Fauré: a Partial Approach to Network Analysis

Fangping Lan, Bin Gui, and Anduo Wang
Temple University
network analysis — significant progress
network analysis — significant progress

- [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
- [NSDI 15] Checking Beliefs in Dynamic Networks.
- [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 Verification

Networks need to run reliably, efficiently, and without users noticing any problems, even as they employ tools that improve the functioning of large-scale datacenter networks.
network analysis — significant progress

[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
[NSDI 20] Plankton: Scalable network configuration verification through model checking.
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[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 Verification

Half-Day Tutorial: Network Verification
Monday 17th August, Afternoon Session

Presenters
George Varghese, Microsoft Research
Nikolaj Bjorner, Microsoft Research

Tutorial location
network analysis — significant progress

[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 analysis — significant progress

- [SIGCOMM'02] Route oscillations in I-BGP route reflection.
- [HotNets'20] Solver-Aided Multi-Party Configuration.
- [TON’02] The Stable Paths Problem and Interdomain Routing.
- [SIGCOMM’19] Validating Datacenters at Scale.
- [NSDI 15] Checking Beliefs in Dynamic Networks.
- [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.
- [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 analysis

network, enterprise, private WANs, inter-domain...

query
reachability, multi-path consistency, convergence...

network analyzer

answer

guarantee, bug, …
network analysis

network

enterprise, private WANs, inter-domain…

query

reachability, multi-path consistency, convergence…

testing, simulation, model checking …

answer

guarantee, bug, …
classical formal analysis

query

definite knowledge of the network

comprehensive evaluation

decisive answer
challenges with complete analysis

classical formal analysis

definite knowledge of the network

comprehensive evaluation

decisive answer

query

control protocol

data plane at t

event at t

query

comprehensive evaluation

decisive answer
challenges with complete analysis

classical formal analysis

definite knowledge of the network

query

comprehensive evaluation

decisive answer

uncertain environment

control protocol

data plane at t

event at t

query

repeated analysis?

decisive answer
challenges with complete analysis

classical formal analysis

definite knowledge of the network

comprehensive evaluation

decisive answer

query

uncertain environment

control protocol

data plane at t

repeated analysis?

decisive answer

query

unknown

known

comprehensive evaluation

decisive answer

query

known
challenges with complete analysis

classical formal analysis

```
| definite knowledge of the network | comprehensive evaluation | decisive answer |
```

uncertain environment

```
| event at t | data plane at t | repeated analysis? | decisive answer |
```

unknown information

```
| unknown | known | stop working entirely? | decisive answer |
```
challenges with complete analysis

classical formal analysis

query

definite knowledge of the network

comprehensive evaluation

decisive answer

query

uncertain environments

?

decisive answer

query

partial information

?

decisive answer
a partial approach

classical formal analysis

query

definite knowledge of the network

comprehensive evaluation

decisive answer

loss-less modeling

query

uncertain environments

partial model

uncorrupted answer

query

partial information

?

decisive answer
classical formal analysis

definite knowledge of the network

comprensive evaluation

decisive answer

loss-less modeling

uncertain environments

partial model

uncorrupted answer

relative-complete verification

partial information

verification

correct answer

“don’t know, need more information”
loss-less modeling

query

definite knowledge of the network

comprehensive evaluation

decisive answer

query

uncertain environments

partial model

uncorrupted answer

query

partial information

verification

correct answer

“don’t know, need more information”
### Forwarding Tables

<table>
<thead>
<tr>
<th></th>
<th>node</th>
<th>node</th>
<th>node</th>
<th>node</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagram

1 → 2 → 3 → 4 → 5 → primary

---

modeling

---
modeling

<table>
<thead>
<tr>
<th>F node</th>
<th>node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 3</td>
</tr>
<tr>
<td>3 4</td>
<td>4 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F node</th>
<th>node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
<td>2 3</td>
</tr>
<tr>
<td>3 4</td>
<td>4 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F node</th>
<th>node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
<td>2 4</td>
</tr>
<tr>
<td>3 5</td>
<td>4 5</td>
</tr>
</tbody>
</table>

forwarding tables

(reachability) query

<table>
<thead>
<tr>
<th>R src dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
</tr>
<tr>
<td>1 3</td>
</tr>
<tr>
<td>1 4</td>
</tr>
<tr>
<td>1 5</td>
</tr>
<tr>
<td>2 3</td>
</tr>
<tr>
<td>2 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R src dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
</tr>
<tr>
<td>1 4</td>
</tr>
<tr>
<td>1 5</td>
</tr>
<tr>
<td>2 3</td>
</tr>
<tr>
<td>2 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R src dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
</tr>
<tr>
<td>1 5</td>
</tr>
<tr>
<td>2 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R src dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
</tr>
<tr>
<td>1 5</td>
</tr>
<tr>
<td>2 4</td>
</tr>
</tbody>
</table>

primary backup
modeling

a tuple can occur only when the condition is satisfied

1: normal
0: failed
a tuple can occur only when the condition is satisfied

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
<th>( z )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Fails: 0

Success: 1

```java
R | src dest
---|---
1  | 2  
1  | 3  
1  | 4  
1  | 5  
2  | 3  
2  | 4  
```

```java
Rep | node node
---|---
1  | 2  
1  | 3  
2  | 3  
3  | 4  
4  | 5  
```

```java
query
```
modeling

\begin{align*}
F \quad & \text{node node} \quad \text{node node} \quad \text{node node} \\
1 & 2 & 1 & 3 & 1 & 3 \\
2 & 3 & 2 & 3 & 2 & 4 \\
3 & 4 & 3 & 4 & 3 & 5 \\
4 & 5 & 4 & 5 & 4 & 5 \\
\end{align*}

\begin{align*}
R \quad & \text{src dest} \quad \text{src dest} \quad \text{src dest} \\
1 & 2 & 1 & 3 & 1 & 3 \\
1 & 3 & 1 & 4 & 1 & 5 \\
1 & 4 & 1 & 5 & 2 & 4 \\
1 & 5 & 2 & 3 & 2 & 4 \\
2 & 3 & 2 & 4 & \ldots & \ldots \\
2 & 4 & \ldots & \ldots & \ldots & \ldots \\
\end{align*}

\begin{align*}
\text{Rep} \quad & \text{node node} \\
1 & 2 & x=1 \\
1 & 3 & x=0 \\
2 & 3 & y=1 \\
2 & 4 & y=0 \\
\end{align*}

\begin{align*}
R \quad & \text{src dest} \\
1 & 2 & x=1 \\
1 & 3 & x=0 \\
1 & 5 & x=1 \land \bar{y}=1 \land \bar{z}=1 \\
1 & 5 & x=0 \land \bar{z}=1 \\
1 & 5 & x=0 \land \bar{z}=0 \\
1 & 5 & x=1 \land \bar{y}=0 \\
2 & 3 & \bar{y}=1 \\
\ldots & \ldots & \ldots \\
\end{align*}
loss-less modeling

all definite instances (regular tables)

partial representation (c-tables)

difference (between regular- and c- tables) not visible to the query
loss-less modeling

all definite instances (regular tables)

partial representation (c-tables)

query

Rep

Rep

query

difference (between regular- and c- tables) not visible to the query
loss-less modeling with SQL?

all definite instances (regular tables)

partial representation (c-tables)

SQL

Rep

extended SQL (well-known)

Rep

difference (between regular- and c- tables) not visible to the SQL
loss-less modeling with SQL?

all definite instances (regular tables)

partial representation (c-tables)

ad hoc data retrieval

static analysis
loss-less modeling with fauré-log

all definite instances (regular tables)

partial representation (c-tables)

ad hoc data retrieval

static analysis
from datalog to fauré-log

<table>
<thead>
<tr>
<th></th>
<th>datalog</th>
<th>fauré-log</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntax (rules (q))</td>
<td>(H(u) : \neg B_1(u_1), \ldots, B_n(u_n)).</td>
<td>(H(u)[(\land_{i=1}^{n}\varphi_i) \land (\land_{i=1}^{m}\psi_i)] : \neg B_1(u_1)[\varphi_1], \ldots, B_n(u_n)[\varphi_n], C_1, \ldots, C_m).</td>
</tr>
<tr>
<td>semantics</td>
<td>(q(I) = {\upsilon(u) \mid \upsilon(u_i) \in I}, I ) is a database over schema (R).</td>
<td></td>
</tr>
</tbody>
</table>

**Notions and Definitions**

<table>
<thead>
<tr>
<th>(u_i) (free tuples)</th>
<th>contains symbols in (\text{var}(q)) and (\text{dom}(R))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{dom}(R)) (attribute domain over schema (R))</td>
<td>constants</td>
</tr>
<tr>
<td>(\upsilon) (valuation)</td>
<td>(\upsilon: \text{var}(q) \rightarrow \text{dom}(R)) (i.e., ({x,y,z,\ldots}) \rightarrow constants (U {\bar{x}, \bar{y}, \bar{z}, \ldots}))</td>
</tr>
<tr>
<td>(\text{var}(q)) (variables)</td>
<td>({x,y,z,\ldots})</td>
</tr>
</tbody>
</table>
Fauré-log queries

/* reachability query */
R(f,n₁,n₂)[φ] :- F(f,n₁,n₂)[φ].
R(f,n₁,n₂)[φFrançois∩François] :- F(f,n₁,n₃)[François], R(f,n₃,n₂)[François].

recursive fauré-log
Fauré-log queries

failure patterns over $R$

$T_1(f,n_1,n_2)[\phi \land \bar{x}+\bar{y}+\bar{z}=1] : - \ R(f,n_1,n_2)[\phi], \bar{x}+\bar{y}+\bar{z}=1$. % reachability under 2-link failure

$T_2(f,2,5)[\phi \land \bar{y}=0] : - \ T_1(f,2,5)[\phi], \bar{y}=0$. % reachability between 2 and 5 under 2-link failure, one of the failure must be (2,3)

$T_3(f,1,n2)[\phi \land \bar{y}+\bar{z}<2] : - \ R(f,1,n_2)[\phi], \bar{y}+\bar{z}<2$. % reachability to 1 with at least 1-link failure

<table>
<thead>
<tr>
<th>R</th>
<th>src</th>
<th>dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$\bar{x}=1$</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>$\bar{x}=1 \land \bar{y}=1 \land \bar{z}=1$</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>$\bar{x}=0 \land \bar{z}=1$</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>$\bar{x}=0 \land \bar{z}=0$</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>$\bar{x}=1 \land \bar{y}=0$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$\bar{y}=1$</td>
</tr>
</tbody>
</table>
relative-complete verification

- Query:
  - Definite knowledge of the network
  - Uncertain environments
  - Partial knowledge

- Comprehensive evaluation:
  - Decisive answer
  - Uncorrupted answer
  - Correct answer
  - "Don't know, need more information"
relative-complete verification

query

network knowledge

verifier

answer

“don’t know” when more information is needed
relative-complete verification

network knowledge

more knowledge

verifier

query

“don’t know”

answer

verifier

“don’t know”

answer
relative-complete verification

query

network knowledge

test (i)

"don't know"

test (ii)

"don't know"

test (n)

more knowledge

...
example relative-complete verification

verification task
- invariants \( (T_1, T_2) \) continue to hold after updates

other teams
- security team maintains \( C_S \)
- TE team maintains \( C_{lb} \)
example relative-complete verification

verIFICATION TASK
- invariants \((T_1, T_2)\) continue to hold after updates

other teams
- security team maintains \(C_S\)
- TE team maintains \(C_{lb}\)
example relative-complete verification

query

subsumption

“don’t know”

rewrite + subsumption

answer

verification task
- invariants \((T_1, T_2)\) continue to hold after updates

other teams
- security team maintains \(C_S\)
- TE team maintains \(C_{lb}\)

other constraints

updates

verification task
- invariants \((T_1, T_2)\) continue to hold after updates

other teams
- security team maintains \(C_S\)
- TE team maintains \(C_{lb}\)
category (i) test: using only constraints

\[ C_S \cup C_{lb} \]

subsumption

\[ T_1, T_2 \]

"T_1 true"

"T_2 don't know"

\[ C_S \]

Mkt. \[ 80, 334, 7000 \] CS

R&D. FW

GS

constraint subsumption

packets to all the servers, must use one of the three ports 80, 334 and 7000, and must pass through a firewall

\[ T_1 \]

Mkt. FW

R&D.

GS

Mkt traffic to the critical server CS to go through a firewall
category (i) test: using only constraints

\[ C_S \cup C_{lb} \]

subsumption

\[ T_1, T_2 \]

“T_1 true”

“T_2 don’t know”

\[ C_S \]

Mkt.

R&D.

FW

GS

C_S

\[ panic : - V_S(x,y,p) \]

\[ V_S(\bar{x},\bar{y},\bar{p}) : - R(\bar{x},\bar{y},\bar{p}), \neg Fw(\bar{x},\bar{y}) \]

\[ V_S(\bar{x},\bar{y},\bar{p}) : - R(\bar{x},\bar{y},\bar{p}), \bar{p}\neq 80, \bar{p}\neq 344, \bar{p}\neq 7000 \]

\[ T_1 \]

Mkt.

R&D.

FW

GS

\[ panic : - R(Mkt,CS,\bar{p}), \neg Fw(Mkt,CS) \]

constraint as 0-ary fauré-log query (panic)
category (i) test: using only constraints

\[ C_S \cup C_{lb} \]

\[ \text{subsumption} \]

\[ T_1, T_2 \]

“T_1 true”

“T_2 don’t know”

**Cs**

- Mkt.
- R&D.
- FW
- CS
- GS

**T_1**

- Mkt.
- R&D.
- FW
- CS
- GS

**C_s**

\[ \text{panic :- } V_s(x,y,p) \]

\[ V_s(\bar{x},\bar{y},\bar{p}) :- R(\bar{x},\bar{y},\bar{p}), \neg Fw(\bar{x},\bar{y}). \]

\[ V_s(\bar{x},\bar{y},\bar{p}) :- R(\bar{x},\bar{y},\bar{p}), \bar{p} \neq 80, \bar{p} \neq 344, \bar{p} \neq 7000. \]

\[ \text{panic :- } R(\text{Mkt,CS,}\bar{p}), \neg Fw(\text{Mkt,CS}) \]
category (i) test: using only constraints

\[ C_S \cup C_{lb} \rightarrow \text{subsumption} \]

\[ \text{“T}_1 \text{ true”} \]

\[ \text{“T}_2 \text{ don’t know”} \]

\[ \text{program containment} \]

\[ \text{panic} : - V_s(x,y,p) \]

\[ V_s(\bar{x},\bar{y},\bar{p}) : - R(\bar{x},\bar{y},\bar{p}), \neg Fw(\bar{x},\bar{y}). \]

\[ V_s(\bar{x},\bar{y},\bar{p}) : - R(\bar{x},\bar{y},\bar{p}), \bar{p} \neq 80, \bar{p} \neq 344, \bar{p} \neq 7000. \]
category (i) test: using only constraints

\( C_S \cup C_{lb} \) → subsumption → “T₁ true”

“T₂ don’t know”

\( \text{panic} :- V_S(x, y, p) \)

\( V_S(\bar{x}, \bar{y}, \bar{p}) :- R(\bar{x}, \bar{y}, \bar{p}), \neg Fw(\bar{x}, \bar{y}). \)

\( V_S(\bar{x}, \bar{y}, \bar{p}) :- R(x, y, p), \quad \bar{p} \neq 80, \bar{p} \neq 344, \quad \bar{p} \neq 7000. \)

\( \text{program containment} \)

violation of \( T_1 \) implies violation of \( C_S \)
category (i) test: using only constraints

proposition [automate subsumption]
\[ \forall p_1, p_2. \ p_1 \text{ and } p_2 \text{ are two constraint programs, then } p_1 \supseteq p_2 \ (p_1 \text{ contains/implies } p_2) \text{ if applying } p_1 \ (i.e., \ fauré-log valuation) \text{ to the “instance” of } p_2 \text{ yeilds panic} \]

\[ \text{panic} :- V_s(x, y, p) \]
\[ \text{Vs}(\bar{x}, \bar{y}, \bar{p}) :- R(\bar{x}, \bar{y}, \bar{p}), \neg Fw(\bar{x}, \bar{y}) \]
\[ \text{Vs}(\bar{x}, \bar{y}, \bar{p}) :- R(x, y, p), \bar{p} \neq 80, \bar{p} \neq 344, \bar{p} \neq 7000. \]
category (ii) test: using constraints & updates

\[ C_S \cup C_{lb} \]

updates:

- Mkt.
- R&D.

\[ T_2 \]

Mkt. \hspace{1cm} R&D. \hspace{1cm} CS \hspace{1cm} GS

\[ T_2 : R&D \text{ traffic to all servers to pass through a load balancer.} \]
category (ii) test: using constraints & updates

\[
T_2 \quad \text{panic} \; :- \; R(R&D,y,7000), \neg Lb(R&D,y).
\]

updates: 
- Mkt. 
- R&D. 

\[
Cs \cup C_{lb}
\]

updates

rewrite + subsumption

“\(T_2\) don’t know”

“\(T_2\)”
category (ii) test: using constraints & updates

Cₜ S U Cₜ lb

updates

rewrite +
subsumption

“T₂ don’t know”

“T₂”

updates:

Mkt. R&D.

LB

CS GS

incorporated into T₂ by rewrites

T₂ panic :- R(R&D, y, 7000),
¬Lb(R&D, y).
category (ii) test: using constraints & updates

```
T_2
panic :- R(R&D, \bar{y}, 7000),
    ~Lb(R&D, \bar{y}).
```

```
T_2'
/* add (R&D,GS) to LB */
Lb(R&D,GS),
Lb_1(\bar{x}, \bar{y}) :- Lb(\bar{x}, \bar{y})

/* delete (Mkt,CS) LB */
Lb_2(\bar{x}, \bar{y}) :- Lb_1(\bar{x}, \bar{y})[\bar{x} \neq \text{Mkt}],
Lb_2(\bar{x}, \bar{y}) :- Lb_1(\bar{x}, \bar{y})[\bar{y} \neq \text{CS}]

/* panic after updates */
panic :- R(R&D, \bar{y}, 7000),
    ~Lb_2(R&D, \bar{y})
```
category (ii) test: using constraints & updates

\[ C_S \cup C_{lb} \]

updates:

- Mkt.
- R&D.

LB

CS

GS

update +
subsumption

"T_2 don't know"

"T_2"

\[ T_2 \]

panic :- R(R&D, \bar{y}, 7000), \neg Lb(R&D, \bar{y}). \]

\[ T_2' \]

/* add (R&D, GS) to LB */
Lb(R&D, GS).
Lb_1(\bar{x}, \bar{y}) :- Lb(\bar{x}, \bar{y})

/* delete (Mkt, CS) LB */
Lb_2(\bar{x}, \bar{y}) :- Lb_1(\bar{x}, \bar{y})[\bar{x}\neq Mkt]
Lb_2(\bar{x}, \bar{y}) :- Lb_1(\bar{x}, \bar{y})[\bar{y}\neq CS]

/* panic after updates */
panic :- R(R&D, \bar{y}, 7000), \neg Lb_2(R&D, \bar{y})

C_{lb}

panic :- Vt(x, y, p)
Vt(\bar{x}, CS, \bar{p}) :-
R(\bar{x}, CS, \bar{p}), \bar{x}\neq Mkt, \bar{x}\neq R&D.
Vt(\bar{x}, CS, \bar{p}) :-
R(\bar{x}, CS, \bar{p}), \neg Lb(\bar{x}, CS)
Vt(\bar{x}, CS, \bar{p}) :-
R(\bar{x}, CS, \bar{p}), \bar{p}\neq 7000
category (ii) test: using constraints & updates

\[ C_S \cup C_{lb} \]

updates

rewrite + subsumption

“T2 don’t know”

“T2”

updates:

Mkt.

R&D.

CS

GS

LB

T2

panic :- R(R&D,\bar{y},7000),
\neg Lb(R&D,\bar{y}).

T2'

/* add (R&D,GS) to LB */
Lb(R&D,GS).
Lb_1(\bar{x},\bar{y}) :- Lb(\bar{x},\bar{y})

/* delete (Mkt,CS) LB */
Lb_2(\bar{x},\bar{y}) :- Lb_1(\bar{x},\bar{y})[\bar{x}\#Mkt]
Lb_2(\bar{x},\bar{y}) :- Lb_1(\bar{x},\bar{y})[\bar{y}\#CS]

/* panic after updates */
panic :- R(R&D,\bar{y},7000),
\neg Lb_2(R&D,\bar{y})

C_{lb}

panic :- Vt(\bar{x},y,p)
Vt(\bar{x},CS,\bar{p}) :-
R(\bar{x},CS,\bar{p}),\bar{x}\#Mkt,\bar{x}\#R&D.
Vt(\bar{x},CS,\bar{p}) :-
R(\bar{x},CS,\bar{p}),\neg Lb(\bar{x},CS)
Vt(\bar{x},CS,\bar{p}) :-
R(\bar{x},CS,\bar{p}),\bar{p}\#7000

subsumes

C_{lb}
Updates:

- LB

Rewrite + Subsumption

"T₂ don't know"

"T₂"

Proposition:

Given a constraint $C$ and an update $U$, incorporate $U$ into $C$ by rewriting $C$ to $C'$: $C$ holds after the update $U$ iff $C'$ holds before the update.

$C_{lb}$

<table>
<thead>
<tr>
<th>panic : Vt($x, y, p$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vt($x, CS, \bar{p}$) :-</td>
</tr>
<tr>
<td>R($x, CS, \bar{p}$), $x \neq \text{Mkt}, x \neq \text{R&amp;D}$</td>
</tr>
<tr>
<td>Vt($x, CS, \bar{p}$) :-</td>
</tr>
<tr>
<td>R($x, CS, \bar{p}$), $\neg Lb$($x, CS$)</td>
</tr>
<tr>
<td>Vt($x, CS, \bar{p}$) :-</td>
</tr>
<tr>
<td>R($x, CS, \bar{p}$), $\bar{p} \neq 7000$</td>
</tr>
</tbody>
</table>

Subsumes

<table>
<thead>
<tr>
<th>Lb($R&amp;D, GS$).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lb$_1$((\bar{x}, \bar{y})) :- Lb($\bar{x}, \bar{y}$)</td>
</tr>
</tbody>
</table>

// delete (Mkt,CS) LB */
| Lb$_2$($\bar{x}, \bar{y}$):= Lb$_1$(\(\bar{x}, \bar{y}\))[\(\bar{x} \neq \text{Mkt}\]|
| Lb$_2$($\bar{x}, \bar{y}$) :- Lb$_1$(\(\bar{x}, \bar{y}\))[\(\bar{y} \neq \text{CS}\]|

//panic after updates */
| panic :- R($R&D, \bar{y}$, 7000), $\neg Lb$_2$(R&D, $\bar{y}$) |

Category (ii) test: using constraints & updates
practical implementation in SQL

- shallow embedding of fauré-log in PostgreSQL + Z3

evaluation

- realistic topology (inferred from BGP announcements)
- synthetic link failures
- representative queries
  - $q_4$-$q_5$ (all pair-wise reachability), $q_6$-$q_8$ (various failure patterns)

<table>
<thead>
<tr>
<th>#prefix</th>
<th>$q_4 - q_5$</th>
<th>$q_6$</th>
<th>$q_7$</th>
<th>$q_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sql</td>
<td>sql</td>
<td>sql</td>
<td>sql</td>
</tr>
<tr>
<td></td>
<td>#tuples</td>
<td>#tuples</td>
<td>#tuples</td>
<td>#tuples</td>
</tr>
<tr>
<td>1000</td>
<td>0.625s</td>
<td>0.85s(0.11%)</td>
<td>796.35s</td>
<td>0.08s(22.86%)</td>
</tr>
<tr>
<td>10000</td>
<td>5.75s</td>
<td>8.96s</td>
<td>-</td>
<td>0.27s</td>
</tr>
<tr>
<td>100000</td>
<td>54.85s</td>
<td>113.48s</td>
<td>-</td>
<td>1.66s(6.18%)</td>
</tr>
<tr>
<td>922067</td>
<td>816.4s</td>
<td>4169.02s</td>
<td>-</td>
<td>11.1s(3.71%)</td>
</tr>
</tbody>
</table>

Running time (seconds) of reachability analysis on four rib inputs: '-' means over 2 hours.

- $q_4$-$q_5$: 1000 (all pair-wise reachability)
- $q_6$: 10000 (all pair-wise reachability)
- $q_7$: 100000 (all pair-wise reachability)
- $q_8$: 922067 (all pair-wise reachability)

Table 4: Preliminary results for reachability analysis on four rib inputs.
recap — partial analysis

classical network analysis

query

definite knowledge
of the network

comprehensive
evaluation

decisive answer

a departure from the
complete approach

query

uncertain environments

loss-less modeling

uncorrupted answer

query

partial knowledge

relative-complete
verification

correct answer

“don’t know, need more information”
recap — realization

classical network analysis

query

definite knowledge of the network

comprehensive evaluation

decisive answer

Fauré

Faure-log

uncertain environments as c-tables

loss-less modeling

uncorrupted answer

Faure-log

partial knowledge as fauré-log

relative-complete verification

correct answer

“don’t know, need more information”
recap — realization

classical network analysis

query

definite knowledge of the network

comprehensive evaluation

decisive answer

Fauré

uncertain environments as c-tables

loss-less modeling

uncorrupted answer

Fauré-log

query c-tables

Fauré-log

partial knowledge as fauré-log

relative-complete verification

correct answer

“don’t know, need more information”
recap — realization

classical network analysis

query

definite knowledge of the network

comprehensive evaluation

decisive answer

Fauré

uncertain environments as c-tables

loss-less modeling

uncorrupted answer

Fauré-log

query c-tables

relative-complete verification

correct answer

Fauré-log

simplify static analysis

partial knowledge as fauré-log

“don’t know, need more information”
thank you

https://github.com/ravel-net/Faure