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

source: https://en.wikipedia.org/wiki/Internet
networking: a wonderful success
innovations take rapid transitions

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.
inside the ‘Net’: a different story

network systems
- increasingly complex

network management
- a black art
software-defined networking (SDN)
formal analysis

[SIGCOMM’02] Route oscillations in I-BGP route reflection.
[TON’02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM’19] Validating Datacenters at Scale.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network configuration 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

(packets)

objects

process

testify

structure

cause (why/how)

semantics
network management, this work

- ad-hoc protocols, more systematic software defined networking

- ping, traceroute, formal analysis with stronger guarantee

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

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

*objects* (packets)

*structure* (relations, datalog)

*process* cause *(why/how)*

*testify* semantics (data dependencies)

explain how and why an intention is implemented by a soup of structures?

which fragment causes what anomaly?

what modification corrects which anomaly?

formalize network minimality — a smallest structure — for an intention

semantics-based network transformation

a procedure finding the minimal?
is the firewall effectively installed?
i.e., can hosts belong to B still send traffic to those in C?

an example

rewrite: (B,C) → (A,C)

rewrite: (A,C) → (A,D)

firewall: drop (B,D)
is the firewall effectively installed?
i.e., can hosts belong to B still send traffic to those in C?

rewrite: 
(B,C) → (A,C)

rewrite: 
(A,C) → (A,D)

firewall: 
drop (B,D)
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
is the firewall effectively installed? 

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

![Diagram of network traffic](image)

plain forwarding program $P$

structural rewrite $P'$

$a$ set of policies, characterizing legitimate packets $\Sigma$

$P'$ reflects $\Sigma$, renders the security hole
a formulation with the chase

a datalog query

\( P \)

data dependencies

\( \Sigma \)

plain forwarding program

\( P \)

a set of policies, characterizing legitimate packets

\( \Sigma \)

the chase

\( P' \)

structural rewrite

\( P' \)

\( P' = P \) on data satisfying \( \Sigma \)

\( P' \) reflects \( \Sigma \), renders the security hole
a formulation with the chase

- given a data dependency \( \sigma (\in \Sigma) \)
- eliminates “useless” evaluation in \( P \) by an intuitive structural rewrite (adding/collapsing/updating elements in the query)
/* \texttt{P}: reachability (forwarding) along R1R2R3 */
\texttt{R(x,y)} :- \texttt{F(f,x,y1,x,1)}, \texttt{F(f,x2,y2,1,2)}, \\
\texttt{F(f,x3,y3,2,3)}, \texttt{F(f,x4,y4,3,4)}, \\
\texttt{F(f,x5,y5,4,5)}, \texttt{F(f,x6,y6,5,6)}, \\
\texttt{F(f,x7,y,6,y)}.

% % permitting header modifications
% % \texttt{F(Flow, Source, Destination, Location, Next-hop)}

\texttt{k: a key dependency}
\texttt{y=y'} :- \texttt{F(f,x,y,u,w)}, \\
\texttt{F(f,x',y',u',w')}.

\textbf{tableau query}

\begin{tabular}{|c|c|c|c|c|c|}
\hline
\texttt{P} & \texttt{F} & \texttt{S} & \texttt{D} & \texttt{L} & \texttt{N} \\
\hline
\texttt{f} & \texttt{x} & \texttt{y1} & \texttt{x} & \texttt{1} & \\
\hline
\texttt{f} & \texttt{x2} & \texttt{y2} & \texttt{1} & \texttt{2} & \\
\hline
\texttt{f} & \texttt{x3} & \texttt{y3} & \texttt{2} & \texttt{3} & \\
\hline
\texttt{f} & \texttt{x4} & \texttt{y4} & \texttt{3} & \texttt{4} & \\
\hline
\texttt{f} & \texttt{x5} & \texttt{y5} & \texttt{4} & \texttt{5} & \\
\hline
\texttt{f} & \texttt{x6} & \texttt{y6} & \texttt{5} & \texttt{6} & \\
\hline
\texttt{f} & \texttt{x7} & \texttt{y} & \texttt{6} & \texttt{7} & \\
\hline
\end{tabular}
the chase

/* \textbf{P: reachability (forwarding) along R1R2R3} */
\texttt{R(x,y) :- F(f,x,y1,x,1), F(f,x2,y2,1,2),}
\texttt{ F(f,x3,y3,2,3), F(f,x4,y4,3,4),}
\texttt{ F(f,x5,y5,4,5), F(f,x6,y6,5,6),}
\texttt{ F(f,x7,y,6,y).}

%%% permitting header modifications
%%% \texttt{F(Flow, Source, Destination, Location, Next-hop)}

\texttt{k: a key dependency}
\texttt{y=y' :- F(f,x,y,u,w),}
\texttt{ F(f,x',y',u',w').}

\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
& \texttt{P} & \texttt{F} & \texttt{S} & \texttt{D} & \texttt{L} & \texttt{N} \\
\hline
\texttt{body} & \texttt{f x y1 1} & \texttt{1} & \texttt{2} & \texttt{3} & \texttt{4} & \texttt{5} & \texttt{6} & \texttt{7} \\
\hline
\texttt{head} & \texttt{x y} & \texttt{1} & \texttt{2} & \texttt{3} & \texttt{4} & \texttt{5} & \texttt{6} & \texttt{7} \\
\end{tabular}
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

<table>
<thead>
<tr>
<th>P</th>
<th>F</th>
<th>S</th>
<th>D</th>
<th>L</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>f</td>
<td>x</td>
<td>y1</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>x2</td>
<td>y1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>x3</td>
<td>y2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>x4</td>
<td>y3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>x5</td>
<td>y4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>x6</td>
<td>y5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>x7</td>
<td>y</td>
<td>6</td>
<td>y</td>
</tr>
</tbody>
</table>

body

|     | f  | x  | y1 | 3  | 4  |
|     | f  | x5 | y1 | 4  | 5  |
|     | f  | x6 | y1 | 5  | 6  |
|     | f  | x7 | y1 | 6  | 7  |

head

|     | x  | y1 |
the chase

/* P: reachability (forwarding) along R1R2R3 */
\[ R(x,y) \leftarrow F(f,x,y_1,x,1), F(f,x_2,y_2,1,2), \\
F(f,x_3,y_3,2,3), F(f,x_4,y_4,3,4), \\
F(f,x_5,y_5,4,5), F(f,x_6,y_6,5,6), \\
F(f,x_7,y,6,y). \]

%% permitting header modifications
%% F(Flow, Source, Destination, Location, Next-hop)

k: a key dependency
\[ y=y' \leftarrow F(f,x,y,u,w), \\
F(f,x',y',u',w'). \]

/* P': the result of chasing r with k */
\[ R(x,y_1) \leftarrow F(f,x,y_1,x,1), F(f,x_2,y_1,1,2), \\
F(f,x_3,y_1,2,3), F(f,x_4,y_1,3,4), \\
F(f,x_5,y_1,4,5), F(f,x_6,y_1,5,6), \\
F(f,x_7,y_1,6,y_1). \]
/* 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

/* \textbf{P}: reachability (forwarding) along R1R2R3 */
\textbf{R}(x,y) :- \textbf{F}(f,x,y1,x,1), \textbf{F}(f,x2,y2,1,2),
    \textbf{F}(f,x3,y3,2,3), \textbf{F}(f,x4,y4,3,4),
    \textbf{F}(f,x5,y5,4,5), \textbf{F}(f,x6,y6,5,6),
    \textbf{F}(f,x7,y,6,y).

% permitting header modifications
% \textbf{F}(flow, source, destination, location, next-hop)

\textbf{k}: a key dependency
y=y' :- \textbf{F}(f,x,y,u,w),
    \textbf{F}(f,x',y',u',w').

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

\textbf{k'}: k restricted to R2 and source other than 1.2.3.4
y=y' :- \textbf{F}(f,x,y,2,3),
    \textbf{F}(f,x',y',3,4),
    x\neq1.2.3.4.

?
the chase, strength

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

$\neg \phi(X,Y) \rightarrow \exists Z. \psi(Y,Z)$

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

chasing with a set $\Sigma$ is Church-Rosser

- terminates with a unique result
- the order of applying $\sigma (\in \Sigma)$ is insignificant
the chase, strength & limitation

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

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

chasing with a set $\Sigma$ is Church-Rosser

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

too limited for network policies
extend the chase to networking
dependency $\sigma (\in \Sigma)$ as general implication

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

Our contribution:
- richer dependencies (of network policies)
- retain the Church-Rosser property

chasing with a set $\Sigma$ is Church-Rosser
- terminates with a unique result
  - the order of applying $\sigma (\in \Sigma)$ is insignificant
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)

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).

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

```plaintext
/* 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)

/* 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 \( r : H_r : -B_r[\phi_r] \), 
| | fauré-dependency \( \sigma : H_{\sigma}[x/y, \psi_{\sigma}] : -B_{\sigma}[\phi_{\sigma}] \) 
| output: | \( r \rightarrow_{\sigma} r' \) 

1. instantiate \( B_r[\phi_r] \) into c-tables \( D \); 
2. let \( q = H_{\sigma}[\psi_{\sigma}] : -B_{\sigma}[\phi_{\sigma}] \); 
3. let \( H'_{\sigma}[\psi'_\sigma] = q(D) \) by fauré-log evaluation; 
4. if \( H_{\sigma}[\psi_{\sigma}] \) is empty; 
5. then halt 
6. else 
7. \[ \phi'_r = \phi_r\{x/y\}, \phi'_\sigma = \phi_\sigma\{x/y\} \] 
8. if \( \phi'_r \land \phi'_\sigma \land \psi'_\sigma \) is UNSAT then halt; 
9. else let \( r' \) be \( H_r\{x/y\} : -B_r\{x/y\}, H'_\sigma, [\phi'_r, \phi'_\sigma, \psi'_\sigma] \) 
10. return \( r' \); 
11. end
generalize the chase step

generalize the substitution of the chase to fauré-log evaluation

**Algorithm 1:** The chase with fauré-dependency

1. instantiate \( B_r[\phi_r] \) into c-tables \( D \);
2. let \( q \) be \( H_\sigma[\psi_\sigma] : -B_\sigma[\phi_\sigma] \);
3. let \( H'_\sigma[\psi'_\sigma] = q(D) \) by fauré-log evaluation;
4. if \( H'_\sigma[\psi'_\sigma] \) is empty;
   5. then halt
   6. else
      7. let \( \phi'_r = \phi_r[x/y] \), \( \phi'_\sigma = \phi_\sigma[x/y] \);
      8. if \( \phi'_r \land \phi'_\sigma \land \psi'_\sigma \) is UNSAT then halt;
      9. else let \( r' \) be \( H_r[x/y] : -B_r[x/y], H'_\sigma', [\phi'_r, \phi'_\sigma, \psi'_\sigma] \)
         return \( r' \);
10. end
11. end
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 \( r : H_r : -B_r[\phi_r] \),
fauré-dependency \( \sigma : H_\sigma[x/y, \psi_\sigma] : -B_\sigma[\phi_\sigma] \)

output: \( r \rightarrow_\sigma r' \)

1 Instantiate \( B_r[\phi_r] \) into c-tables \( D \);
2 Let \( q \) be \( H_\sigma[\psi_\sigma] : -B_\sigma[\phi_\sigma] \);
3 Let \( H'_\sigma[\psi'_\sigma] = q(D) \) by fauré-log evaluation;
4 If \( H'_\sigma[\psi'_\sigma] \) is empty;
5 Then halt
6 Else
7 Let \( \phi'_r = \phi_r[x/y], \phi'_\sigma = \phi_\sigma[x/y] \);
8 If \( \phi'_r \land \phi'_\sigma \land \psi'_\sigma \) is UNSAT then halt;
9 Else let \( r' \) be \( H_r[x/y] : -B_r[x/y], H'_\sigma, [\phi'_r, \phi'_\sigma, \psi'_\sigma] \)
   return \( r' \);
10 End
11 End

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

incompatible $\sigma$ leads to invalid output rule (halt)

Algorithm 1: The chase with fauré-dependency

| input : | fauré-log rule $r : H_r : -B_r[\phi_r],$
| | fauré-dependency $\sigma : H_\sigma[x/y, \psi_\sigma] : -B_\sigma[\phi_\sigma]$
| output: | $r \rightarrow_{\sigma} r'$

1. instantiate $B_r[\phi_r]$ into c-tables $D$;
2. let $q$ be $H_\sigma[\psi_\sigma] : -B_\sigma[\phi_\sigma]$;
3. let $H'_\sigma[\psi'_\sigma] = q(D)$ by fauré-log evaluation;
4. if $H'_\sigma[\psi_\sigma]$ is empty:
   then halt
5. else
   let $\phi'_r = \phi_r[x/y], \phi'_\sigma = \phi_\sigma[x/y]$;
   if $\phi'_r \land \phi'_\sigma \land \psi'_\sigma$ is UNSAT then halt;
9. else let $r'$ be $H_r[x/y] : -B_r[x/y], H'_r, [\phi'_r, \phi'_\sigma, \psi'_\sigma]$
   return $r'$;
11. end

(application of) incompatible policies renders an impossible network behavior that cannot be described any fauré-log rule
the new chase

given a set of fauré-log dependencies 
\( \Sigma = \{ \sigma_1, \ldots, \sigma_n \}, n \geq 1 \), chase a program \( P \) with \( \Sigma \) 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 $\sigma \ (\in \Sigma)$, the chase of $\Sigma$ yields a unique result
preserving Church-Rosser

Church-Rosser: regardless of the order of applying $\sigma \ (\in \Sigma)$, the chase of $\Sigma$ yields a unique result

the classic chase (of $\Sigma$)

unique rule
preserving Church-Rosser

Church-Rosser: regardless of the order of applying $\sigma \ (\in \Sigma)$, the chase of $\Sigma$ yields a unique result.
**preserving Church-Rosser**

*Church-Rosser*: regardless of the order of applying $\sigma$ ($\in \Sigma$), the chase of $\Sigma$ yields a unique result.
preserving Church-Rosser

Church-Rosser: regardless of the order of applying $\sigma \in \Sigma$, the chase of $\Sigma$ yields a unique result
Church-Rosser: regardless of the order of applying $\sigma \in \Sigma$, the chase of $\Sigma$ yields a unique result.
preserving Church-Rosser

Church-Rosser: regardless of the order of applying $\sigma \ (\in \Sigma)$, the chase of $\Sigma$ 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
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