Optimizing Horn Solvers for Network Repair

Hossein Hojjat\textsuperscript{1,4} Philipp Rümer \textsuperscript{2}
Jedidiah McClurg\textsuperscript{3} Pavol Černý\textsuperscript{3} Nate Foster \textsuperscript{1}

\textsuperscript{1}Cornell University, \textsuperscript{2}Uppsala University, \textsuperscript{3}University of Colorado Boulder,
\textsuperscript{4}Rochester Institute of Technology

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

Aggregation

ToR

Host

$C_1$

$filter(H_1)$

$C_2$

not safe for $H_1$ traffic

Down for Maintenance

$A_1$

$A_2$

$A_3$

$A_4$

$T_1$

$T_2$

$T_3$

$T_4$

$H_1$

$H_2$

$H_3$

$H_4$
not safe for $H_1$ traffic
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

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
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

Horn Clauses:

\[ \forall \bar{v}. \phi_0(\bar{v}) \land R_{1,0}(\bar{v}) \land \cdots \land R_{n,0}(\bar{v}) \rightarrow R_{0,0}(\bar{v}) \]

\[ \forall \bar{v}. \phi_1(\bar{v}) \land R_{1,1}(\bar{v}) \land \cdots \land R_{n,1}(\bar{v}) \rightarrow R_{0,1}(\bar{v}) \]

\[ \forall \bar{v}. \phi_m(\bar{v}) \land R_{1,m}(\bar{v}) \land \cdots \land R_{n,m}(\bar{v}) \rightarrow R_{0,m}(\bar{v}) \]
Our Repair Approach

\[
\forall \bar{v}. \quad \psi_0(\bar{v}) \land R_{1,0}(\bar{v}) \land \cdots \land R_{n,0}(\bar{v}) \rightarrow R_{0,0}(\bar{v})
\]

\[
\forall \bar{v}. \quad \psi_1(\bar{v}) \land R_{1,1}(\bar{v}) \land \cdots \land R_{n,1}(\bar{v}) \rightarrow R_{0,1}(\bar{v})
\]

\vdots

\[
\forall \bar{v}. \quad \psi_m(\bar{v}) \land R_{1,m}(\bar{v}) \land \cdots \land R_{n,m}(\bar{v}) \rightarrow R_{0,m}(\bar{v})
\]

\[
\forall \bar{v}. \quad \phi_{m'}(\bar{v}) \land R_{1,m'}(\bar{v}) \land \cdots \land R_{n,m'}(\bar{v}) \rightarrow false
\]
Our Repair Approach

∀\bar{v}. R^*_0(\bar{v}) \land \psi_0(\bar{v}) \land R_{1,0}(\bar{v}) \land \cdots \land R_{n,0}(\bar{v}) \rightarrow R_{0,0}(\bar{v})

∀\bar{v}. R^*_1(\bar{v}) \land \psi_1(\bar{v}) \land R_{1,1}(\bar{v}) \land \cdots \land R_{n,1}(\bar{v}) \rightarrow R_{0,1}(\bar{v})

\vdots

∀\bar{v}. R^*_m(\bar{v}) \land \psi_m(\bar{v}) \land R_{1,m}(\bar{v}) \land \cdots \land R_{n,m}(\bar{v}) \rightarrow R_{0,m}(\bar{v})

∀\bar{v}. R^*_m'(\bar{v}) \land \phi_m'(\bar{v}) \land R_{1,m'}(\bar{v}) \land \cdots \land R_{n,m'}(\bar{v}) \rightarrow false

Weaken

- Conjoin fresh relation symbols $R^*_i$ to the bodies of Horn clauses
Our Repair Approach

\[ \forall \bar{v}. \ R^{*}_{0}(\bar{v}) \land \psi_{0}(\bar{v}) \land R_{1,0}(\bar{v}) \land \cdots \land R_{n,0}(\bar{v}) \rightarrow R_{0,0}(\bar{v}) \]

\[ \forall \bar{v}. \ R^{*}_{1}(\bar{v}) \land \psi_{1}(\bar{v}) \land R_{1,1}(\bar{v}) \land \cdots \land R_{n,1}(\bar{v}) \rightarrow R_{0,1}(\bar{v}) \]

\[ \vdots \]

\[ \forall \bar{v}. \ R^{*}_{m}(\bar{v}) \land \psi_{m}(\bar{v}) \land R_{1,m}(\bar{v}) \land \cdots \land R_{n,m}(\bar{v}) \rightarrow R_{0,m}(\bar{v}) \]

\[ \forall \bar{v}. \ R^{*}_{m'}(\bar{v}) \land \phi_{m'}(\bar{v}) \land R_{1,m'}(\bar{v}) \land \cdots \land R_{n,m'}(\bar{v}) \rightarrow false \]

Weaken

- Conjoin fresh relation symbols \( R^{*}_{i} \) 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

\( \forall \bar{v}. R^*_0(\bar{v}) \land \psi_0(\bar{v}) \land R_{1,0}(\bar{v}) \land \cdots \land R_{n,0}(\bar{v}) \rightarrow R_{0,0}(\bar{v}) \)
\( \forall \bar{v}. R^*_1(\bar{v}) \land \psi_1(\bar{v}) \land R_{1,1}(\bar{v}) \land \cdots \land R_{n,1}(\bar{v}) \rightarrow R_{0,1}(\bar{v}) \) \hspace{1cm} \models \text{false}
\[ \vdots \]
\( \forall \bar{v}. R^*_m(\bar{v}) \land \psi_m(\bar{v}) \land R_{1,m}(\bar{v}) \land \cdots \land R_{n,m}(\bar{v}) \rightarrow R_{0,m}(\bar{v}) \)
\( \forall \bar{v}. R^*_m'(\bar{v}) \land R_{1,m'}(\bar{v}) \land \cdots \land R_{n,m'}(\bar{v}) \rightarrow \text{false} \)

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

Strengthen
- Add more constraints to rule out undesirable solutions
- User can select the “best” repairs (e.g. reject \( \text{false} \) 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

Space of all interpretations of relation symbols

Solutions

Best Solutions

Space of all interpretations of relation symbols
Goal: find solutions for set of Horn clauses subject to objective function

Space of all interpretations of relation symbols

Solutions

Best Solutions

Space of all interpretations of relation symbols
Goal: find solutions for set of Horn clauses subject to objective function

Space of all interpretations of relation symbols

Solutions

Best Solutions

∅ 1 2 3 4 · · ·

1 ∪ 2 2 ∪ 3 3 ∪ 4 · · ·

Feasibility Frontier

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

- Space of all interpretations of relation symbols
- Solutions
- Best Solutions

\[ \emptyset \subset 1 \cup 2 \subset 2 \cup 3 \subset 3 \cup 4 \subset \cdots \]

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

Space of all interpretations of relation symbols

Solutions

Best Solutions

∅ ⊆ · · · 1 ∪ 2 ∪ 3 ∪ 4 · · ·

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

**Objective function:**
Rank nodes of lattice monotonically

\[
\emptyset \cup 1 \cup 2 \cup 3 \cup 4 \cdots \cup 1 \cup 2 \cup 3 \cup 4 \cdots \subseteq \cdots \subseteq \text{Feasibility Frontier} \subseteq \cdots \subseteq \text{all interpretations}
\]
**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

Feasibility Frontier

all interpretations
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

Feasibility Frontier

\[
\begin{align*}
\emptyset \\
\subseteq \\
\vdots \\
\vdots
\end{align*}
\]
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)

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

$$ obj(I) = \begin{cases} 
1 & \text{if } I = [x, y] \text{ or } I = (-\infty, y] \\
-\infty & \text{if } I = [x, \infty) \text{ or } I = (-\infty, \infty) \\
\infty & \text{if } I = \emptyset 
\end{cases} $$
Interval lattices are useful to filter out a range of packets
Example: TTL scoping (for network details see paper)

\[
\text{obj}(I) = \begin{cases} 
1 & \text{if } I = [x, y] \text{ or } I = (-\infty, y] \\
-\infty & \text{if } I = [x, \infty) \text{ or } I = (-\infty, \infty) \\
\infty & \text{if } I = \emptyset 
\end{cases}
\]
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 \( \{2, 4\} \)

- Pick a minimal incomparable node

- Local Maximum

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

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

$$ (typ = 0 \land dst \in \{2, 3, 4\}) \rightarrow t_1(dst, typ) $$

$$ (typ > 0 \land typ < 8 \land dst \in \{1, 3, 4\}) \rightarrow t_2(dst, typ) $$

$$ (typ > 0 \land typ < 8 \land dst \in \{1, 2, 4\}) \rightarrow t_3(dst, typ) $$

$$ (typ > 0 \land typ < 8 \land dst \in \{1, 2, 3\}) \rightarrow t_4(dst, typ) $$
Horn Clauses for Network

We use a special relation symbol $D$ for dropping a packet

$$t_1(dst, typ) \land (dst \neq 1) \rightarrow a_1(dst, typ)$$
$$t_1(dst, typ) \land (dst \neq 1) \rightarrow a_2(dst, typ)$$
$$t_1(dst, typ) \land \neg((dst \geq 1) \land (dst \leq 4)) \land (typ \geq 0) \land (typ \leq 7)) \rightarrow D(dst, typ)$$

not safe for $H_1$ traffic
Horn Clauses for Network

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

\[ t_4(dst, typ) \land (typ = 0) \rightarrow false \]
\[ D(dst, typ) \land (typ = 0) \rightarrow false \]
Bandwidth Repair

We use tokens to represent the sizes of the flows

\[ C(r_1, b_2, g_3, r_4, b_4, g_4, r_5, b_5, g_5, r_6, b_6, g_6, q_7, q_8, q_9) \]

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

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

- 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