Structural Semantics Management: an Application of the Chase in Networking

*Anduo Wang**, Mubashir Anwar✔, Fangping Lan*, Matthew Caesar✔
*Temple University ✔UIUC

MASCOTS 2023 (October 16-18)

networking: a wonderful success

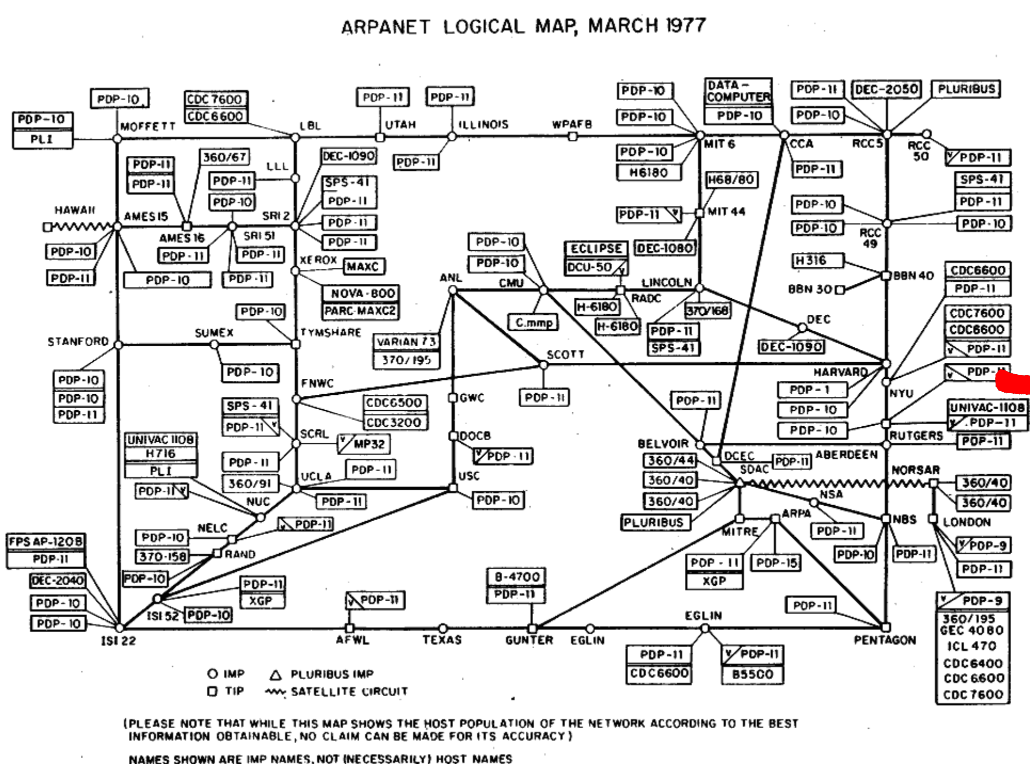
for everyday life

- Web, VoIP, social networking, content providers …

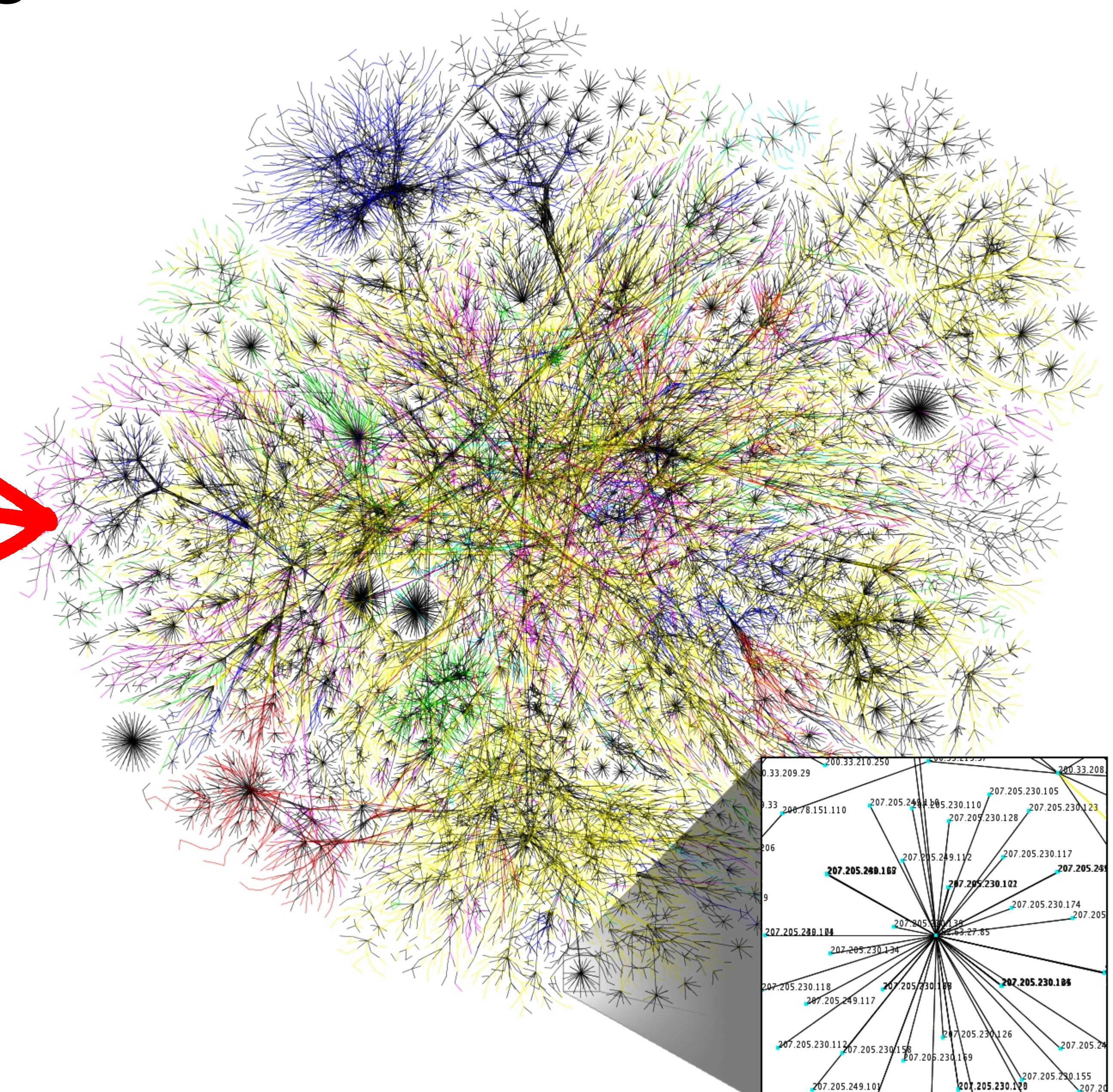
networking: a wonderful success

(Internet) a remarkable story

- from research experiment to global infrastructure



ARPANET, 1977



today

source: https://en.wikipedia.org/wiki/Internet

networking: a wonderful success

innovations take rapid transitions

Ahmed Khurshid., et al. *"VeriFlow: Verifying Network-Wide Invariants in Real Time"*
https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/khurshid
NSDI **2013**

3 years, $8.2 million

Veriflow Nabs $8.2 Million For Clever Ideas About Network Outage Prevention

JULY 19, 2016 BY DREW CONRY-MURRAY

Startup Veriflow Networks has landed $8.2 million in series A funding. The A round was led by Menlo Ventures, along with its existing investor New Enterprise Associates.

http://packetpushers.net/veriflow-nabs-8-2-million-clever-ideas-network-outage-prevention/

inside the 'Net': a different story

network systems

- increasingly complex

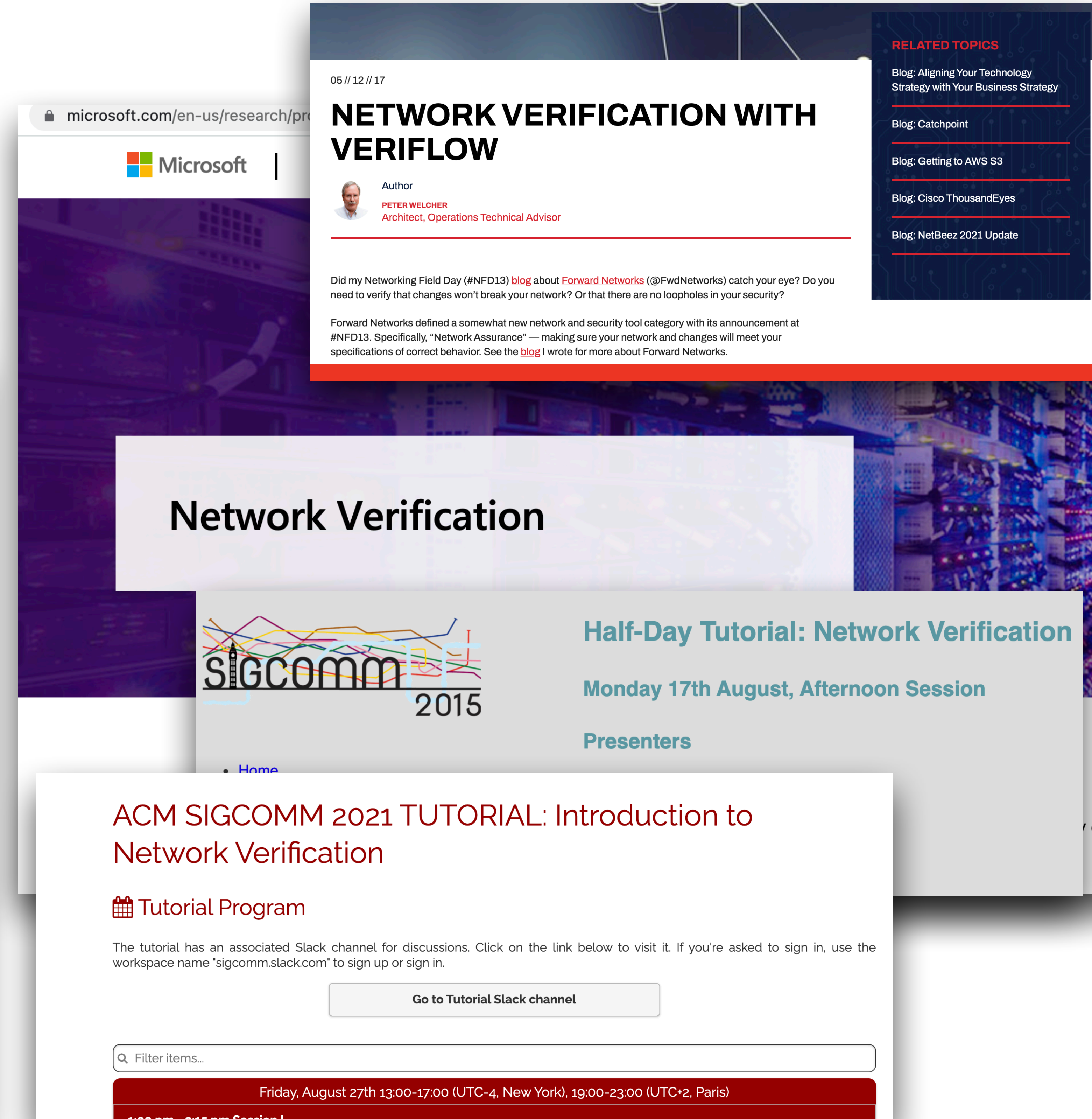
network management

- *a black art*

software-defined networking (SDN)

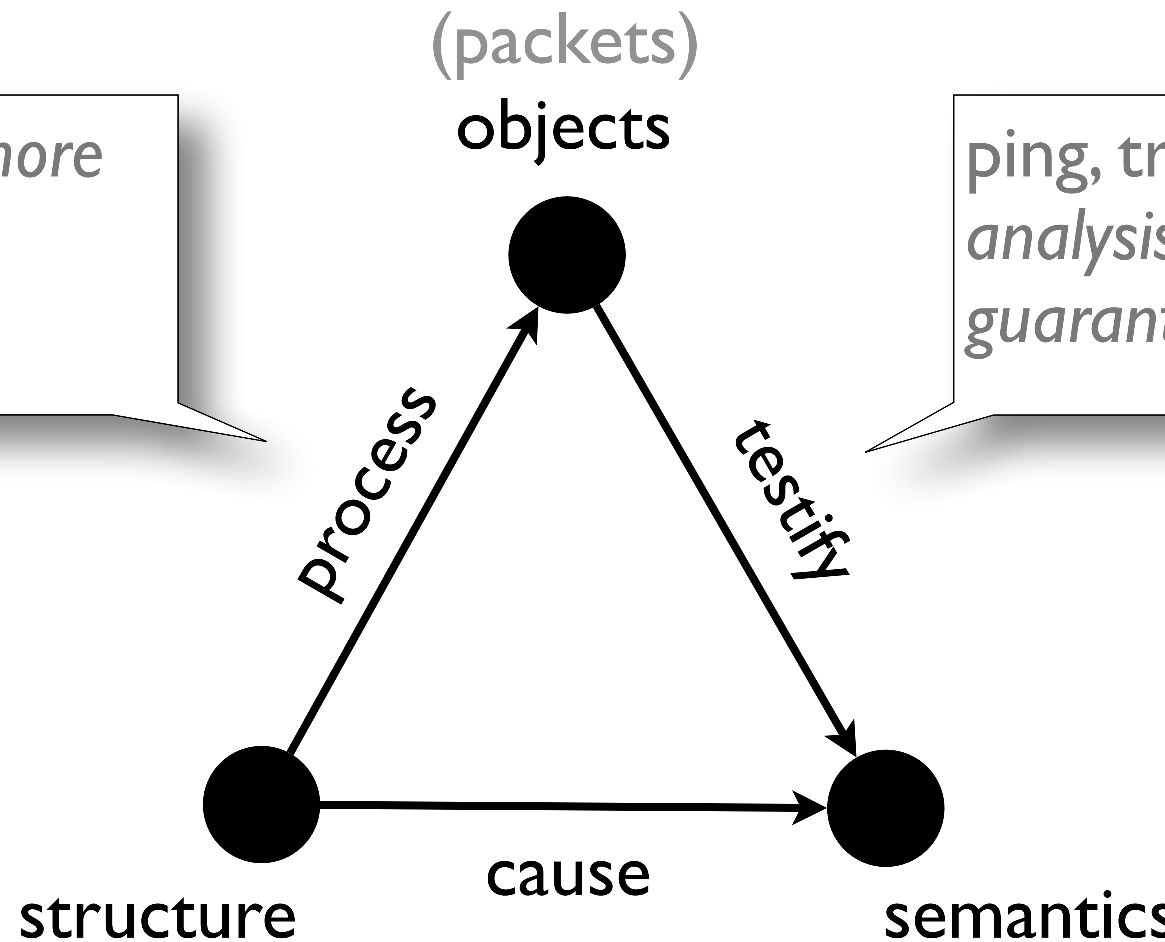
formal analysis

[NSDI 20] Tiramisu: Fast Multilayer Network Verification.
[SIGCOMM'02] Route oscillations in I-BGP route reflection.
[HotNets'20] Solver-Aided Multi-Party Configuration.
[NSDI'15] General Approach to Network Configuration Analysis.
[SIGCOMM'16] Fast Control Plane Analysis Using an Abstract Representation.
[TON'02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM'19] Validating Datacenters at Scale.
[CoNEXT 20] AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[NSDI 15] Checking Beliefs in Dynamic Net- works.
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network config- uration verification through model checking
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[INFOCOM 18] Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[INFOCOM 05] On static reachability analysis of IP networks
[SIGCOMM 20] Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN
[HotNets 20] Incremental Network Configuration Verification
[NSDI 20] APKeep: Realtime Verification for Real Networks
… …



network management, an anatomy

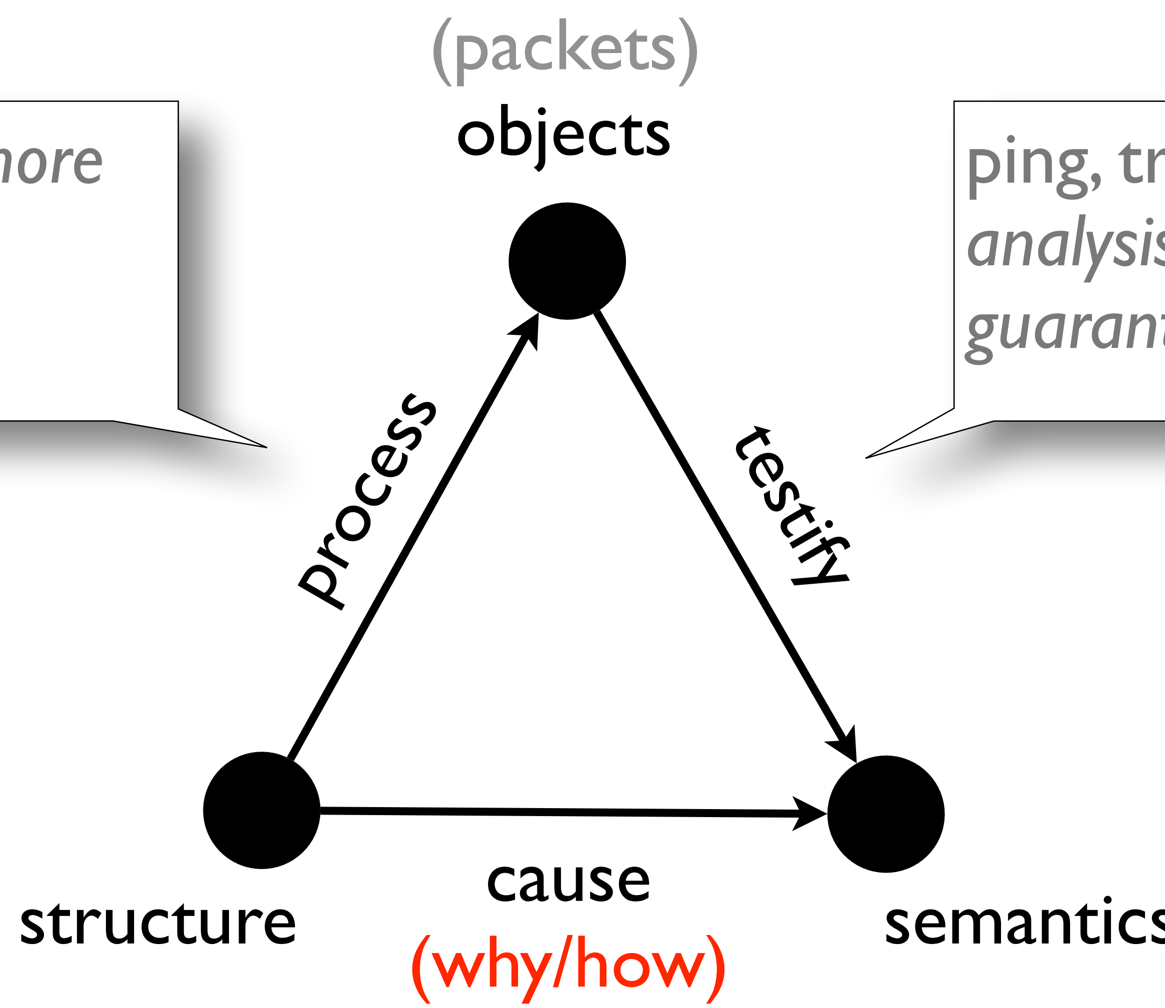
ad-hoc protocols, *more systematic software defined networking*



ping, traceroute, *formal analysis with stronger guarantee*

network management, an anatomy

ad-hoc protocols, *more systematic software defined networking*

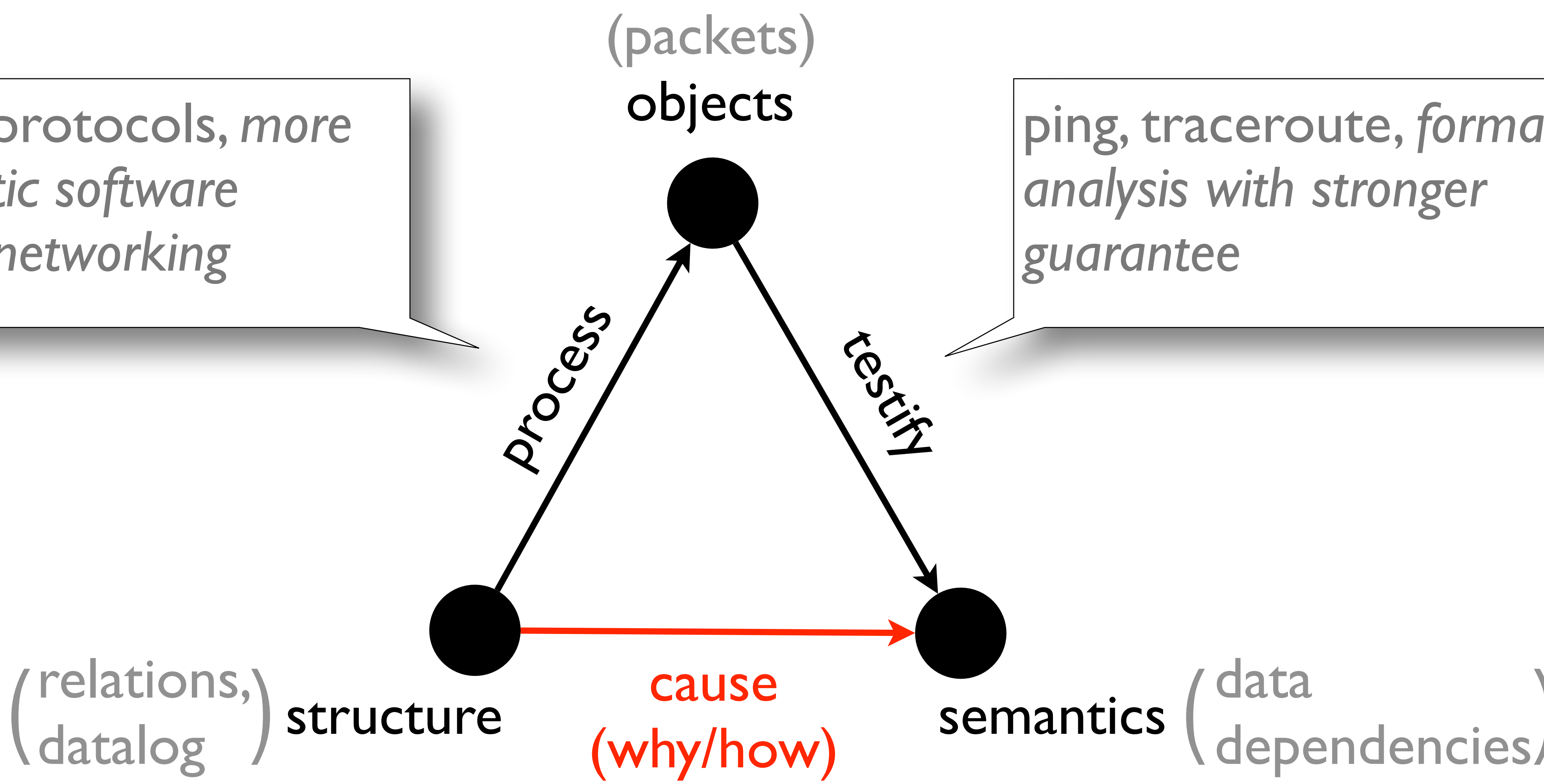


ping, traceroute, *formal analysis with stronger guarantee*

network management, this work

ad-hoc protocols, *more systematic software defined networking*

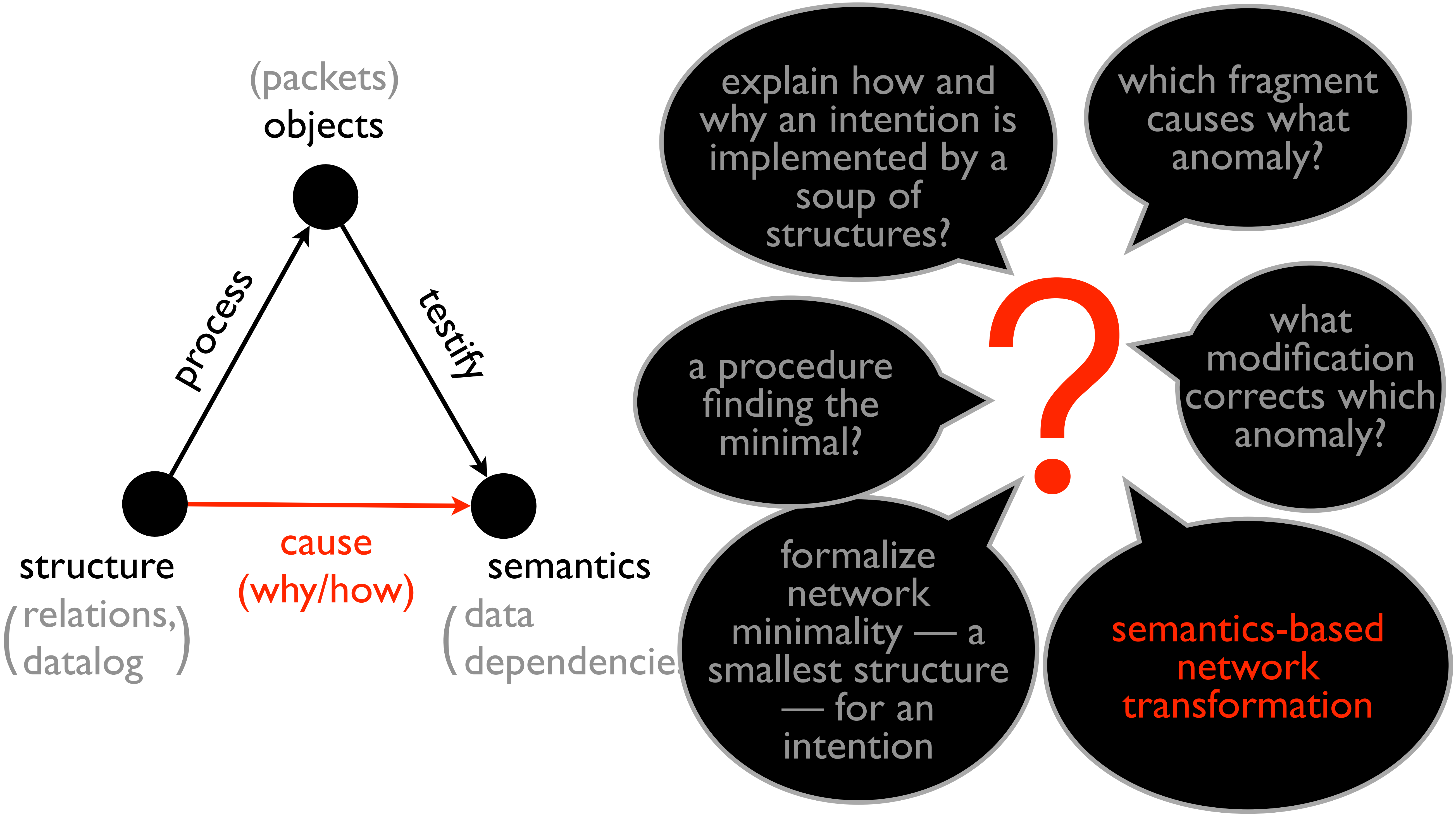
ping, traceroute, *formal analysis with stronger guarantee*



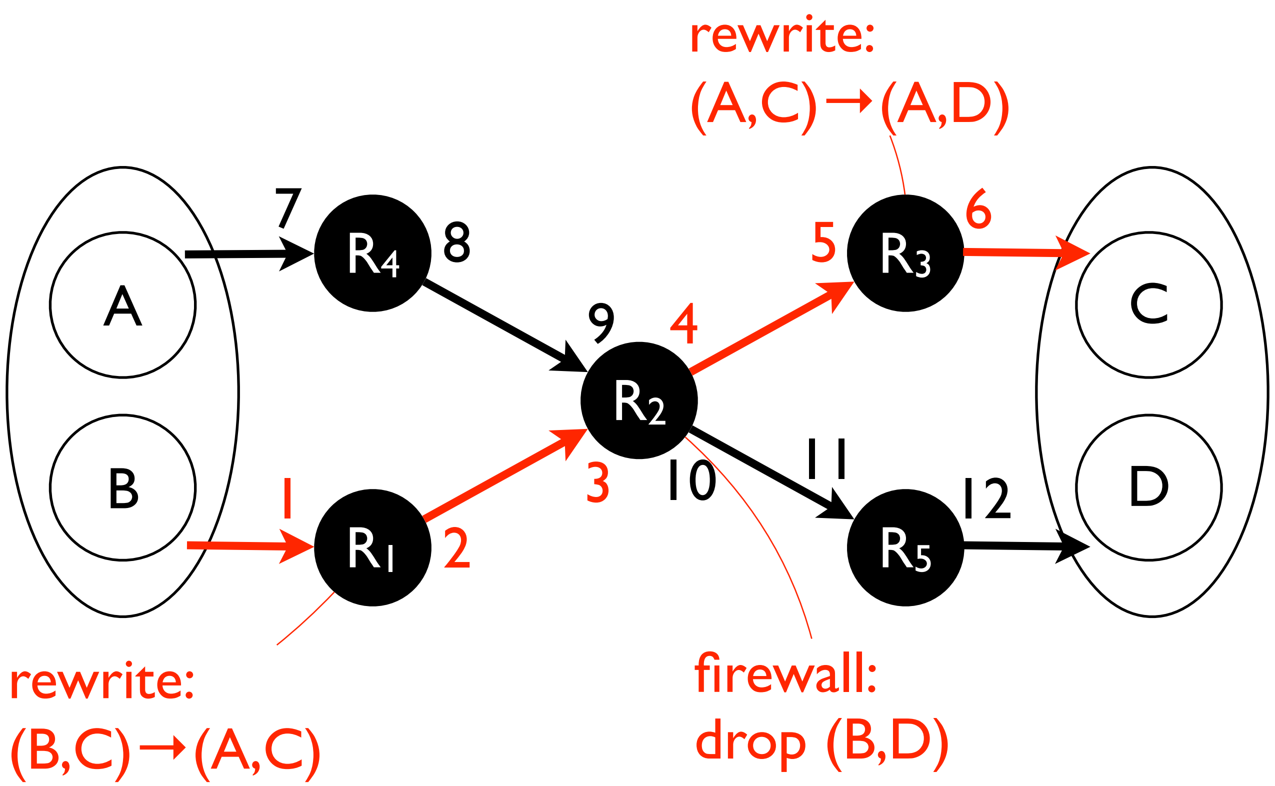
our approach: relational database and its neighboring disciplines — deductive reasoning, knowledge representation and reasoning, logic programming, artificial intelligence …

(the chase)

network management, this work



an example

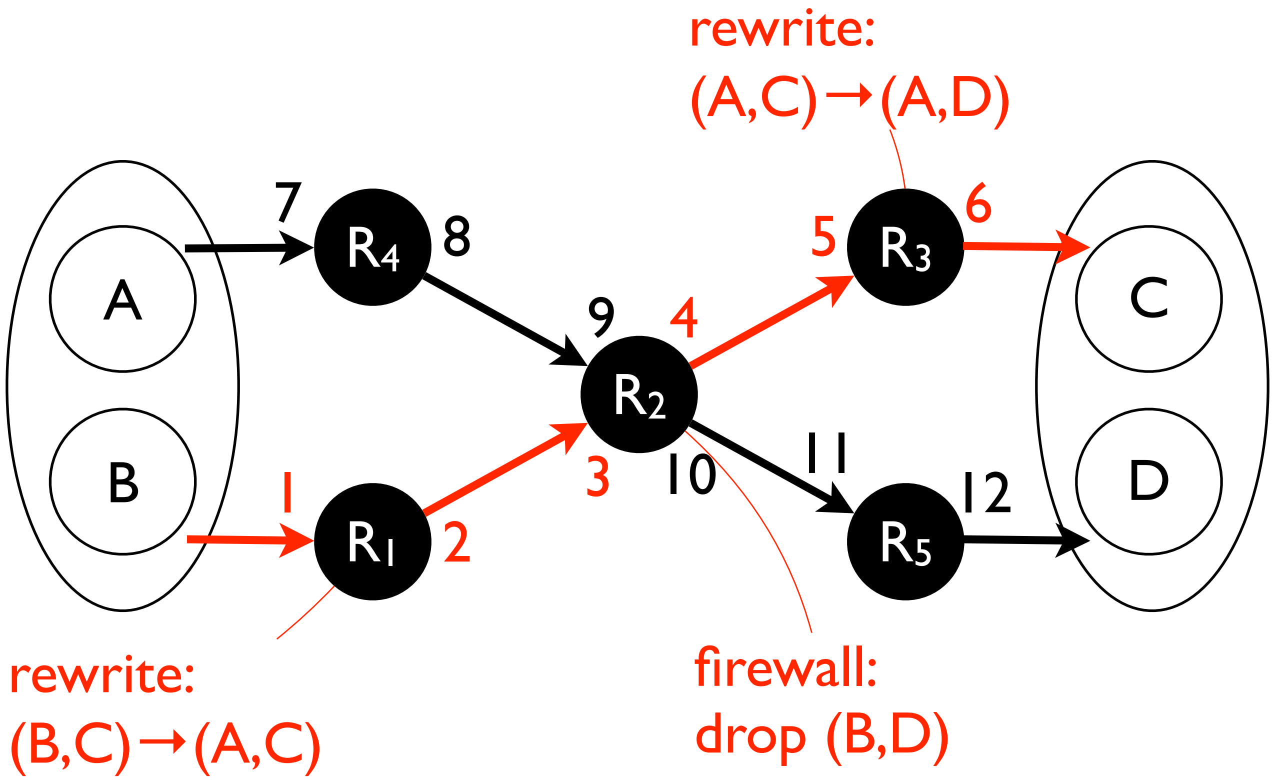


is the firewall *effectively* installed?

i.e., can hosts belong to B still send traffic to those in C?

?

an example

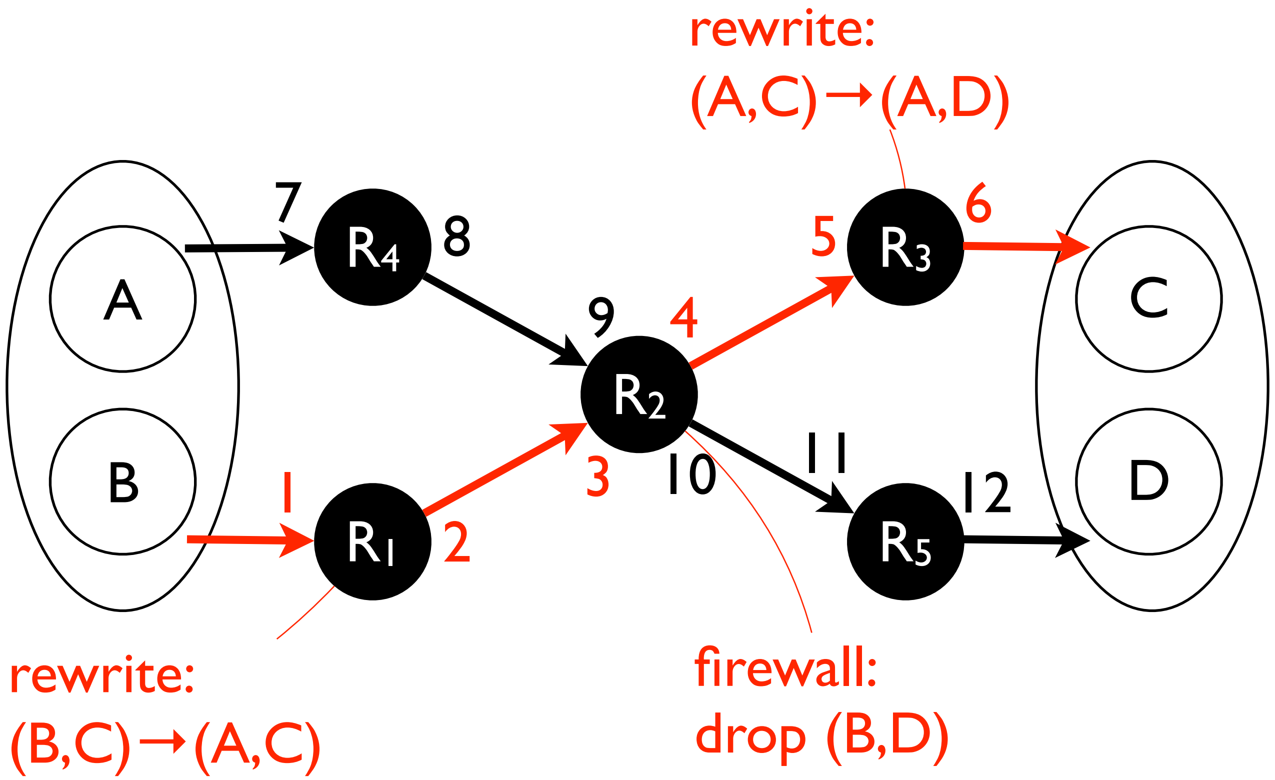


is the firewall *effectively* installed?

i.e., can hosts belong to B still send traffic to those in C?

?

an example



is the firewall *effectively* installed?

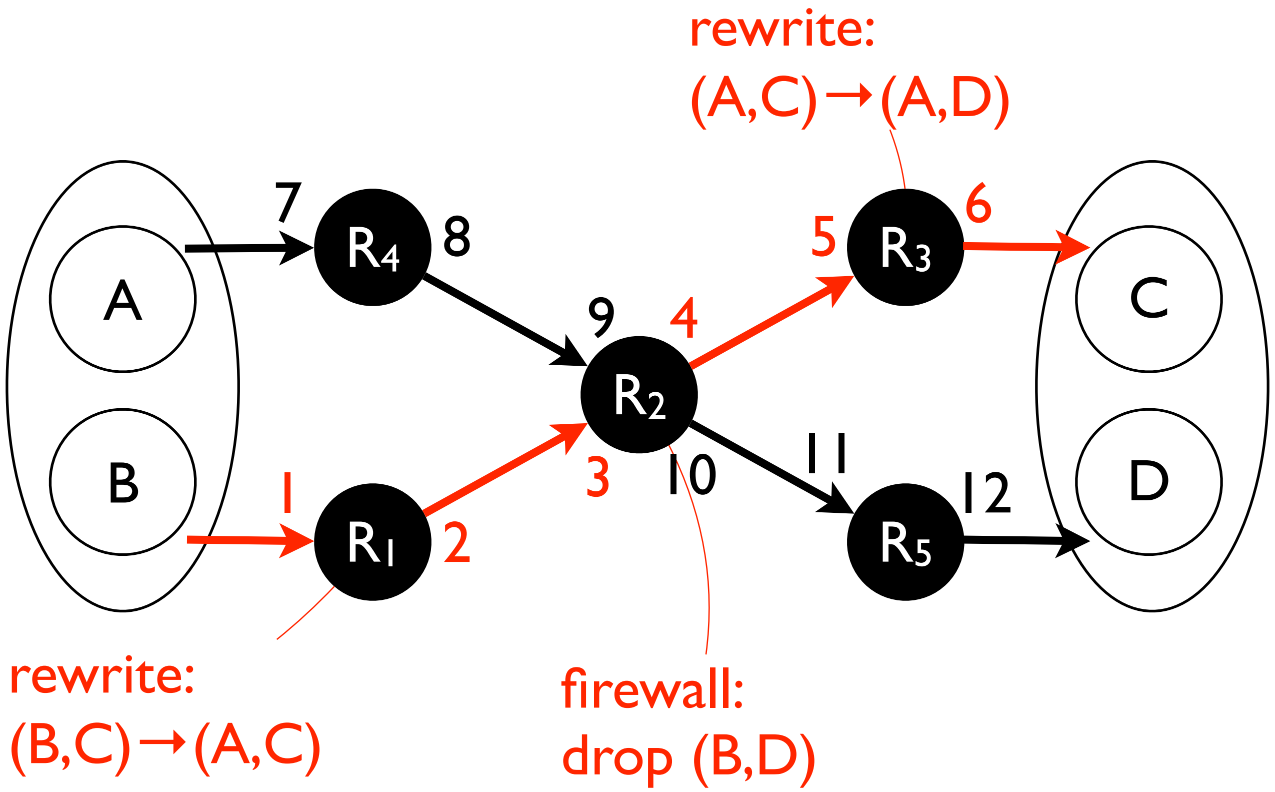
i.e., can hosts belong to B still send traffic to those in C?

?

existing approach

- inject input traffic into B, and observe output at C

semantics-based network transformation



is the firewall *effectively* installed?

i.e., can hosts belong to B still send traffic to those in C?

?

plain forwarding program
P

a set of policies, characterizing legitimate packets
Σ

structural rewrite

→ *P'*

P' reflects Σ, renders the security hole

a formulation with the chase

a datalog query

P

data dependencies

Σ

the chase

P'

P'=P on data satisfying Σ

plain forwarding program

P

a set of policies, characterizing legitimate packets

Σ

structural rewrite

P'

P' reflects Σ, renders the security hole

a formulation with the chase

a datalog query
P

data dependencies
Σ

P'

P'=P on data satisfying Σ

the chase

- given a data dependency σ (∈Σ)
- eliminates "useless" evaluation in *P* by an intuitive structural rewrite (adding/collapsing/updating elements in the query)

the chase

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
      F(f,x3,y3,2,3), F(f,x4,y4,3,4),
      F(f,x5,y5,4,5), F(f,x6,y6,5,6),
      F(f,x7,y,6,y).
%% permitting header modifications
%% F(Flow, Source, Destination, Location, Next-hop)
```

```
k: a key dependency
y=y' :- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

?

tableau query

P	F	S	D	L	N
	f	x	y_1	x	1
	f	x_2	y_2	1	2
	f	x_3	y_3	2	3
body	f	x_4	y_4	3	4
	f	x_5	y_5	4	5
	f	x_6	y_6	5	6
	f	x_7	y	6	7
head		x	y		

the chase

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
     F(f,x3,y3,2,3), F(f,x4,y4,3,4),
     F(f,x5,y5,4,5), F(f,x6,y6,5,6),
     F(f,x7,y,6,y).
%% permitting header modifications
%% F(Flow, Source, Destination, Location, Next-hop)
```

```
k: a key dependency
y=y' :- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

?

tableau query

P	F	S	D	L	N
	f	x	y_1	x	1
	f	x_2	y_1	1	2
	f	x_3	y_3	2	3
body	f	x_4	y_4	3	4
	f	x_5	y_5	4	5
	f	x_6	y_6	5	6
	f	x_7	y	6	7
head		x	y		

the chase

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
      F(f,x3,y3,2,3), F(f,x4,y4,3,4),
      F(f,x5,y5,4,5), F(f,x6,y6,5,6),
      F(f,x7,y,6,y).
%% permitting header modifications
%% F(Flow, Source, Destination, Location, Next-hop)
```

```
k: a key dependency
y=y' :- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

?

tableau query

P	F	S	D	L	N
	f	x	y_1	x	1
	f	x_2	y_1	1	2
	f	x_3	y_1	2	3
body	f	x_4	y_1	3	4
	f	x_5	y_1	4	5
	f	x_6	y_1	5	6
	f	x_7	y_1	6	7
head		x	y_1		

the chase

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
     F(f,x3,y3,2,3), F(f,x4,y4,3,4),
     F(f,x5,y5,4,5), F(f,x6,y6,5,6),
     F(f,x7,y,6,y).
%% permitting header modifications
%% F(Flow, Source, Destination, Location, Next-hop)
```

```
k: a key dependency
y=y':- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

```
/* P': the result of chasing r with k */
R(x,y1 ) :-F(f,x,y1,x,1),F(f,x2,y1,1,2),
      F(f,x3,y1,2,3), F(f,x4,y1,3,4),
      F(f,x5,y1,4,5), F(f,x6,y1,5,6),
      F(f,x7,y1,6,y1 ).
```

tableau query

P	F	S	D	L	N
	f	x	y_1	x	1
	f	x_2	y_1	1	2
	f	x_3	y_1	2	3
body	f	x_4	y_1	3	4
	f	x_5	y_1	4	5
	f	x_6	y_1	5	6
	f	x_7	y_1	6	7
head		x	y_1		

the chase

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
     F(f,x3,y3,2,3), F(f,x4,y4,3,4),
     F(f,x5,y5,4,5), F(f,x6,y6,5,6),
     F(f,x7,y,6,y).
%% permitting header modifications
%% F(flow, source, destination, location, next-hop)
```

```
k: a key dependency
y=y' :- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

```
/* P': the result of chasing r with k */
R(x,y1 ) :-F(f,x,y1,x,1),F(f,x2,y1,1,2),
      F(f,x3,y1,2,3), F(f,x4,y1,3,4),
      F(f,x5,y1,4,5), F(f,x6,y1,5,6),
      F(f,x7,y1,6,y1 ).
```

the chase, limitation

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
     F(f,x3,y3,2,3), F(f,x4,y4,3,4),
     F(f,x5,y5,4,5), F(f,x6,y6,5,6),
     F(f,x7,y,6,y).
%% permitting header modifications
%% F(flow, source, destination, location, next-hop)
```

```
k: a key dependency
y=y′:- F(f,x,y,u,w),
    F(f,x′,y′,u′,w′).
```

```
k′: k restricted to R2
and source other than
1.2.3.4
y=y′:- F(f,x,y,2,3),
     F(f,x′,y′,3,4),
     x≠1.2.3.4.
```

```
/* P′: the result of chasing r with k */
R(x,y1 ) :-F(f,x,y1,x,1),F(f,x2,y1,1,2),
      F(f,x3,y1,2,3), F(f,x4,y1,3,4),
      F(f,x5,y1,4,5), F(f,x6,y1,5,6),
      F(f,x7,y1,6,y1 ).
```

?

the chase, limitation

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
      F(f,x3,y3,2,3), F(f,x4,y4,3,4),
      F(f,x5,y5,4,5), F(f,x6,y6,5,6),
      F(f,x7,y,6,y).
%% permitting header modifications
%% F(flow, source, destination, location, next-hop)
```

P	F	S	D	L	N
		...			
body	f	x_3	y_3	2	3
	f	x_4	y_4	3	4
		...			
head		x	y		

$x_3 \neq$ 1.2.3.4?

```
k: a key dependency
y=y' :- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

```
k': k restricted to R2
and source other than
1.2.3.4
y=y' :- F(f,x,y,2,3),
     F(f,x',y',3,4),
     x≠1.2.3.4.
```

```
/* P': the result of chasing r with k */
R(x,y1 ) :-F(f,x,y1,x,1),F(f,x2,y1,1,2),
       F(f,x3,y1,2,3), F(f,x4,y1,3,4),
       F(f,x5,y1,4,5), F(f,x6,y1,5,6),
       F(f,x7,y1,6,y1 ).
```

?

the chase, strength

dependency σ ($\in\Sigma$) as general implication

- $\phi(X,Y) \rightarrow \exists Z.\psi(Y,Z)$
  - X,Y,Z are vectors of variables, ϕ and ψ are conjunction of predicates (including equations)
  - subsume all common (integrity) constraints in database applications

chasing with a set Σ is *Church-Rosser*

- terminates with a unique result
  - the order of applying σ ($\in\Sigma$) is insignificant

the chase, strength & limitation

dependency σ (∈Σ) as general implication

- $\phi(X,Y) \rightarrow \exists Z.\psi(Y,Z)$
  - X,Y,Z are vectors of variables, ϕ and ψ are conjunction of predicates (including equations)
  - subsume all common (integrity) constraints in database applications

too limited for network policies

chasing with a set Σ is *Church-Rosser*

- terminates with a unique result
  - the order of applying σ (∈Σ) is insignificant

extend the chase to networking

dependency σ ($\in\Sigma$) as general implication

- $\phi(X,Y) \rightarrow \exists Z.\psi(Y,Z)$
  - X,Y,Z are vectors of variables, ϕ and ψ are conjunction of predicates (including equations)
  - subsume all common (integrity) constraints in database applications

chasing with a set Σ is *Church-Rosser*

- terminates with a unique result
  - the order of applying σ ($\in\Sigma$) is insignificant

our contribution

richer dependencies (of network policies)

retain the *Church-Rosser* property

extend the chase, insight

```
/* P: reachability (forwarding) along
R1R2R3 */
R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),
     F(f,x3,y3,2,3), F(f,x4,y4,3,4),
     F(f,x5,y5,4,5), F(f,x6,y6,5,6),
     F(f,x7,y,6,y).
%% permitting header modifications
%% F(flow, source, destination, location, next-hop)
```

P	F	S	D	L	N
			...		
body	f	x_3	y_3	2	3
	f	x_4	y_4	3	4
			...		
head		x	y		

$x_3 \neq$1.2.3.4?

```
k: a key dependency
y=y' :- F(f,x,y,u,w),
    F(f,x',y',u',w').
```

```
k': k restricted to R2
and source other than
1.2.3.4
y=y' :- F(f,x,y,2,3),
     F(f,x',y',3,4),
     x≠1.2.3.4.
```

```
/* P': the result of chasing r with k */
R(x,y1 ) :-F(f,x,y1,x,1),F(f,x2,y1,1,2),
      F(f,x3,y1,2,3), F(f,x4,y1,3,4),
      F(f,x5,y1,4,5), F(f,x6,y1,5,6),
      F(f,x7,y1,6,y1 ).
```

view *P* (body) as an incomplete database instance,
evaluate **k'** on *D*

fauré-log

a datalog variant for incomplete information

```
/* network query on symbolic state */
H(u)[C(u)] :- B1(u1),···, Bn(un), [C1(u1),···,Cn(un)].
%% u,u1,...,un are tuples with constants and variables(conditioned by
constraints C,C1,…,Cn)
```

fauré-log, richer dependencies

a datalog variant for incomplete information

```
/* network query on symbolic state */
H(u)[C(u)] :- B1(u1),···, Bn(un), [C1(u1),···,Cn(un)].
%% u,u1,...,un are tuples with constants and variables(conditioned by
constraints C,C1,…,Cn)
```

fauré-log dependencies, for network policies

```
/* network dependencies chasable on symbolic states */
H(u) :- B1(u1),···,Bn (un ), [C1(u1),···,Cn(un)]. % tgd: the presence
of Bi's under the conditions Ci's implies H

[x/y, C(u)] :- B1(u1),···, Bn(un),[C1(u1),···,Cn(un)]. % egd:
substitute symbol x for y, Ci's are conjunction of (in)equality and
auxiliary predicates
```

generalize the chase step

generalize the substitution of the chase to fauré-log evaluation

Algorithm 1: The chase with *fauré*-dependency

input : *fauré-log* rule $\mathtt{r} : \mathtt{H_r} :- \mathtt{B_r}[\phi_\mathtt{r}]$,
fauré-dependency $\sigma : \mathtt{H}_\sigma[\underline{\mathtt{x/y}}, \psi_\sigma] :- \mathtt{B}_\sigma[\phi_\sigma]$

output: $\mathtt{r} \rightarrow_\sigma \mathtt{r}'$

equality generation dependency *(egd)*

```
instantiate B_r[φ_r] into c-tables D ;
let q be H_σ[ψ_σ] :- B_σ[φ_σ] ;
let H'_σ[ψ'_σ] = q(D) by fauré-log evaluation ;
if H'_σ[ψ_σ] is empty;
 then halt
   else
       let φ'_r = φ_r{x/y}, φ'_σ = φ_σ{x/y} ;
       if φ'_r ∧ φ'_σ ∧ ψ'_σ is UNSAT then halt;
       else let r' be H_r{x/y} :- B_r{x/y}, H'_σ, [φ'_r, φ'_σ, ψ'_σ]
         return r';
   end
end
```

collapsing atoms in the rule

generalize the chase step

generalize the substitution of the chase to fauré-log evaluation

Algorithm 1: The chase with *fauré*-dependency

input : *fauré-log* rule $\mathtt{r} : \mathtt{H}_\mathtt{r} :- \mathtt{B}_\mathtt{r}[\phi_\mathtt{r}]$,
fauré-dependency $\sigma : \mathtt{H}_\sigma[\mathtt{x}/\mathtt{y}, \psi_\sigma] :- \mathtt{B}_\sigma[\phi_\sigma]$

output: $\mathtt{r} \rightarrow_\sigma \mathtt{r}'$

```
instantiate B_r[φ_r] into c-tables D ;
let q be H_σ[ψ_σ] :- B_σ[φ_σ] ;
let H'_σ[ψ'_σ] = q(D) by fauré-log evaluation ;
if H'_σ[ψ'_σ] is empty;
 then halt
    else
        let φ'_r = φ_r{x/y}, φ'_σ = φ_σ{x/y} ;
        if φ'_r ∧ φ'_σ ∧ ψ'_σ is UNSAT then halt;
        else let r' be H_r{x/y} :- B_r{x/y}, H'_σ, [φ'_r, φ'_σ, ψ'_σ]
          return r';
    end
end
```

tuple generation dependency *(tgd)*

generate the new (implied) atoms by fauré-log

add new atoms in the rule

generalize the chase step

generalize the substitution of the chase to fauré-log evaluation

Algorithm 1: The chase with *fauré*-dependency

input : *fauré-log* rule $\mathtt{r} : \mathtt{H_r} :- \mathtt{B_r}[\phi_{\mathtt{r}}]$,
fauré-dependency $\sigma : \mathtt{H}_\sigma[\mathtt{x}/\mathtt{y}, \psi_\sigma] :- \mathtt{B}_\sigma[\phi_\sigma]$

output: $\mathtt{r} \rightarrow_\sigma \mathtt{r}'$

instantiate $\mathtt{B_r}[\phi_{\mathtt{r}}]$ into c-tables $\mathtt{D}$;
let $\mathtt{q}$ be $\mathtt{H}_\sigma[\psi_\sigma] :- \mathtt{B}_\sigma[\phi_\sigma]$;
let $\mathtt{H}'_\sigma[\psi'_\sigma] = \mathtt{q}(\mathtt{D})$ by *fauré-log* evaluation ;
if $\mathtt{H}'_\sigma[\psi_\sigma]$ *is empty;*
 then halt
  else
   let $\phi'_{\mathtt{r}} = \phi_{\mathtt{r}}\{\mathtt{x}/\mathtt{y}\}, \phi'_\sigma = \phi_\sigma\{\mathtt{x}/\mathtt{y}\}$;
   if $\phi'_{\mathtt{r}} \wedge \phi'_\sigma \wedge \psi'_\sigma$ *is UNSAT* **then** halt;
   else let $\mathtt{r}'$ be $\mathtt{H_r}\{\mathtt{x}/\mathtt{y}\} :- \mathtt{B_r}\{\mathtt{x}/\mathtt{y}\}, \mathtt{H}'_\sigma, [\phi'_{\mathtt{r}}, \phi'_\sigma, \psi'_\sigma]$
    return $\mathtt{r}'$;
  end
end

systematic management of semantic constraints in the conditional tables (c-tables)

generalize the chase step

incompatible σ leads to invalid output rule (halt)

Algorithm 1: The chase with *fauré*-dependency

input : *fauré-log* rule $\mathtt{r} : \mathtt{H_r} :- \mathtt{B_r}[\phi_\mathtt{r}]$,
fauré-dependency $\sigma : \mathtt{H}_\sigma[\mathtt{x}/\mathtt{y}, \psi_\sigma] :- \mathtt{B}_\sigma[\phi_\sigma]$

output: $\mathtt{r} \rightarrow_\sigma \mathtt{r}'$

instantiate $\mathtt{B_r}[\phi_\mathtt{r}]$ into c-tables $\mathtt{D}$;
let $\mathtt{q}$ be $\mathtt{H}_\sigma[\psi_\sigma] :- \mathtt{B}_\sigma[\phi_\sigma]$;
let $\mathtt{H}'_\sigma[\psi'_\sigma] = \mathtt{q}(\mathtt{D})$ by *fauré-log* evaluation ;
if $\mathtt{H}'_\sigma[\psi_\sigma]$ *is empty;*
then halt
    else
        let $\phi'_\mathtt{r} = \phi_\mathtt{r}\{\mathtt{x}/\mathtt{y}\}, \phi'_\sigma = \phi_\sigma\{\mathtt{x}/\mathtt{y}\}$;
        if $\phi'_\mathtt{r} \wedge \phi'_\sigma \wedge \psi'_\sigma$ *is UNSAT* **then** halt;
        else let $\mathtt{r}'$ be $\mathtt{H_r}\{\mathtt{x}/\mathtt{y}\} :- \mathtt{B_r}\{\mathtt{x}/\mathtt{y}\}, \mathtt{H}'_\sigma, [\phi'_\mathtt{r}, \phi'_\sigma, \psi'_\sigma]$
        return $\mathtt{r}'$;
    end
end

(application of) incompatible policies renders an *impossible* network behavior that can*not be described any fauré-log rule*

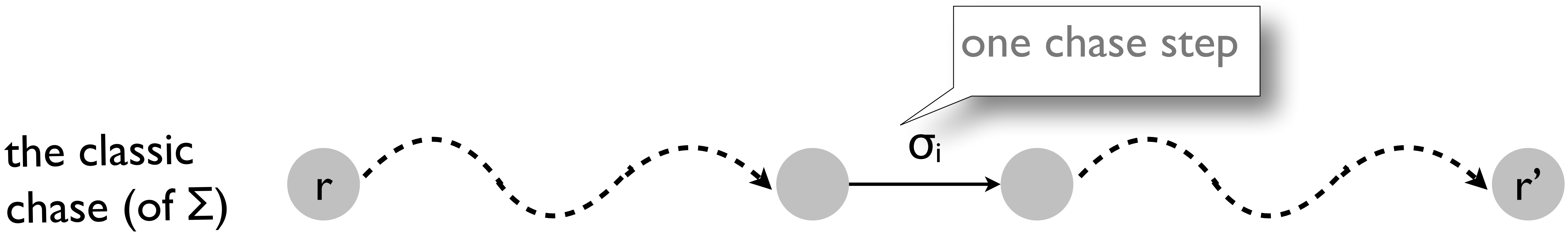
the new chase

given a set of fauré-log dependencies $\Sigma=\{\sigma_1,\ldots,\sigma_n\}, n \geqslant 1$, chase a program P with Σ by

- repeatedly chase each rule $r \in P$ with a randomly selected fauré-log dependency $\sigma \in \Sigma$

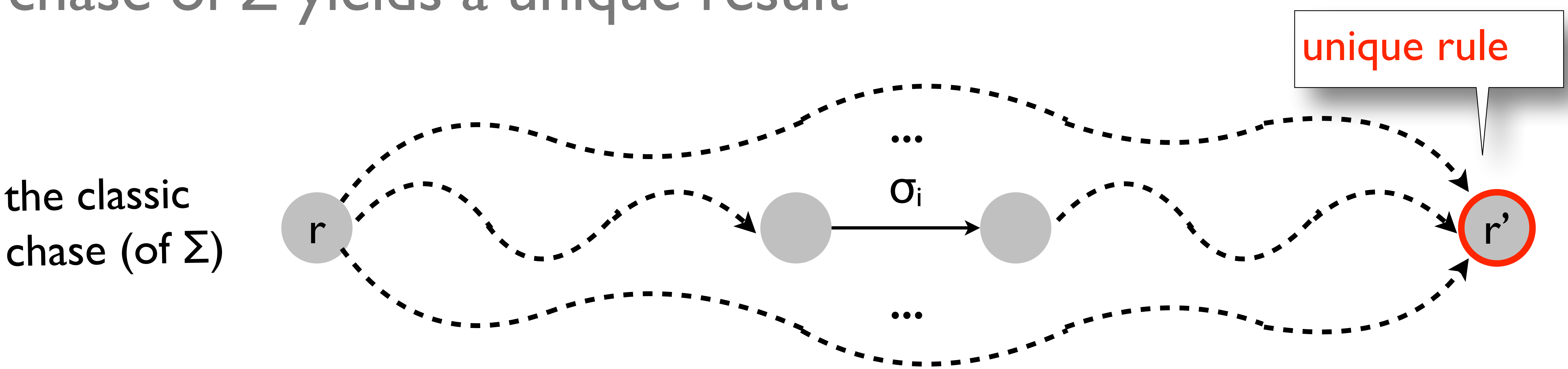
preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



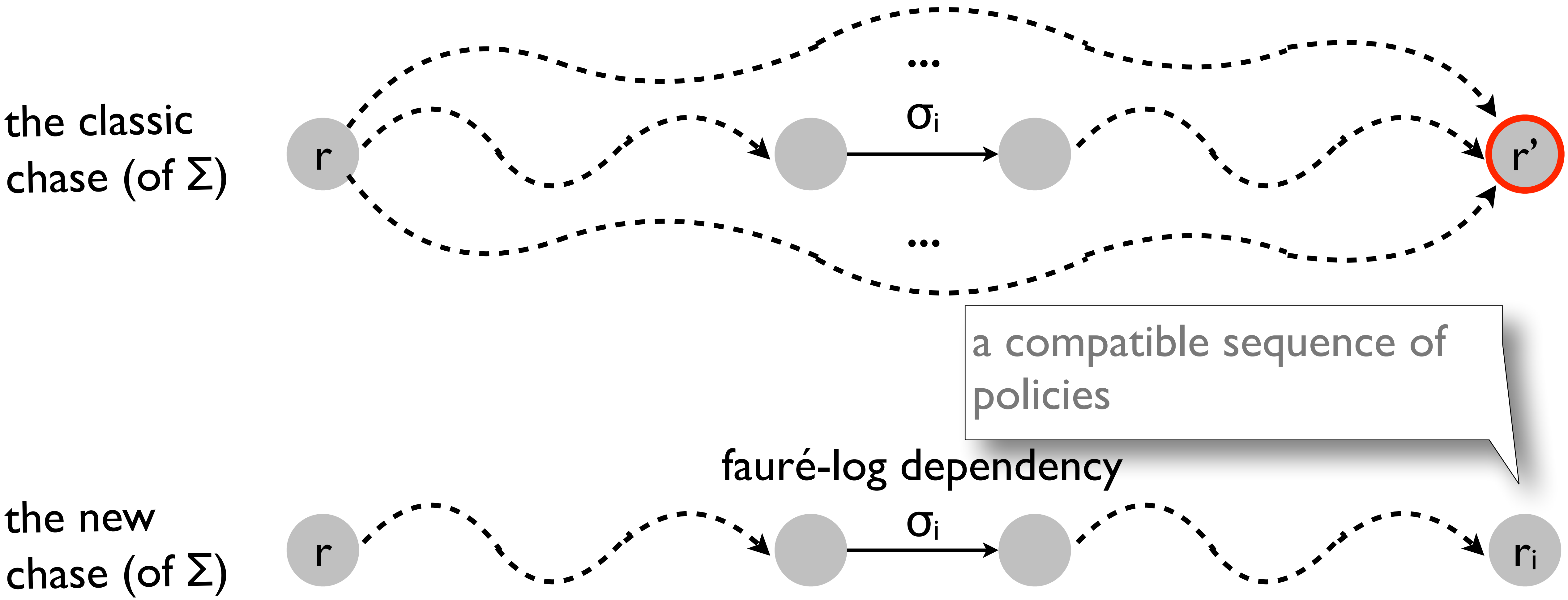
preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



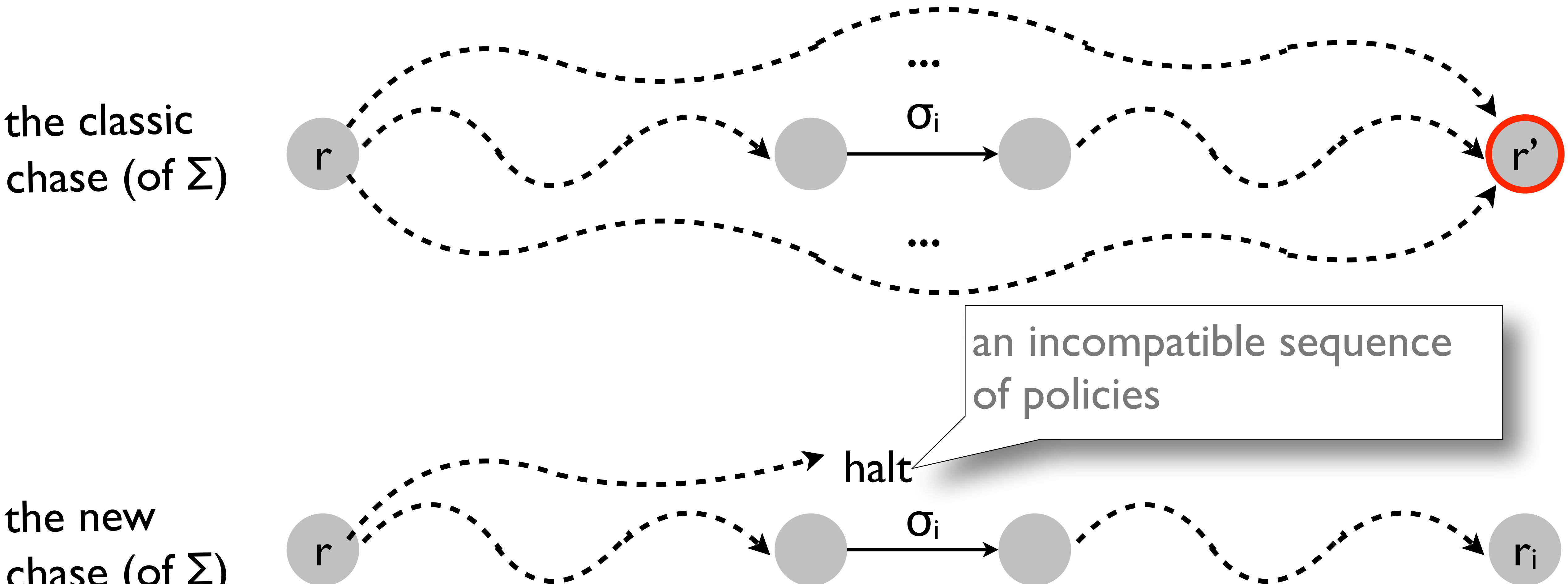
preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



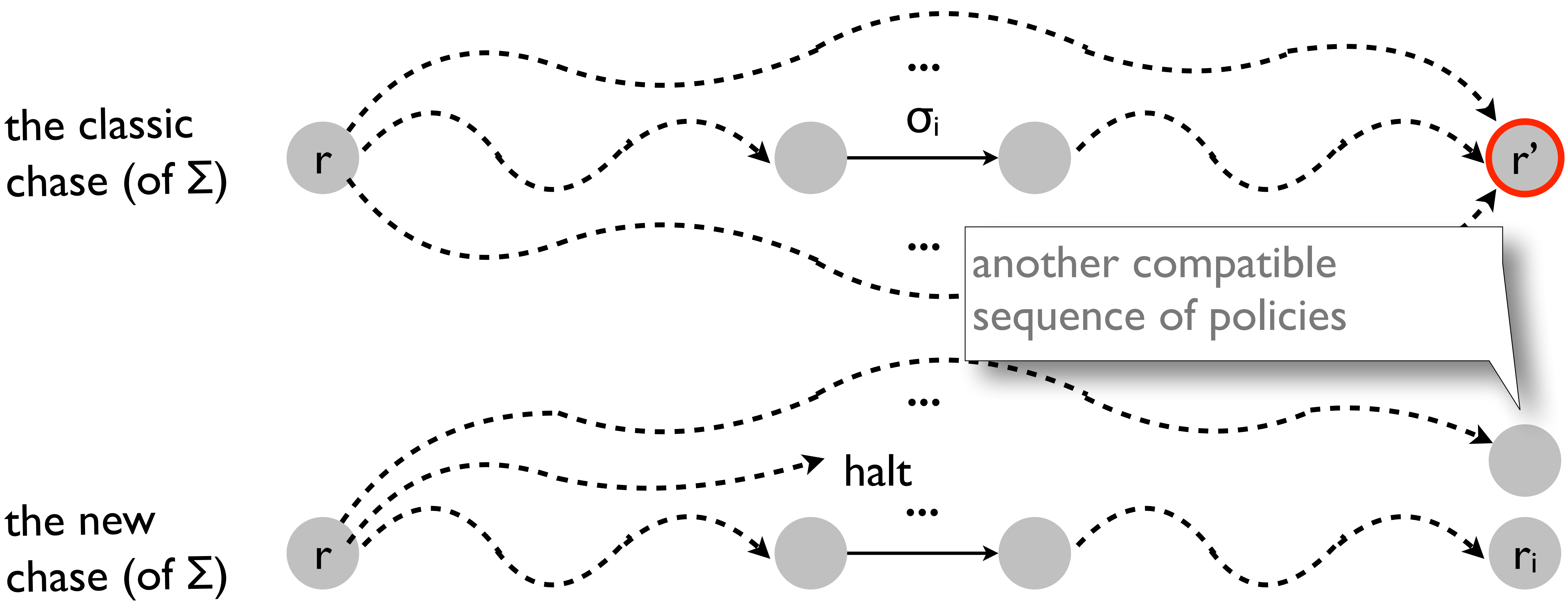
preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



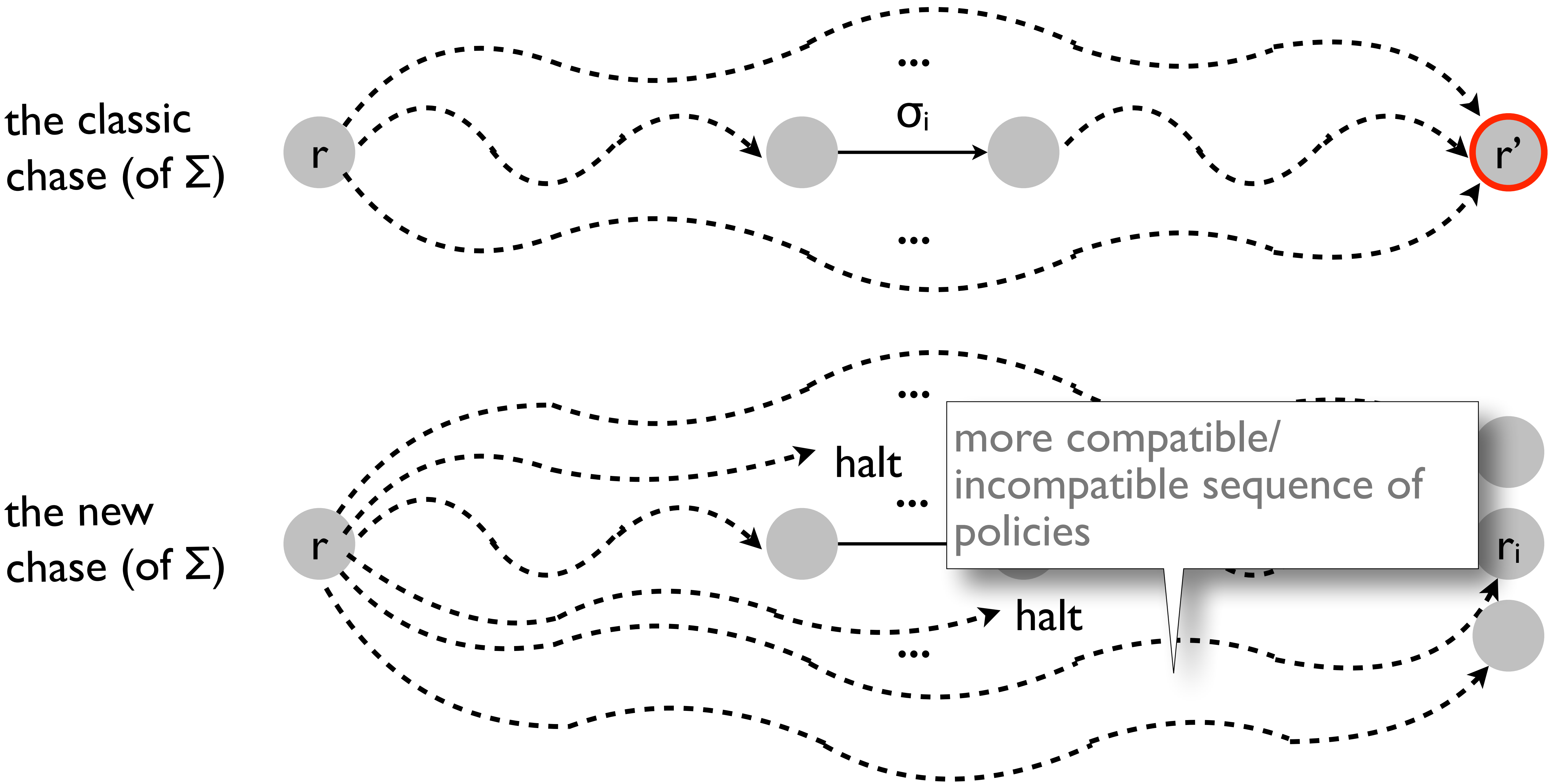
preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



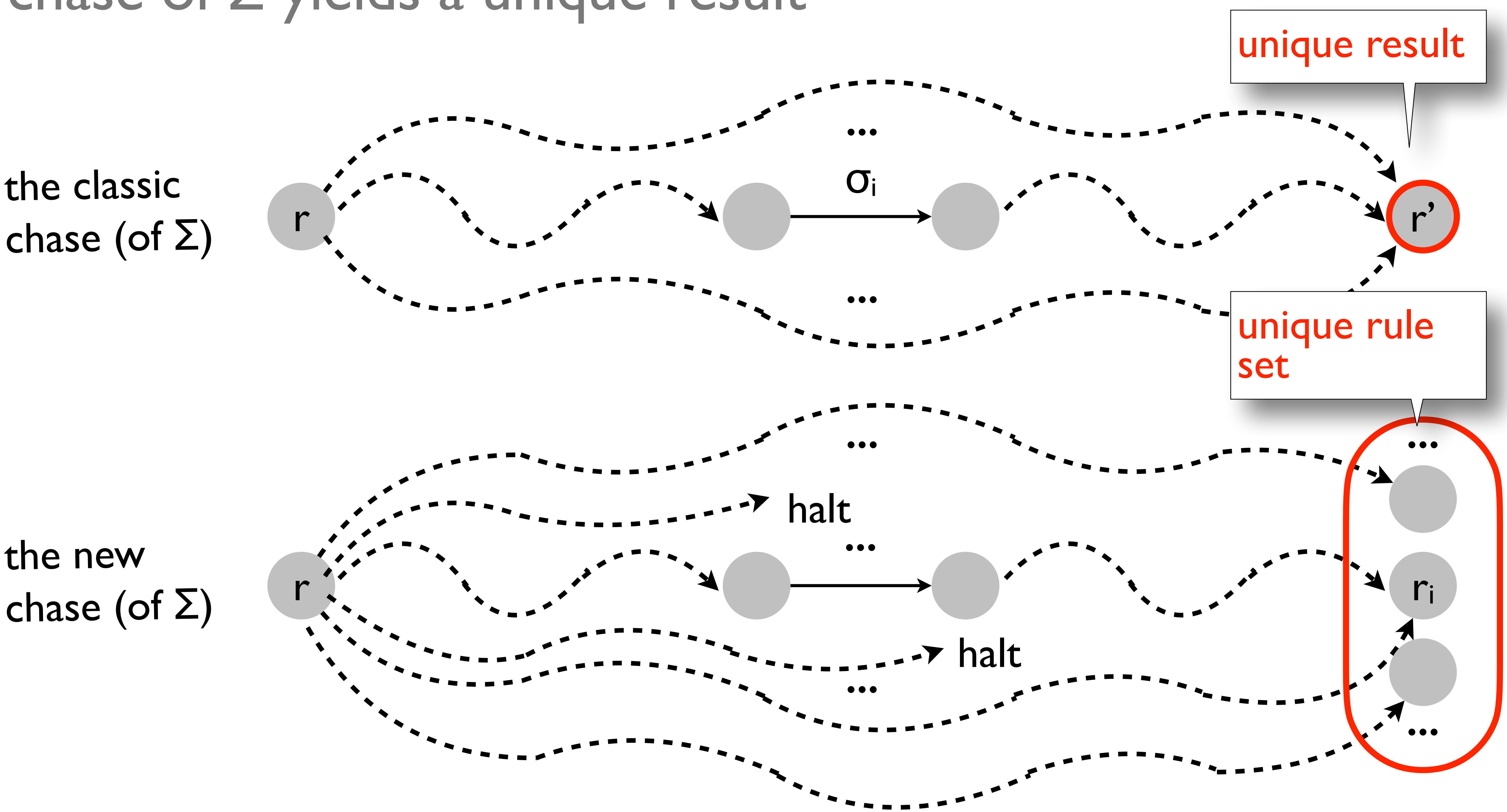
preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



preserving *Church-Rosser*

Church-Rosser: regardless of the order of applying σ (∈Σ), the chase of Σ yields a unique result



moving forward

formalization

- formalize fauré-log based chase

recursion

- extend the new chase to recursive fauré-log?

termination analysis

- non-deterministic / deterministic variants
- domain-specific notion of compatibility (of network policies)

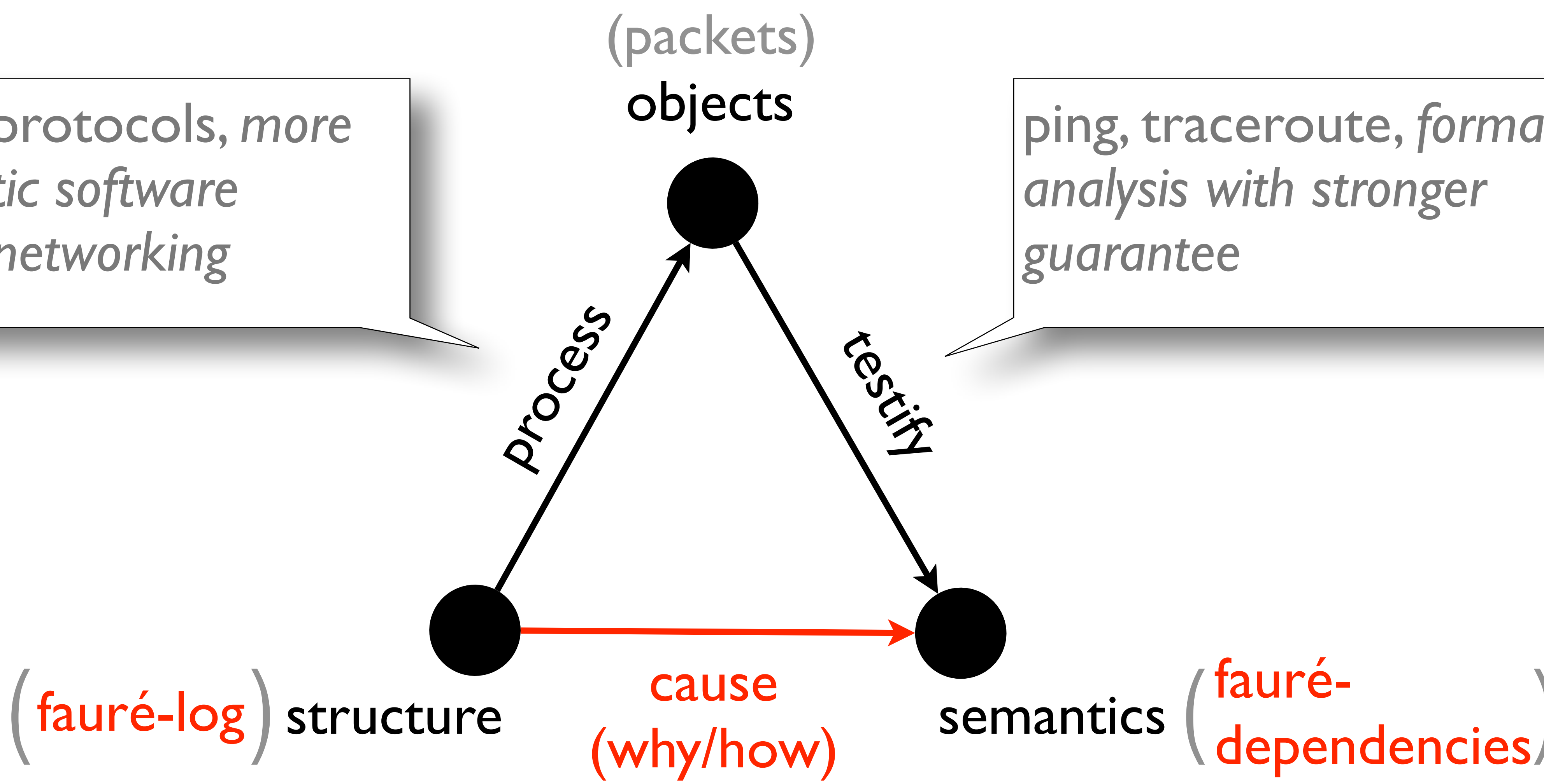
empirical study

- benchmarking performance of the new chase on network policies

conclusion

ad-hoc protocols, *more systematic software defined networking*

ping, traceroute, *formal analysis with stronger guarantee*



semantics-based network transformation with *the new chase algorithm*

- accommodates richer dependencies of network policies
- preserves *Church-Rosser*