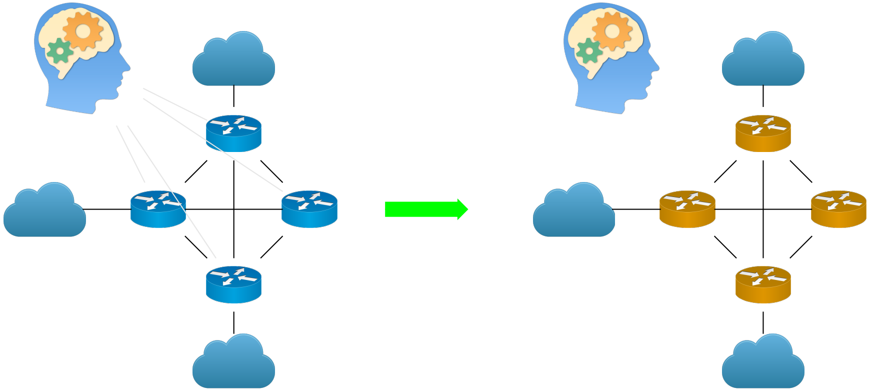
Optimizing Horn Solvers for Network Repair

Hossein Hojjat[1,4] Philipp Rümmer [2]
Jedidiah McClurg[3] Pavol Černý[3] Nate Foster [1]

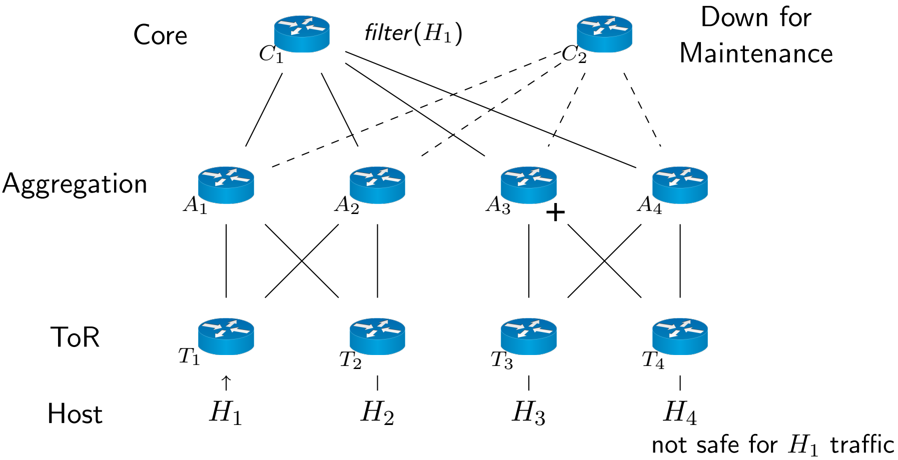
[1]Cornell University, [2]Uppsala University, [3]University of Colorado Boulder,
[4]Rochester Institute of Technology

16th International Conference on Formal Methods in Computer Aided Design
October 6th, 2016

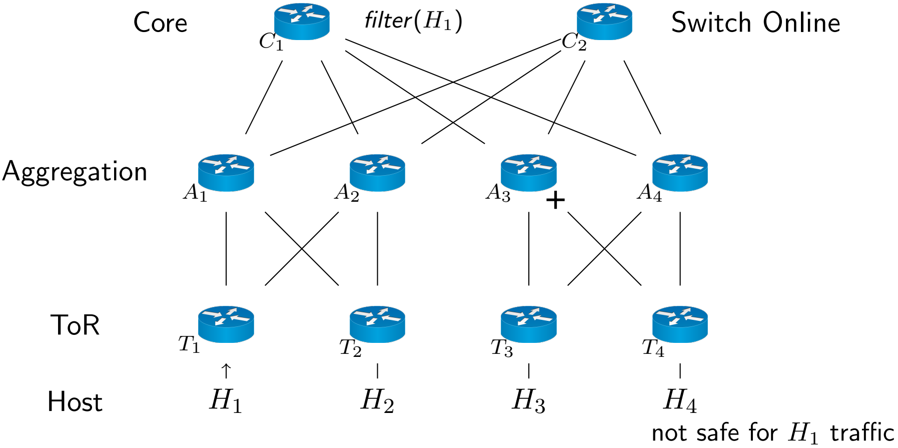
Software-Defined Networking (SDN)

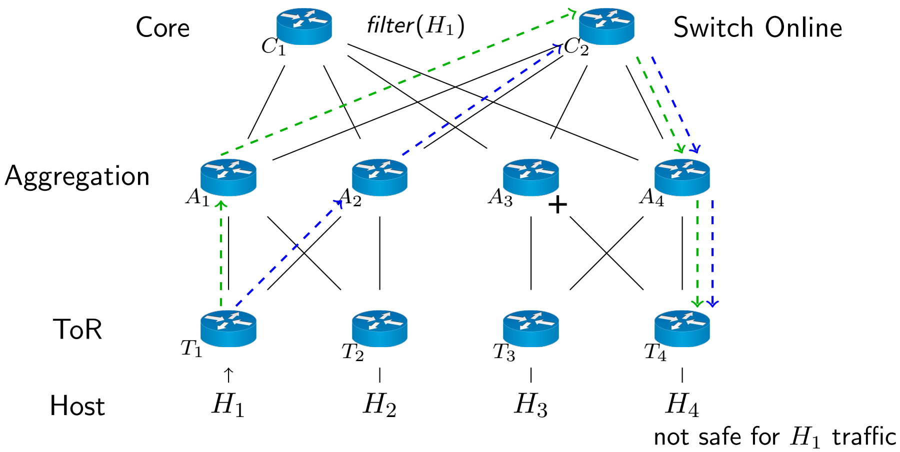


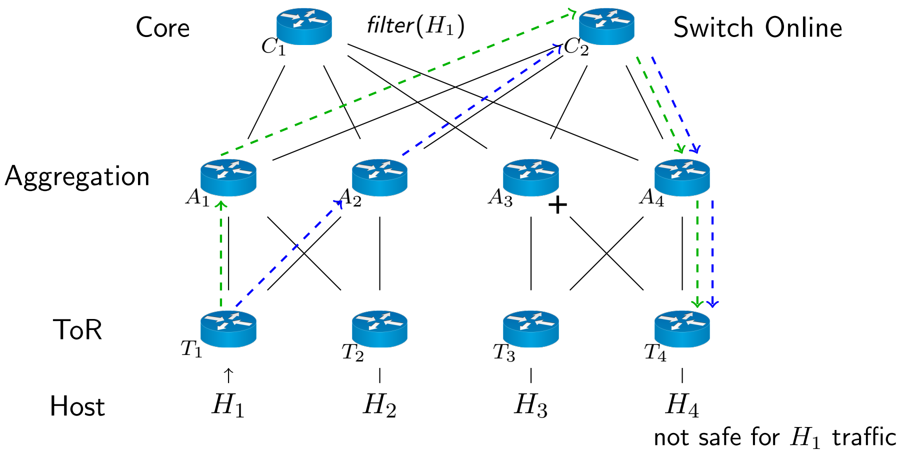
- Software-Defined Networking (SDN): emerging network architecture
- SDN Controllers are the **brains** of network
  - Determine how the switches and routers should handle network traffic
  - Can update the forwarding tables of switches



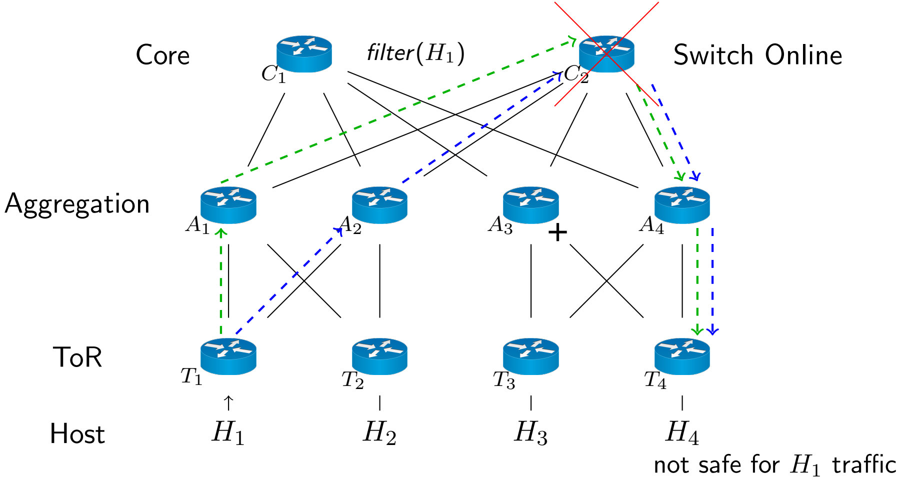
Core
filter(H_1)
Down for Maintenance
C_1
C_2
Aggregation
A_1
A_2
A_3
A_4
ToR
T_1
T_2
T_3
T_4
Host
H_1
H_2
H_3
H_4
not safe for H_1 traffic


Core
C_1
filter(H_1)
C_2
Switch Online
Aggregation
A_1
A_2
A_3
+
A_4
ToR
T_1
T_2
T_3
T_4
Host
H_1
H_2
H_3
H_4
not safe for H_1 traffic

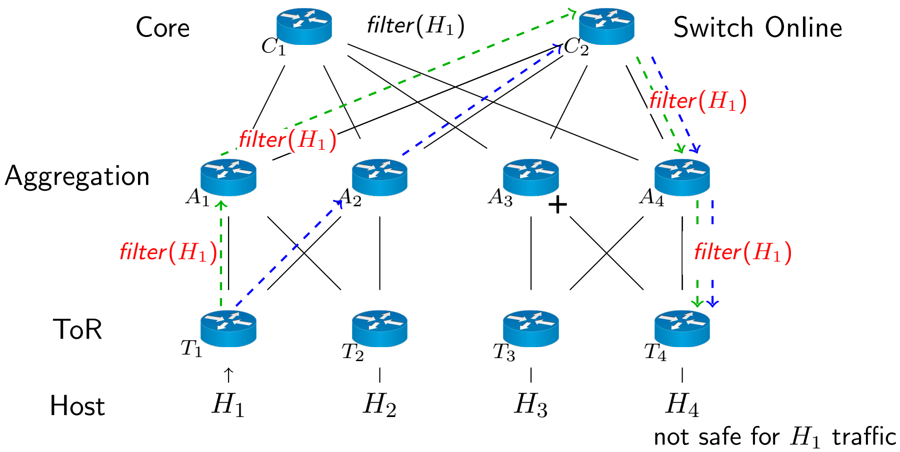

Core
filter(H_1)
Switch Online
C_1
C_2
Aggregation
A_1
A_2
A_3
A_4
+
ToR
T_1
T_2
T_3
T_4
Host
H_1
H_2
H_3
H_4
not safe for H_1 traffic



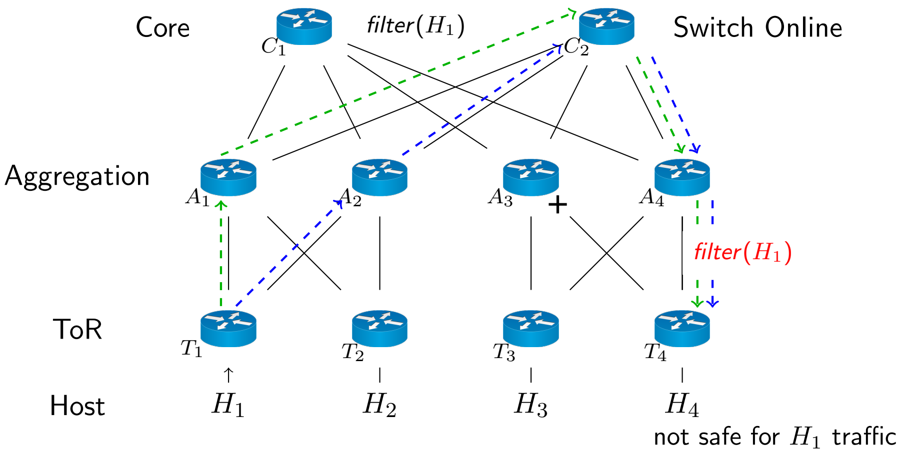
- How can we return back to safety by adding filters on links?
- There are several possible repair solutions
- Interested in **best** solutions:
  - e.g. the ones that touch minimal number of switches
  - and maintain connectivity



- How can we return back to safety by adding filters on links?
- There are several possible repair solutions
- Interested in **best** solutions:
  - e.g. the ones that touch minimal number of switches
  - and maintain connectivity



- How can we return back to safety by adding filters on links?
- There are several possible repair solutions
- Interested in **best** solutions:
  - e.g. the ones that touch minimal number of switches
  - and maintain connectivity

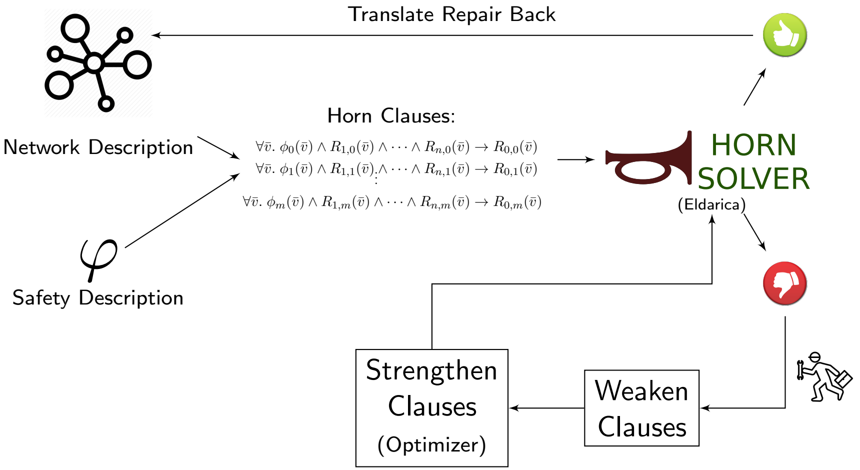


- How can we return back to safety by adding filters on links?
- There are several possible repair solutions
- Interested in **best** solutions:
  - e.g. the ones that touch minimal number of switches
  - and maintain connectivity

Contributions

1. Translation of network and its correctness conditions to Horn clauses
2. Repair unsatisfiable Horn clauses (i.e. buggy system violating correctness)
3. New lattice-based optimization procedure for Horn clause repair

Repair Framework



Our Repair Approach

$$\begin{array}{ll} \forall \bar{v}. & \psi_0(\bar{v}) \land \boldsymbol{R}_{1,0}(\bar{v}) \land \cdots \land \boldsymbol{R}_{n,0}(\bar{v}) \rightarrow \boldsymbol{R}_{0,0}(\bar{v}) \\ \forall \bar{v}. & \psi_1(\bar{v}) \land \boldsymbol{R}_{1,1}(\bar{v}) \land \cdots \land \boldsymbol{R}_{n,1}(\bar{v}) \rightarrow \boldsymbol{R}_{0,1}(\bar{v}) \\ & \vdots \\ \forall \bar{v}. & \psi_m(\bar{v}) \land \boldsymbol{R}_{1,m}(\bar{v}) \land \cdots \land \boldsymbol{R}_{n,m}(\bar{v}) \rightarrow \boldsymbol{R}_{0,m}(\bar{v}) \\ \forall \bar{v}. & \phi_{m'}(\bar{v}) \land \boldsymbol{R}_{1,m'}(\bar{v}) \land \cdots \land \boldsymbol{R}_{n,m'}(\bar{v}) \rightarrow \mathit{false} \end{array} \models \mathit{false}$$

Our Repair Approach

$$\begin{array}{l}\forall \bar{v}.\ \boldsymbol{R^*}_0(\bar{v}) \ \wedge\ \psi_0(\bar{v}) \wedge \boldsymbol{R}_{1,0}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,0}(\bar{v}) \rightarrow \boldsymbol{R}_{0,0}(\bar{v}) \\ \forall \bar{v}.\ \boldsymbol{R^*}_1(\bar{v}) \ \wedge\ \psi_1(\bar{v}) \wedge \boldsymbol{R}_{1,1}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,1}(\bar{v}) \rightarrow \boldsymbol{R}_{0,1}(\bar{v}) \\ \qquad\qquad \vdots \\ \forall \bar{v}.\ \boldsymbol{R^*}_m(\bar{v}) \ \wedge\ \psi_m(\bar{v}) \wedge \boldsymbol{R}_{1,m}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,m}(\bar{v}) \rightarrow \boldsymbol{R}_{0,m}(\bar{v}) \\ \forall \bar{v}.\ \boldsymbol{R^*}_{m'}(\bar{v}) \wedge\ \phi_{m'}(\bar{v}) \wedge \boldsymbol{R}_{1,m'}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,m'}(\bar{v}) \rightarrow \mathit{false} \end{array} \models \mathit{false}$$

Weaken

- Conjoin fresh relation symbols $\boldsymbol{R}_i^*$ to the bodies of Horn clauses

Our Repair Approach

$$\forall \bar{v}.\ \boldsymbol{R^*}_0(\bar{v}) \wedge \psi_0(\bar{v}) \wedge \boldsymbol{R}_{1,0}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,0}(\bar{v}) \to \boldsymbol{R}_{0,0}(\bar{v})$$
$$\forall \bar{v}.\ \boldsymbol{R^*}_1(\bar{v}) \wedge \psi_1(\bar{v}) \wedge \boldsymbol{R}_{1,1}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,1}(\bar{v}) \to \boldsymbol{R}_{0,1}(\bar{v})$$
$$\vdots$$
$$\forall \bar{v}.\ \boldsymbol{R^*}_m(\bar{v}) \wedge \psi_m(\bar{v}) \wedge \boldsymbol{R}_{1,m}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,m}(\bar{v}) \to \boldsymbol{R}_{0,m}(\bar{v})$$
$$\forall \bar{v}.\ \boldsymbol{R^*}_{m'}(\bar{v}) \wedge \phi_{m'}(\bar{v}) \wedge \boldsymbol{R}_{1,m'}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,m'}(\bar{v}) \to \mathit{false}$$

$$\not\models \mathit{false}$$

Weaken

- Conjoin fresh relation symbols $\boldsymbol{R}_i^*$ to the bodies of Horn clauses
- Weaker system is satisfiable, may have undesirable solutions
- Any of the new relation symbols can be false
  - (effectively removing the clause)

Our Repair Approach

$$\begin{array}{l}
\forall \bar{v}.\ \boldsymbol{R^*}_0(\bar{v}) \ \wedge\ \psi_0(\bar{v}) \wedge \boldsymbol{R}_{1,0}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,0}(\bar{v}) \rightarrow \boldsymbol{R}_{0,0}(\bar{v}) \\
\forall \bar{v}.\ \boldsymbol{R^*}_1(\bar{v}) \ \wedge\ \psi_1(\bar{v}) \wedge \boldsymbol{R}_{1,1}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,1}(\bar{v}) \rightarrow \boldsymbol{R}_{0,1}(\bar{v}) \\
\quad \vdots \\
\forall \bar{v}.\ \boldsymbol{R^*}_m(\bar{v}) \ \wedge\ \psi_m(\bar{v}) \wedge \boldsymbol{R}_{1,m}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,m}(\bar{v}) \rightarrow \boldsymbol{R}_{0,m}(\bar{v}) \\
\forall \bar{v}.\ \boldsymbol{R^*}_{m'}(\bar{v}) \wedge\ \phi_{m'}(\bar{v}) \wedge \boldsymbol{R}_{1,m'}(\bar{v}) \wedge \cdots \wedge \boldsymbol{R}_{n,m'}(\bar{v}) \rightarrow \mathit{false}
\end{array} \quad \not\models \mathit{false}$$

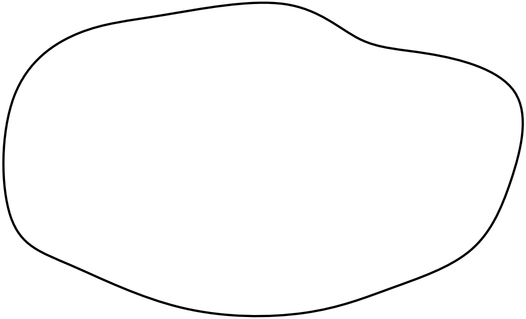
Weaken

- Conjoin fresh relation symbols $\boldsymbol{R}_i^*$ to the bodies of Horn clauses
- Weaker system is satisfiable, may have undesirable solutions
- Any of the new relation symbols can be false
  - (effectively removing the clause)

Strengthen

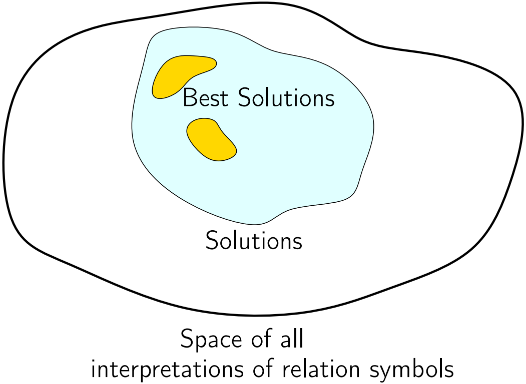
- Add more constraints to rule out undesirable solutions
- User can select the "best" repairs (e.g. reject false solutions,if possible)

Goal: find solutions for set of Horn clauses subject to objective function

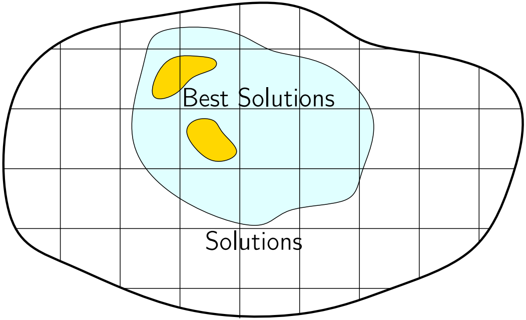


Space of all
interpretations of relation symbols

Goal: find solutions for set of Horn clauses subject to objective function

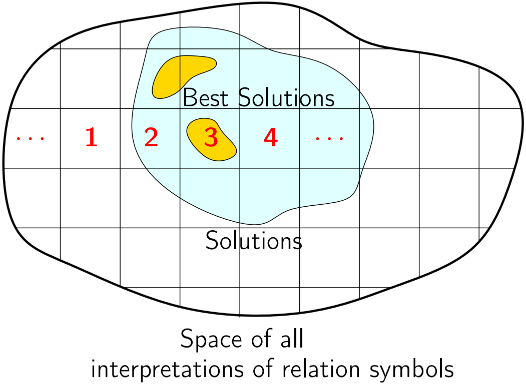


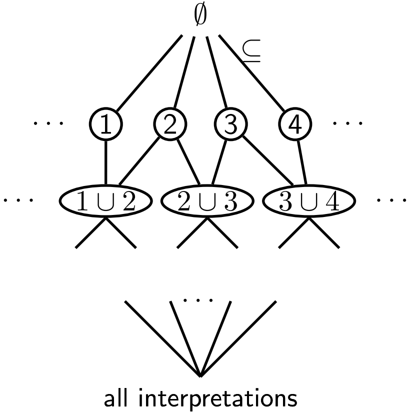
Goal: find solutions for set of Horn clauses subject to objective function



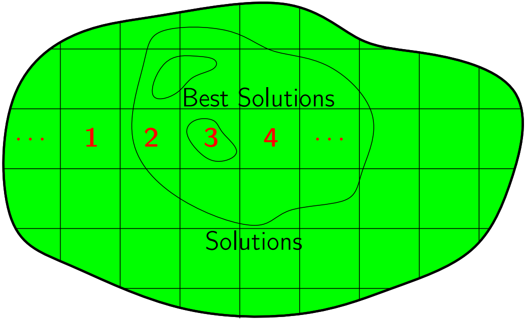
Space of all
interpretations of relation symbols

Goal: find solutions for set of Horn clauses subject to objective function

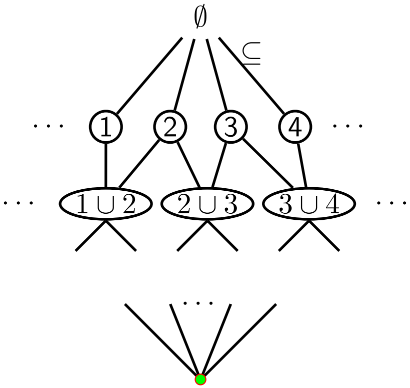




Goal: find solutions for set of Horn clauses subject to objective function

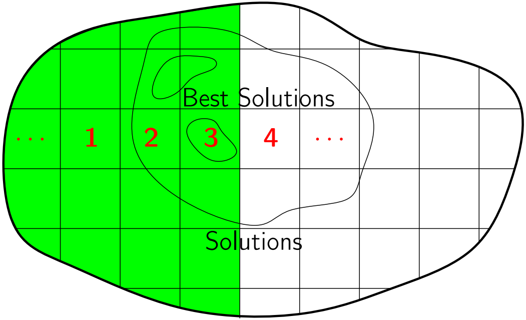


Space of all
interpretations of relation symbols

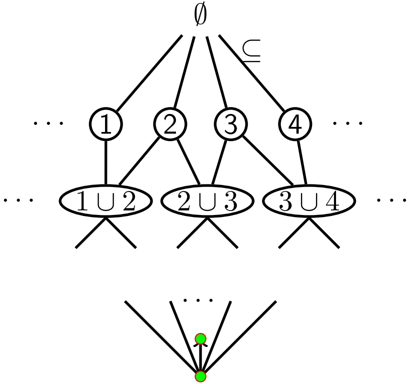


all interpretations

Goal: find solutions for set of Horn clauses subject to objective function

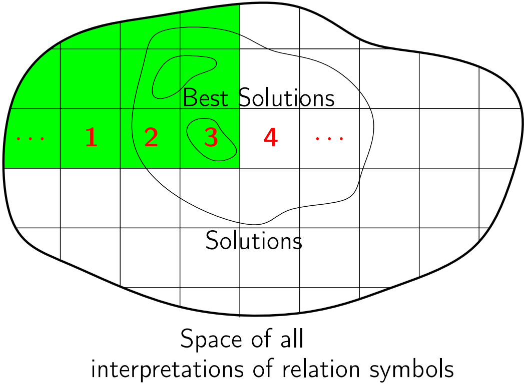


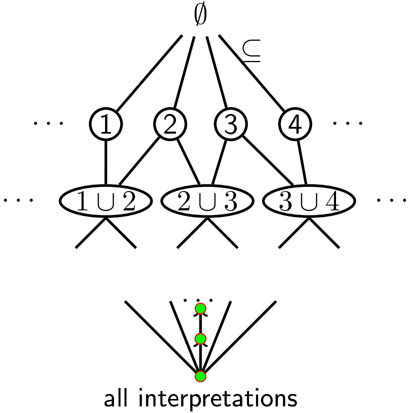
Space of all
interpretations of relation symbols



all interpretations

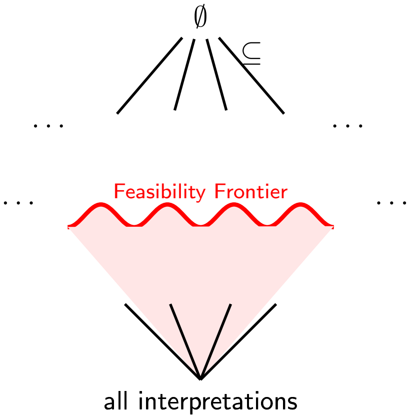
Goal: find solutions for set of Horn clauses subject to objective function





Goal: find solutions for set of Horn clauses subject to objective function

Objective function:
Rank nodes of lattice monotonically



Goal: find solutions for set of Horn clauses subject to objective function

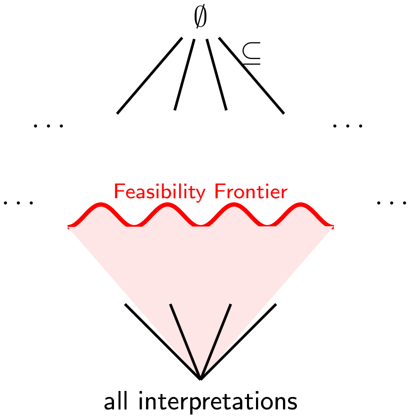
Objective function:

Rank nodes of lattice monotonically

Search Algorithm:

Walk smartly in the lattice to find the **best** solution:

- inside the feasibility cone
- has maximum ranking



Goal: find solutions for set of Horn clauses subject to objective function

Objective function:

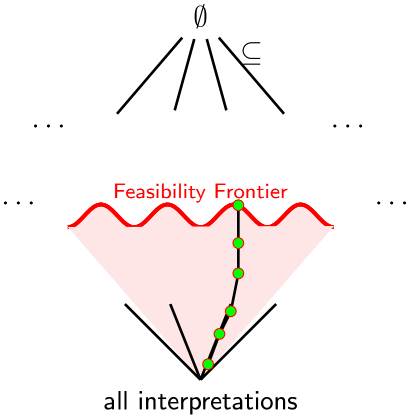
Rank nodes of lattice monotonically

Search Algorithm:

Walk smartly in the lattice to find the **best** solution:

- inside the feasibility cone
- has maximum ranking

1. Pick a feasible node and walk until reach frontier



Goal: find solutions for set of Horn clauses subject to objective function

Objective function:

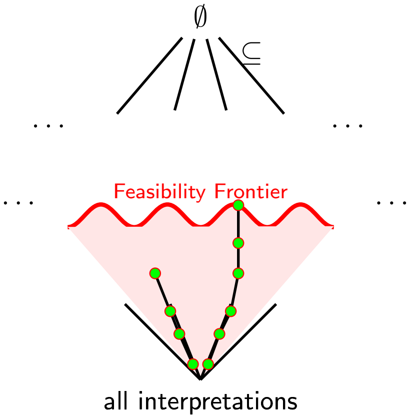
Rank nodes of lattice monotonically

Search Algorithm:

Walk smartly in the lattice to find the **best** solution:

- inside the feasibility cone
- has maximum ranking

1. Pick a feasible node and walk until reach frontier
2. Pick a lower rank incomparable node and walk again



Goal: find solutions for set of Horn clauses subject to objective function

Objective function:

Rank nodes of lattice monotonically

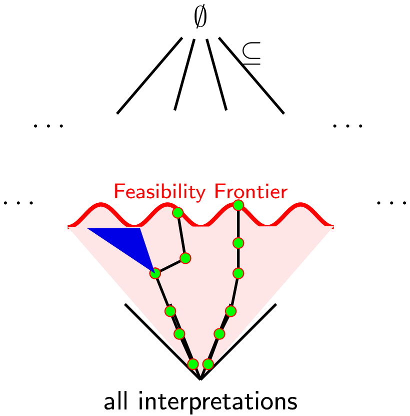
Search Algorithm:

Walk smartly in the lattice to find the **best** solution:

- inside the feasibility cone
- has maximum ranking

1. Pick a feasible node and walk until reach frontier
2. Pick a lower rank incomparable node and walk again

- Use feasibility bounds as heuristic to prune search



Goal: find solutions for set of Horn clauses subject to objective function

Objective function:

Rank nodes of lattice monotonically

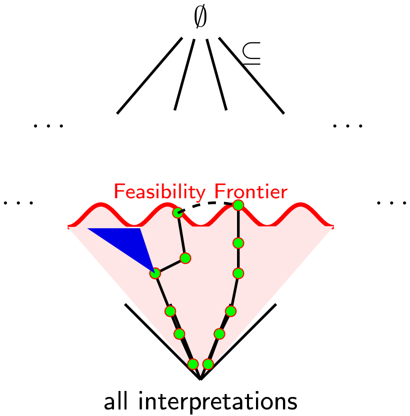
Search Algorithm:

Walk smartly in the lattice to find the **best** solution:

- inside the feasibility cone
- has maximum ranking

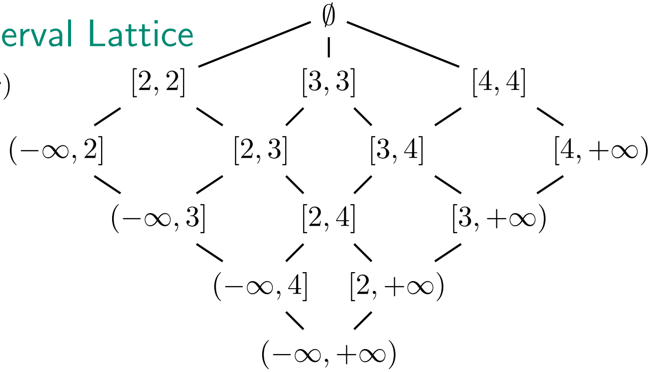
1. Pick a feasible node and walk until reach frontier
2. Pick a lower rank incomparable node and walk again

- Use feasibility bounds as heuristic to prune search



Example: Interval Lattice

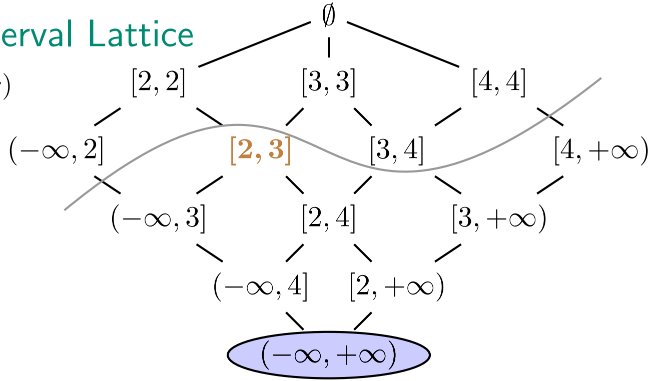
Interval lattice $f(x)$ for $\{2,\ 4\}$



- Interval lattices are useful to filter out a range of packets

Example: Interval Lattice

Interval lattice $f(x)$ for $\{2,\ 4\}$

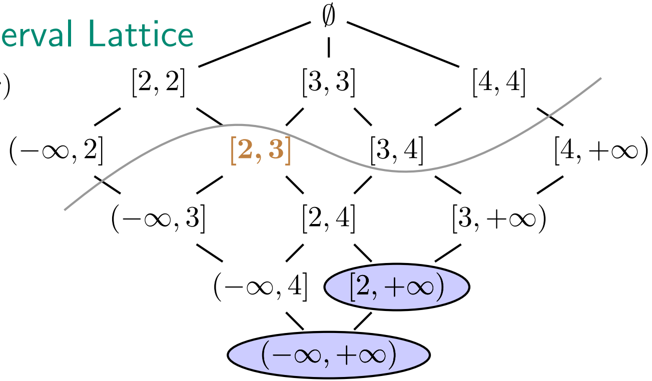


- Interval lattices are useful to filter out a range of packets
- Example: TTL scoping (for network details see paper)

$$obj(I) = \begin{cases} 1 & \text{if } I = [x, y] \text{ or } I = (-\infty, y] \\ -\infty & \text{if } I = [x, \infty) \text{ or } I = (-\infty, \infty) \\ \infty & \text{if } I = \emptyset \end{cases}$$

Example: Interval Lattice

Interval lattice $f(x)$ for $\{2, 4\}$

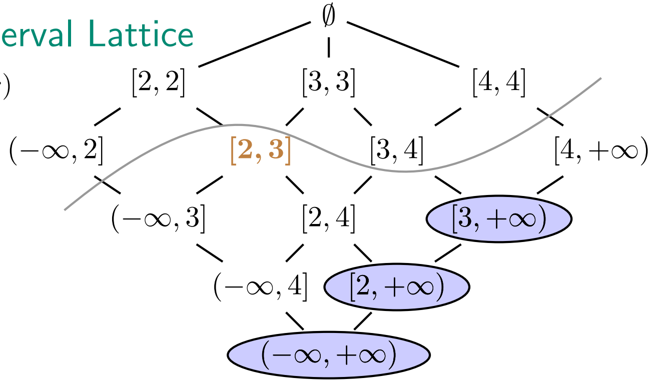


- Interval lattices are useful to filter out a range of packets
- Example: TTL scoping (for network details see paper)

$$obj(I) = \begin{cases} 1 & \text{if } I = [x, y] \text{ or } I = (-\infty, y] \\ -\infty & \text{if } I = [x, \infty) \text{ or } I = (-\infty, \infty) \\ \infty & \text{if } I = \emptyset \end{cases}$$

Example: Interval Lattice

Interval lattice $f(x)$ for $\{2,\ 4\}$

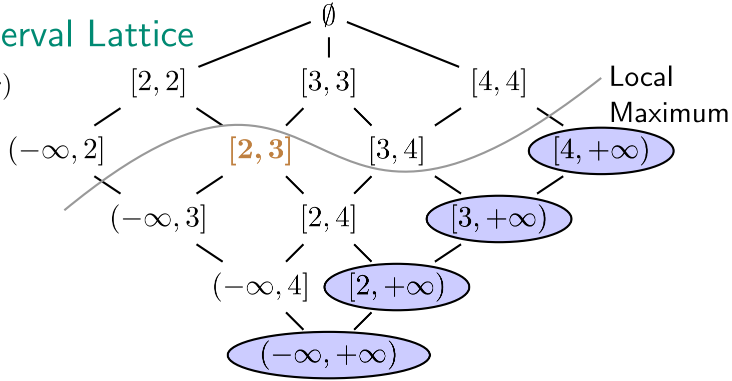


- Interval lattices are useful to filter out a range of packets
- Example: TTL scoping (for network details see paper)

$$obj(I) = \begin{cases} 1 & \text{if } I = [x,y] \text{ or } I = (-\infty, y] \\ -\infty & \text{if } I = [x,\infty) \text{ or } I = (-\infty,\infty) \\ \infty & \text{if } I = \emptyset \end{cases}$$

Example: Interval Lattice

Interval lattice $f(x)$ for $\{2,\ 4\}$

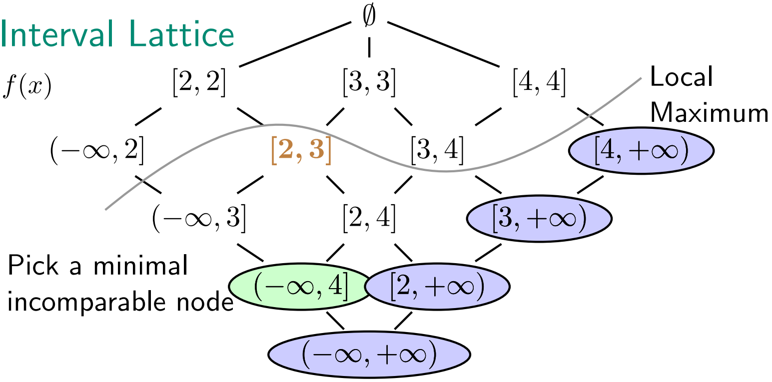


- Interval lattices are useful to filter out a range of packets
- Example: TTL scoping (for network details see paper)

$$obj(I) = \begin{cases} 1 & \text{if } I = [x,y] \text{ or } I = (-\infty, y] \\ -\infty & \text{if } I = [x,\infty) \text{ or } I = (-\infty,\infty) \\ \infty & \text{if } I = \emptyset \end{cases}$$

Example: Interval Lattice

Interval lattice $f(x)$ for $\{2,\ 4\}$

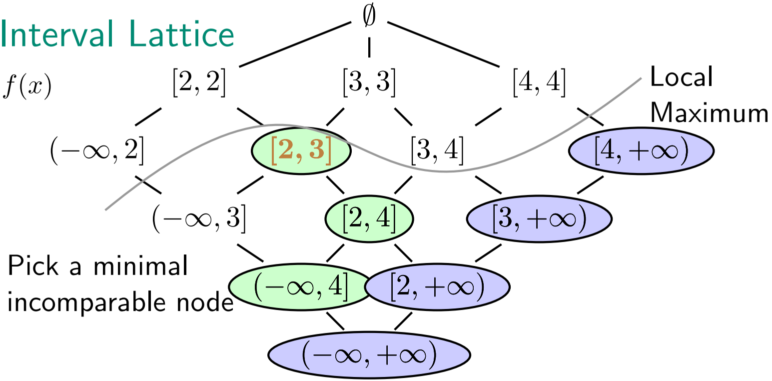


- Interval lattices are useful to filter out a range of packets
- Example: TTL scoping (for network details see paper)

$$obj(I) = \begin{cases} 1 & \text{if } I = [x, y] \text{ or } I = (-\infty, y] \\ -\infty & \text{if } I = [x, \infty) \text{ or } I = (-\infty, \infty) \\ \infty & \text{if } I = \emptyset \end{cases}$$

Example: Interval Lattice

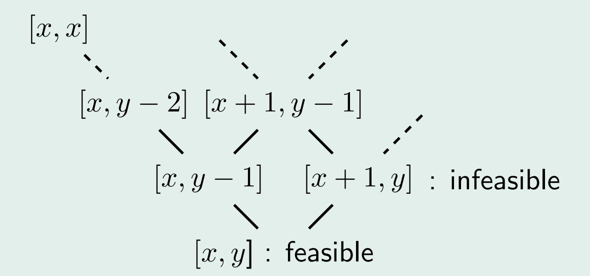
Interval lattice $f(x)$ for $\{2,\ 4\}$

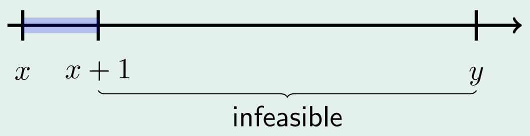


- Interval lattices are useful to filter out a range of packets
- Example: TTL scoping (for network details see paper)

$$obj(I) = \begin{cases} 1 & \text{if } I = [x, y] \text{ or } I = (-\infty, y] \\ -\infty & \text{if } I = [x, \infty) \text{ or } I = (-\infty, \infty) \\ \infty & \text{if } I = \emptyset \end{cases}$$

Heuristic (Feasibility Bound)





- Every feasible interval I above $[x, y]$ must be below (or equal to) $[x, x]$
  - Feasibility is anti-monotonic

Correctness

- Search algorithm is guaranteed to terminate on finite lattices

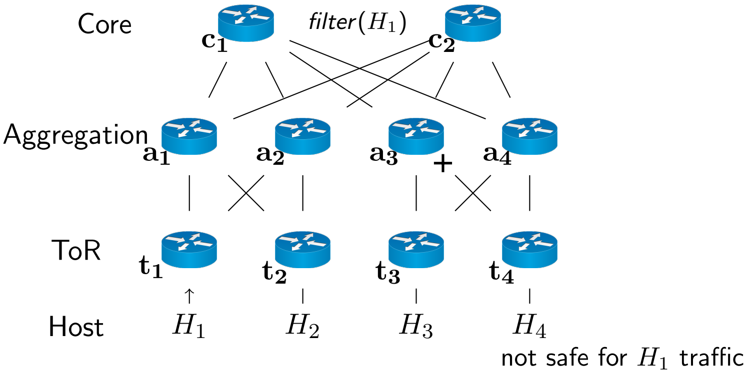
Theorem

- Optimization algorithm is sound and complete
  - Always finds the global optimum

Proof

- Induction on lattice structure
  - use monotonicity of feasibility and objective function

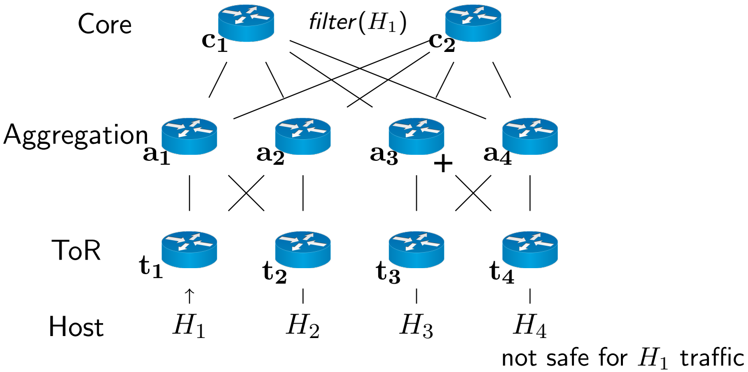
Horn Clauses for Network



Ingress. H_1 sends out the special traffic type 0

$$
\begin{aligned}
(typ = 0 \wedge dst \in \{2,3,4\}) &\quad\rightarrow\quad \mathbf{t_1}(dst, typ) \\
(typ > 0 \wedge typ < 8 \wedge dst \in \{1,3,4\}) &\quad\rightarrow\quad \mathbf{t_2}(dst, typ) \\
(typ > 0 \wedge typ < 8 \wedge dst \in \{1,2,4\}) &\quad\rightarrow\quad \mathbf{t_3}(dst, typ) \\
(typ > 0 \wedge typ < 8 \wedge dst \in \{1,2,3\}) &\quad\rightarrow\quad \mathbf{t_4}(dst, typ)
\end{aligned}
$$

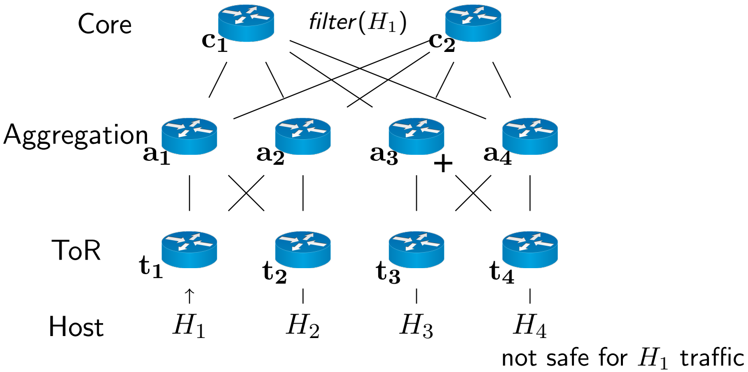
Horn Clauses for Network



We use a special relation symbol $\mathbf{D}$ for dropping a packet

$$\begin{aligned}
\mathbf{t_1}(dst, typ) \wedge (dst \neq 1) &\rightarrow \mathbf{a_1}(dst, typ) \\
\mathbf{t_1}(dst, typ) \wedge (dst \neq 1) &\rightarrow \mathbf{a_2}(dst, typ) \\
\mathbf{t_1}(dst, typ) \wedge \neg\big((dst \geq 1) \wedge & \\
(dst \leq 4) \wedge (typ \geq 0) \wedge (typ \leq 7)\big) &\rightarrow \mathbf{D}(dst, typ)
\end{aligned}$$

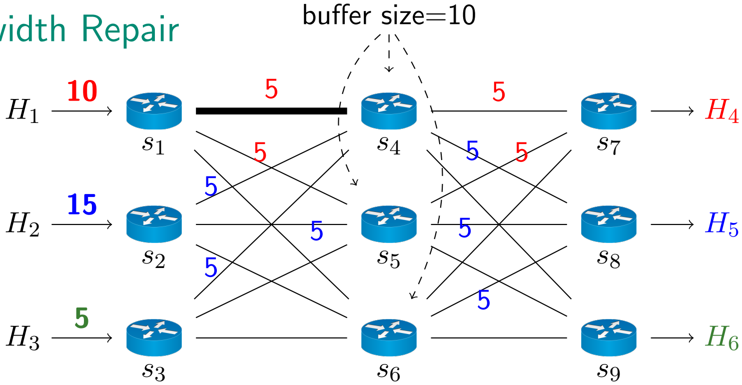
Horn Clauses for Network



Properties. Flow 0 should not reach destination 4 or the drop state

$$\mathbf{t_4}(dst, typ) \wedge (typ = 0) \quad \rightarrow \quad \mathit{false}$$
$$\mathbf{D}(dst, typ) \wedge (typ = 0) \quad \rightarrow \quad \mathit{false}$$

Bandwidth Repair



Not safe for green.

- We use **tokens** to represent the sizes of the flows

$$\mathbf{C}(r_1, b_2, g_3, r_4, b_4, g_4, r_5, b_5, g_5, r_6, b_6, g_6, q_7, q_8, q_9)$$
$$\wedge (r'_1 > 0) \wedge (r_1 \geq r'_1)$$
$$\wedge (r_1 - r'_1 = r'_4 - r_4) \wedge (r'_4 + b_4 + g_4 \leq 10) \rightarrow$$
$$\mathbf{C}(r'_1, b_2, g_3, r'_4, b_4, g_4, r_5, b_5, g_5, r_6, b_6, g_6, q_7, q_8, q_9)$$

Implementation and Experiments

- We use Internet Topology Zoo - real world topologies
- Randomly generate forwarding tables to connect hosts
- Make a set of nodes unsafe for certain types of traffics
- Repair the buggy network with updating a minimal number of switches

Implementation and Experiments

Benchmarks	#Nodes	#Links	#Rels.	#Lattice	#EId	Time(s)
Cesnet200304	29	33	3	2.22×10^{10}	145	4.98
Arpanet19706	9	10	3	2.22×10^{10}	91	2.98
Oxford	20	26	8	3.89×10^{27}	664	16.70
Garr200902	54	71	6	4.92×10^{20}	3045	107.62
Getnet	7	8	2	7.90×10^{6}	61	1.45
Surfnet	50	73	3	2.22×10^{10}	101	3.49
Itnet	11	10	1	2.81×10^{3}	17	0.18
Garr199904	23	25	1	2.81×10^{3}	19	0.33
Darkstrand	28	31	5	1.75×10^{17}	425	14.81
Carnet	44	43	2	7.90×10^{6}	37	0.49
Atmnet	21	22	1	2.81×10^{3}	15	0.67
HiberniaCanada	13	14	11	8.63×10^{37}	1795	84.56
Evolink	37	45	1	2.81×10^{3}	14	0.20
Ernet	30	32	4	6.23×10^{13}	140	4.94
Bren	37	38	6	4.92×10^{20}	974	25.14

Related Work

- Nikolaj Bjørner, Arie Gurfinkel, Ken McMillan, and Andrey Rybalchenko:
  " Horn clause solvers for program verification", 2015.
- Shambwaditya Saha, M. Prabhu, P. Madhusudan:
  "NETGEN: Synthesizing Data-plane configurations for Network Policies", SOSR 2015.
- Aws Albarghouthi, Yi Li, Arie Gurfinkel, Marsha Chechik:
  "UFO: A Framework for Abstraction- and Interpolation-Based Software Verification", CAV 2012.
- Sergey Grebenshchikov, Nuno P. Lopes, Corneliu Popeea, Andrey Rybalchenko:
  "Synthesizing Software Verifiers from Proof Rules", PLDI 2012.
- Anvesh Komuravelli, Arie Gurfinkel, Sagar Chaki and Edmund M. Clarke:
  "Automated Abstraction in SMT-Based Unbounded Software Model Checking", CAV 2013

Summary

Conservative repair procedure:

- Does not add new clauses
- Does not change the structure of the relation symbols
- Can only add constraints to the bodies of clauses

Pros:

- Relation symbols have normally a specific interpretation in the problem domain
- Translation of the repair solution back to the domain is easy

- There are many applications
  - e.g. in software defined networking