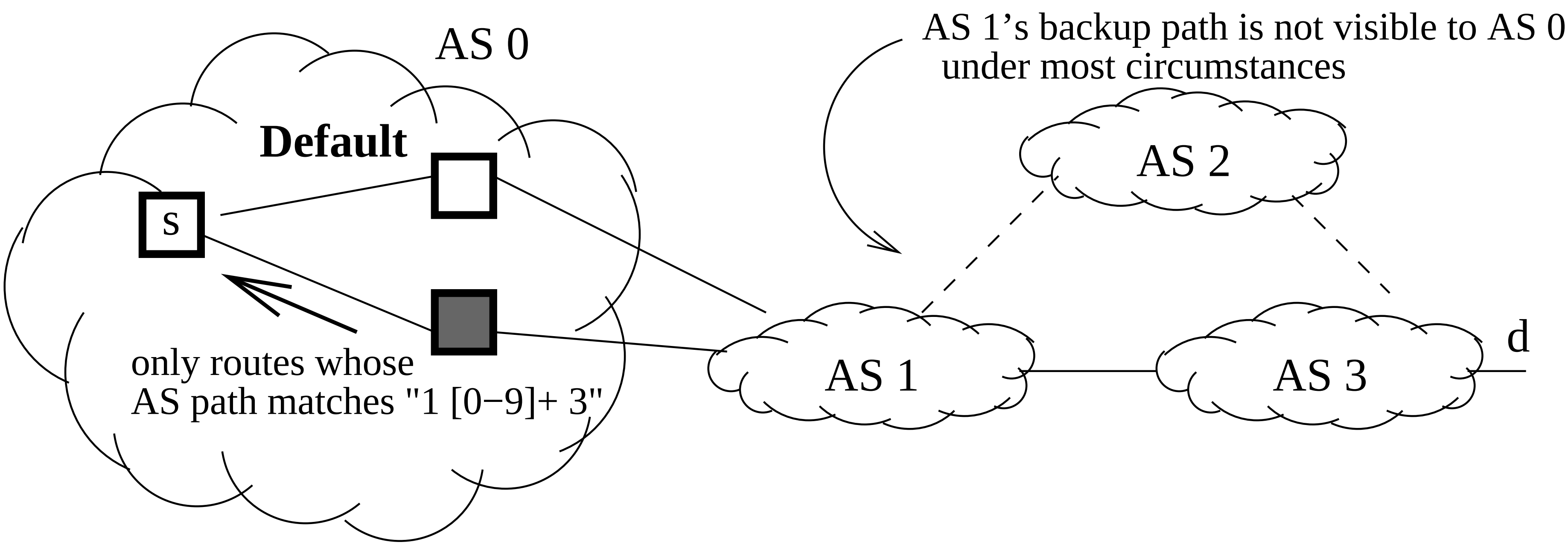
- BGP needs systematic verification techniques, regardless of configuration language.

- Verification can inspire the design of new configuration languages.

- Early version of the tool (RoLex) is available.
  - Several operators have downloaded the tool
  - Talking with Cisco about incorporating configuration checking on the routers themselves.

http://nms.lcs.mit.edu/bgp/rolex

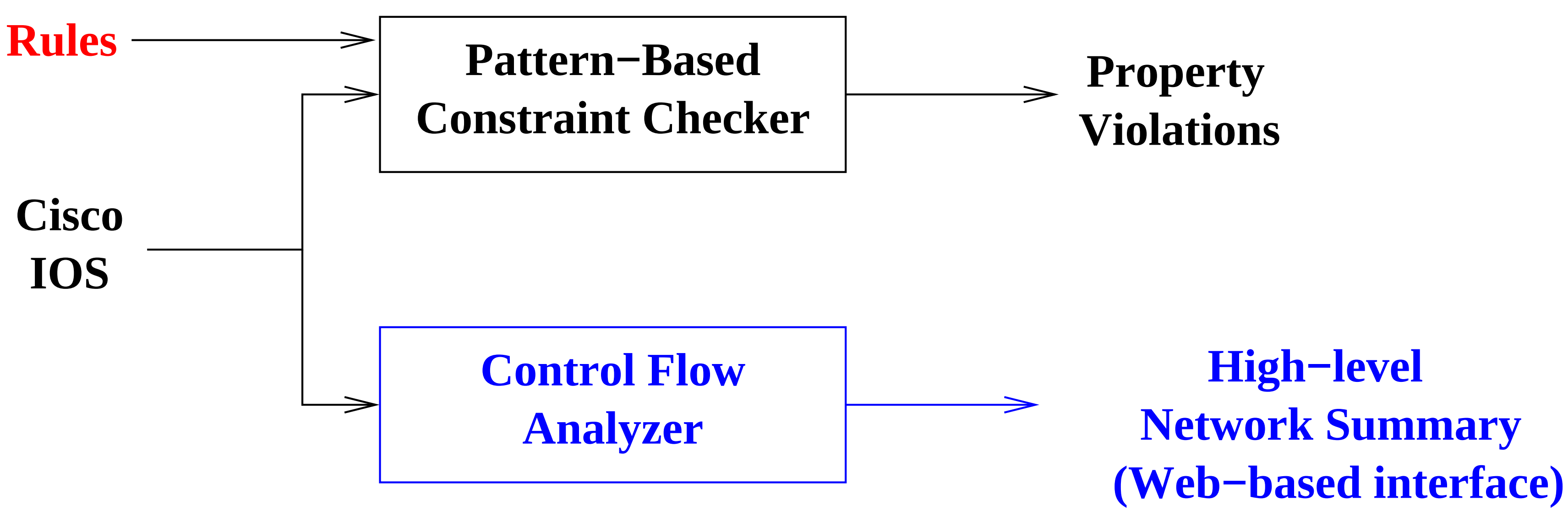
Why Not Model Checking?



- State-space explosion
- More importantly, some states may be hidden

Control Flow Analyzer

- Some constraints (e.g., import/export policies) best expressed in terms of higher-level semantics.
- Abstracts mechanisms, gives operators a higher-level view of network configuration.



Control Flow Analyzer: Features

- Graph the network at router-level, labelling route maps on edges.

- Database-backed Web interface.
  - View the number of BGP sessions to each AS.
  - View sessions, import and export route maps:
    - by router
    - associated with a particular remote AS
  - Easily compare policies across routers.

- Policies are "normalized" according to what they do, not what they are called.

Control Flow Analyzer: View By Neighbor AS

Routers Peering with AS 1239

Router	Neighbor	Neighbor AS	Import Route Map	Export Route Map
atlga-gw1	ebgp_AS1239_0	1239	25	26
cgcil-gw1	ebgp_AS1239_1	1239	25	26
dlxtx-gw2	ebgp_AS1239_2	1239	114	26
laxca-gw1	ebgp_AS1239_3	1239	25	26
			Show All Import	Show All Export

- Network-wide view of import/export policies to an AS.
- Easy to see when differences exist.

Control Flow Analyzer: View By Neighbor AS

Routers Peering with AS 1239

Router	Neighbor	Neighbor AS	Import Route Map	Export Route Map
atlga-gw1	ebgp_AS1239_0	1239	25	26
cgcil-gw1	ebgp_AS1239_1	1239	25	26
dlstx-gw2	ebgp_AS1239_2	1239	114	26
laxca-gw1	ebgp_AS1239_3	1239	25	26
			Show All Import	Show All Export

- Network-wide view of import/export policies to an AS.
- Easy to see when differences exist.

Other Information-flow Control Examples

Goal: Verify that route advertisements conform to intended information-flow policy.

- Partial peering
- Controlling prefix propagation
  - Bogons
  - "No Export" prefixes
- Conditional advertisements
- Signaling (e.g., with communities)

Towards Intent-based Configuration

Verification requires a specification of intent,
which can inspire configuration language design.

- How to specify the information flow lattice?
  - Must be intuitive.
  - Must express varying levels of detail (i.e., AS-level, session-level, prefix-level, etc.)
  - Must express positive requirements, too.

Understanding Correctness Constraints

- What correctness property does it address?
- What type of rule will verify it?
- One router, or multiple?
- Need information from other routing protocols?
- Need a specification of intended behavior?
- Need external information?
- Single AS, or more than one?
- Can static analysis catch the error?

Constraints: Next-hop Reachability

- What correctness property does it address? **validity**
- What type of rule will verify it? **pattern-based**
- One router, or multiple? **multiple**
- Need information from other routing protocols? **IGP**
- Need a specification of intended behavior? **no**
- Need external information? **no**
- Single AS, or more than one? **single AS**
- Can static analysis catch the error? **yes**

Constraints: eBGP Route propagation

- What property does it address? **information-flow**
- What type of rule will verify it? **control-flow**
- One router, or multiple? **multiple**
- Need information from other routing protocols? **(IGP)**
- Need a specification of intended behavior? **yes**
- Need external information? **no**
- Single AS, or more than one? **single AS**
- Can static analysis catch the error? **yes**

Towards Intent-based Configuration

Verification requires a specification of intent, which can inspire configuration language design.

- Expressing intended behavior will improve routing.
  - **Verification:** check existing configurations against intent.
  - **Synthesis:** generate configurations according to intent.
- **Example:** Controlling propagation of eBGP routes
  - ACLs, filters, communities, etc. are prone to mistakes.
  - Why not simply specify the intended policy?
- **Example:** Aggregation
  - Tradeoffs: hiding information about failures, TE, scalability.
  - Operator should specify intended goals, not the mechanism.