Programmable Networks with Synthesis

Ahmed ElHassany  Petar Tsankov  Laurent Vanbever  Martin Vechev
Network Misconfigurations are Common
What Makes Network Configuration Hard?

- High-level, global routing requirements
- Low-level, local router configurations
- Multiple interacting routing protocols (OSPF, BGP, ..)

Network N1

Network N2

Network N3
R1: Packets from N1 to N2 must follow the path A → D
R2: Packets from N1 to N3 must follow the path A → B → C → D
Example

R1: Packets from **N1** to **N2** must follow the path **A → D**
R2: Packets from **N1** to **N3** must follow the path **A → B → C → D**
Example

R1: Packets from N1 to N2 must follow the path A → D
R2: Packets from N1 to N3 must follow the path A → B → C → D
Example

R1: Packets from N1 to N2 must follow the path \(A \rightarrow D\)

R2: Packets from N1 to N3 must follow the path \(A \rightarrow B \rightarrow C \rightarrow D\)
Example

<table>
<thead>
<tr>
<th>Network</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>B</td>
</tr>
</tbody>
</table>

R1: Packets from **N1** to **N2** must follow the path **A → D**

R2: Packets from **N1** to **N3** must follow the path **A → B → C → D**

Add a static route to A configuration

Configure A to prefer static routes over OSPF

Configure OSPF link cost

OSPF link cost

Network N1

Network N2

Network N3

Static routes table
R1: Packets from N1 to N2 must follow the path A → D

R2: Packets from N1 to N3 must follow the path A → B → C → D
Example

R1: Packets from **N1** to **N2** must follow the path **A → D**

R2: Packets from **N1** to **N3** must follow the path **A → B → C → D**

---

Network **N1**

Network **N2**

Network **N3**

<table>
<thead>
<tr>
<th>Network</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>B</td>
</tr>
</tbody>
</table>

Static routes table: Lower cost to 5
Example

Router configurations must be such that:

1. A prefers static routes over OSPF
2. A has a static route to B for N3
3. A → D → N2 must be lowest cost from A to N2
4. B → C → D → N3 must be lowest cost from B to N3

R1: Packets from N1 to N2 must follow the path A → D
R2: Packets from N1 to N3 must follow the path A → B → C → D
Current Practice

Initially not configured

Network topology

Routing requirements

Operators manually configure each router

All routers are configured
Current Practice

Initially not configured

Problems and Challenges

- **Diversity** in protocol expressiveness
- Protocol **dependencies**
- No **correctness** guarantees

All routers are configured
Wanted: Programmable Networks with Synthesis

How is the behavior of routers captured?

How to express relevant requirements?

Routing requirements

Network topology

Automatically configure routers with synthesis

How to find a configuration that conforms to the requirements?
Programmable Networks with *Synthesis*: Dimensions

<table>
<thead>
<tr>
<th>Deployment scenarios</th>
<th>Datacenter Incremental</th>
<th>ISP Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing protocols</td>
<td>Deterministic</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>OSPF</td>
<td>BGP</td>
<td>ECMP</td>
</tr>
<tr>
<td>Static routes</td>
<td>MPLS</td>
<td>Gossip</td>
</tr>
<tr>
<td>Requirements</td>
<td>Paths</td>
<td>Reachability</td>
</tr>
<tr>
<td>Isolation</td>
<td>Waypointing</td>
<td>Failures</td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td>Congestion</td>
</tr>
<tr>
<td>Synthesis Techniques</td>
<td>Enumerative learning</td>
<td>Probabilistic</td>
</tr>
<tr>
<td>Symbolic execution</td>
<td>SyNET**</td>
<td>Constraint solving</td>
</tr>
<tr>
<td></td>
<td>CEGIS</td>
<td></td>
</tr>
</tbody>
</table>

**SyNET**: http://synet.ethz.ch
Capturing Network Behavior

Key idea: Express routing protocols, along with their dependencies, in *stratified Datalog*
**Datalog Example**

**Input**

\[\text{parent}(bob, alice)\]
\[\text{parent}(carol, alice)\]
... 

**Program**

\[\text{anc}(X, Y) \leftarrow \text{parent}(X, Y)\]
\[\text{anc}(X, Y) \leftarrow \text{parent}(X, Z), \text{anc}(Z, Y)\]

**Query**

\[\text{anc}(dave, alice)?\]

How is this related to networks?
Datalog Syntax (1/2)

To define a Datalog program, we need:

**Constants:** \( \mathcal{C} = \{ alice, bob, carol, \ldots \} \)
**Variables:** \( \mathcal{V} = \{ X, Y, \ldots \} \)
**Predicates:** \( \mathcal{P} = \{ parent, anc, \ldots \} \)

The sets above can be used to construct the “atoms” of a Datalog program:

**Ground atoms:** \( A_{\mathcal{P}(\mathcal{C})} = \{ p(t_1, \ldots, t_n) \mid p \in \mathcal{P}, \forall 0 \leq i \leq n. t_i \in \mathcal{C} \} \)

**Atoms:** \( A_{\mathcal{P}(\mathcal{C},\mathcal{V})} = \{ p(t_1, \ldots, t_n) \mid p \in \mathcal{P}, \forall 0 \leq i \leq n. t_i \in \mathcal{C} \cup \mathcal{V} \} \)

Example: \( parent(dave, alice) \)
Example: \( parent(X, Y) \)
A *Datalog program* is a set of rules of the form

$$a \leftarrow l_1, \ldots, l_n$$

where $a$ is an atom and $l_1, \ldots, l_n$ are literals of the form $a$ or $\neg a$.

A Datalog program is **well-formed** if for any rule in the program, all variables that appear in the head also appear in the body.

Is this rule well-formed $\text{anc}(X,Y) \leftarrow \text{parent}(Y,Z)$?
Semantics of **Positive** Datalog Programs

Each Datalog program $P$ is associated with a consequence operator $T_P$:

**Interpretations:** $\mathcal{I} = 2^{A_P(c)}$ (an interpretation $I$ is a set of ground atoms)

**Substitutions:** $\sigma: \mathcal{V} \rightarrow \mathcal{C}$ (a substitution maps variables to constants)

**Consequence operator:** $T_P: \mathcal{I} \rightarrow \mathcal{I}$

\[
T_P(I) = \{ \sigma(a) \mid a \leftarrow l_1, \ldots, l_n \in P, \forall i \in [1 \ldots n], I \vdash \sigma(l_i) \} \]

where:
- $I \vdash l_i$ if $l_i = a$ and $a \in I$
- $I \vdash l_i$ if $l_i = \neg a$ and $a \notin I$

A Datalog program $P$ is **positive** if the negation operator does not appear in its rules.

Is $T_P$ monotone if $P$ is a positive Datalog program?

The **semantics** of a positive Datalog program $P$ is given by the least-fixed point of $T_P$.

How can we compute the least-fixed point of $T_P$?
Example

Compute the least-fixed point of the following Datalog program:

\[ p(a) \leftarrow q(X) \]
\[ q(b) \leftarrow r(a), p(b) \]
\[ p(b) \leftarrow r(a) \]

defined over the signature \( \mathcal{C} = \{a, b, c\}, \mathcal{V} = \{X\}, \) and \( \mathcal{P} = \{p, q, r\} \)
Datalog \textit{Inputs}

We can split the set $\mathcal{P}$ of predicates into

1. \textit{input predicates}: predicates that do not appear in the head of rules, and
2. \textit{output predicates}: all remaining predicates

Which are the input/output predicates of this program?

\texttt{anc}(X,Y) \leftarrow \texttt{parent}(X,Y)
\texttt{anc}(X,Y) \leftarrow \texttt{parent}(X,Z), \texttt{anc}(Z,Y)

An \textit{input} for a program $P$ is an interpretation $I$ that contains only atoms constructed using input predicates.

The \textit{semantics} of a positive Datalog program $P$ given an \textit{input} $I$ for $P$ is given by the smallest fixed point of $T_P$ that contains $I$. Let’s denote this by $[P]_I$.

How can we compute $[P]_I$?
Datalog and Negation

What should be the semantics of this program?

\[
p(X) \leftarrow \neg q(X) \\
q(X) \leftarrow p(X) \\
p(X) \leftarrow r(X)
\]

**Problem:** For a Datalog program \( P \) with negation, the consequence operator \( T_P \) is not guaranteed to be monotone!

What about the semantics of this program?

\[
p(X) \leftarrow \neg q(X) \\
q(X) \leftarrow r(X)
\]

This Datalog program is called “stratified”
Semantics of Stratified Datalog

A *Datalog* program $P$ is **stratified** if its rules can be partitioned into sets $P_1, \ldots, P_n$ called strata, such that:

1. for every predicate $p$, all rules with $p$ in their heads are in one stratum $P_i$
2. if a predicate symbol $p$ occurs in a positive literal in $P_i$, then all rules with $p$ in their heads are in a stratum $P_j$ with $j \leq i$
3. if a predicate symbol $p$ occurs in a negative literal in $P_i$, then all rules with $p$ in their heads are in a stratum $P_j$ with $j < i$

What is an example of a Datalog program that is/is not stratified?

The semantics of a stratified Datalog program $P$, with strata $P_1, \ldots, P_n$, and an input $I$ for $P$, is given by the fixed-point $M_n$ where $M_0 = I$, and $M_i = [[P_i]]_{M_{i-1}}$, for $i \in [1, n]$.

Is $M_n$ unique for any stratified Datalog program? What if we partition the rules into different partitions?
Encoding Network Behavior in stratified Datalog
Datalog (2/3 Graph Reachability)

**Input**

\[ \text{link}(n1, a) \]
\[ \text{link}(a, b) \]
... 

**Program**

\[ \text{path}(X, Y) \leftarrow \text{link}(X, Y) \]
\[ \text{path}(X, Y) \leftarrow \text{link}(X, Z), \text{path}(Z, Y) \]

**Query**

\[ \text{path}(n1, n2) \]?

Can we capture the network’s forwarding plane?
Datalog (3/3 Shortest-path Routing)

**Input**

Input

\[
\text{link}(n_1, a, 10)
\]

**Program**

Program

\[
\begin{align*}
\text{path}(\text{Router}, \text{Net}, \text{Net}, \text{Cost}) & \leftarrow \\
\quad & \text{link}(\text{Router}, \text{Net}, \text{Cost}) \\
\text{path}(\text{Router}, \text{Net}, \text{Net}, \text{Net}, \text{Cost}, C_1 + C_2) & \leftarrow \\
\quad & \text{link}(\text{Router}, \text{Net}, \text{Net}, \text{Net}, \text{Cost}, C_1), \\
\quad & \text{path}(\text{Net}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, X, C_2) \\
\text{sp}(\text{Router}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, \text{min}(C)) & \leftarrow \\
\quad & \text{path}(\text{Router}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, C) \\
\text{fwd}(\text{Router}, \text{Net}, \text{Net}) & \leftarrow \\
\quad & \text{sp}(\text{Router}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, \text{Net}, C)
\end{align*}
\]

**Query**

Query

\[
\text{fwd}(a, n_2, ?)
\]
Routing Requirements as Datalog Queries

**Paths**

*Packets for traffic class TC must follow the path*

\[ r_1 \to \cdots \to r_n \]
Routing Requirements as Datalog Queries

Paths
Packets for traffic class $TC$ must follow the path $r_1 \rightarrow \cdots \rightarrow r_n$

Traffic isolation
The paths for two distinct traffic classes $tc_1$ and $tc_2$ do not share links in the same direction

\[ \forall R_1, R_2. \ fwd(R_1, tc_1, R_2) \Rightarrow \neg fwd(R_1, tc_2, R_2) \]

\[ \text{fwd}(r_1, tc, r_2) \land \cdots \land \text{fwd}(r_{n-1}, tc, r_n) \]
Routing Requirements as Datalog Queries

**Paths**
Packets for traffic class $TC$ must follow the path $r_1 \rightarrow \cdots \rightarrow r_n$

**Traffic isolation**
The paths for two distinct traffic classes $tc_1$ and $tc_2$ do not share links in the same direction

**Reachability**
Packets for traffic class $tc$ can reach router $r_2$ from router $r_1$
Routing Requirements as Datalog Queries

**Paths**
Packets for traffic class $TC$ must follow the path $r_1 \rightarrow \cdots \rightarrow r_n$

**Traffic isolation**
The paths for two distinct traffic classes $tc_1$ and $tc_2$ do not share links in the same direction

\[ \forall R_1, R_2. \text{fwd}(R_1, tc_1, R_2) \Rightarrow \neg \text{fwd}(R_1, tc_2, R_2) \]

**Reachability**
Packets for traffic class $tc$ can reach router $r_2$ from router $r_1$

\[ \text{reach}(r_1, tc, r_2) \]

**Loop-freeness**
The forwarding plane has no loops

\[ \forall TC, R. \neg \text{reach}(R, TC, R) \]
Analysis of Network Configurations in Datalog
Analysis of Network Configurations in Datalog

**Network-wide configuration** $C$
(Protocol configurations for routers)

**Network specification** $N$
(OSPF, BGP, MPLS, ...)

**Routing requirements** $R$
(Isolation, reachability, reliability)

**Datalog input** $I$

**Datalog program** $P$

**Datalog query** $Q$

**Analysis question:**
Does the network $N$ configured with $C$ satisfy the requirements $R$?

**Query entailment:**
Does $P, I \models Q$ hold?
Analysis of Network Configurations in Datalog

**Network-wide configuration** $C$
(Protocol configurations for routers)

**Network specification** $N$
(OSPF, BGP, MPLS, ...)

**Routing requirements** $R$
(Isolation, reachability, reliability)

---

**Datalog input $I$**

**Datalog program $P$**

---

**Theorem:** Query entailment in Datalog is in PTIME

---

**Analysis question:**
*Does the network $N$ configured with $C$ satisfy the requirements $R$?*

---

**Query entailment:**
*Does $P, I \models Q$ hold?*
Network-wide Configuration Synthesis
Network-wide Configuration Synthesis

Network specification $N$
(OSPF, BGP, MPLS, ...)

Routing requirements $R$
(isolation, reachability, reliability)

Synthesis problem:
Find a configuration $C$ such that $N$ configured with $C$ satisfies $R$

Network-wide configuration $C$
(protocol configurations for routers)

Datalog program $P$

Datalog query $Q$

(Input) Synthesis problem:
Find an input $I$ such that $P, I \models Q$

Datalog input $I$
Network-wide Configuration Synthesis

**Network specification** $N$  
(OSPF, BGP, MPLS, ...)

**Routing requirements** $R$  
(isolation, reachability, reliability)

**Synthesis problem:**  
Find a configuration $C$ such that $N$ configured with $C$ satisfies $R$

**Datalog program** $P$

**Datalog query** $Q$

**Network-wide configuration** $C$  
(protocol configurations for routers)

**Problems:**
- No input synthesis tools for Datalog
- Problem is undecidable

**Datalog input** $I$
Input Synthesis for Datalog

Key idea: Reduce to solving SMT constraints
Input Synthesis for Positive Datalog (First Attempt)

Datalog program $P$

\[
\begin{align*}
\text{path}(X,Y) & \leftarrow \text{link}(X,Y) \\
\text{path}(X,Y) & \leftarrow \text{link}(X,Z), \text{path}(Z,Y)
\end{align*}
\]

Datalog query $Q$

\[
\begin{align*}
\text{path}(a,c) & \land \neg \text{link}(a,c)
\end{align*}
\]

Generate SMT constraints

\[
\begin{align*}
\forall X,Y. \text{path}(X,Y) & \leftarrow \text{link}(X,Y) \\
\forall X,Y. \text{path}(X,Y) & \leftarrow \exists Z. \text{link}(X,Z) \land \text{path}(Z,Y) \\
\text{path}(a,c) & \land \neg \text{link}(a,c)
\end{align*}
\]

Solve SMT constraints

\[
\begin{align*}
\text{path}(a,c)
\end{align*}
\]

Derive input (by project on input predicates)

\[
\emptyset
\]

Unfortunately, we get $P, I \not\models Q$
Input Synthesis for Positive Datalog (1/2)

Datalog program $\mathbf{P}$
\begin{align*}
\text{path}(X,Y) & \leftarrow \text{link}(X,Y) \\
\text{path}(X,Y) & \leftarrow \text{link}(X,Z), \text{path}(Z,Y)
\end{align*}

Datalog query $\mathbf{Q}$
\begin{align*}
\text{path}(a,c) & \land \neg \text{link}(a,c)
\end{align*}

Generate SMT constraints
\begin{align*}
\forall X,Y. & \text{path}_1(X,Y) \iff \text{link}(X,Y) \\
\forall X,Y. & \text{path}_2(X,Y) \iff (\text{link}(X,Y) \lor (\exists Z. (\text{link}(X,Z) \land \text{path}_1(Z,Y)))) \\
\text{path}_2(a,c) & \land \neg \text{link}(a,c)
\end{align*}

Bounded unrolling for positive queries

SMT Constraints $\mathbf{\psi}$
\begin{align*}
\text{link}(a,b), & \text{link}(b,c), \\
\text{path}_1(a,b), & \text{path}_1(b,c) \\
\text{path}_2(a,b), & \text{path}_2(b,c), \text{path}_2(a,c)
\end{align*}

Solve SMT constraints
\begin{align*}
\text{link}(a,b), & \text{link}(b,c), \\
\text{path}_1(a,b), & \text{path}_1(b,c) \\
\text{path}_2(a,b), & \text{path}_2(b,c), \text{path}_2(a,c)
\end{align*}

Model $\mathbf{M} \models \mathbf{\psi}$

Derive input
\begin{align*}
\text{link}(a,b), & \text{link}(b,c)
\end{align*}

Datalog input $\mathbf{I}$

\textit{path}(a,c) is a positive query
Input Synthesis for Positive Datalog (2/2)

Datalog program \( P \)

\[
\begin{align*}
\text{path}(X,Y) & \leftarrow \text{link}(X,Y) \\
\text{path}(X,Y) & \leftarrow \text{link}(X,Z), \text{path}(Z,Y)
\end{align*}
\]

Datalog query \( Q \)

\[
\begin{align*}
\neg \text{path}(a,c) & \land \neg \text{link}(a,c)
\end{align*}
\]

Generate SMT constraints

\[
\begin{align*}
\forall X, Y. \text{path}(X,Y) & \leftarrow \text{link}(X,Y) \\
\forall X, Y. \text{path}(X,Y) & \leftarrow \exists Z. \text{link}(X,Z) \land \text{path}(Z,Y) \\
\neg \text{path}(a,c) & \land \neg \text{link}(a,c)
\end{align*}
\]

Solve SMT constraints

\[
\emptyset
\]

Derive input

\[
\emptyset
\]

\( \text{path}(a,c) \) is a negative query

No unrolling for negative queries

SMT constraints \( \psi \)

Model \( M \models \psi \)

Datalog input \( I \)
Input Synthesis for Positive Datalog (2/2)

Datalog program $P$

- $\text{path}(X,Y) \leftarrow \text{link}(X,Y)$
- $\text{path}(X,Y) \leftarrow \text{link}(X,Z), \text{path}(Z,Y)$

Datalog query $Q$

- $\neg \text{path}(a,c) \land \neg \text{link}(a,c)$

Generate SMT constraints

SMT Constraints $\psi$

- $\forall X, Y. \text{path}(X,Y) \iff \text{link}(X,Y)$
- $\forall X, Y. \text{path}(X,Y) \iff \exists Z. \text{link}(X,Z) \land \text{path}(Z,Y)$
- $\neg \text{path}(a,c) \land \neg \text{link}(a,c)$

Solve SMT constraints

Model $M \models \psi$

Derive input

Datalog input $I$

Summary:

- Unroll rules for positive queries, do not unrolling rules for negative queries
- Combine both kinds of constraints for constraints that contain both positive/negative queries.
Input Synthesis for *Stratified* Datalog

Suppose we have a program $P$ with strata $P_1, \ldots, P_n$, and a query $Q$.

High-level idea:

**Synth $P_n$**: Compute input $I_n$ for stratum $P_n$ such that $\llbracket P_n \rrbracket_{I_n}$ satisfies $Q$.

**Synth $P_{n-1}$ ... $P_1$**: Compute input $I_i$ for stratum $P_i$ such that $\llbracket P_i \rrbracket_{I_i}$ produces the input $I_{i+1}$ synthesized by the previous step.

**Back step**: Backtrack to step Synth $P_i$ if the step Synth $P_{i-1}$ returns unsat.
Network-wide Configuration Synthesis

Encode as a Datalog program

Datalog input identifies correct configuration

Network topology

Routing requirements

Automatically configure routers with synthesis

Constraints on the forwarding plane computed by the routers

Via reduction to input synthesis for Datalog
Software Synthesis @ SRL

Develop new synthesis techniques to solve practical system challenges

Intent

Desired properties

Software Synthesis

Probabilistic

CEGIS

Oracle-guided

Constraint-based

Big Code

Application domains

Computer networks

Security and privacy

Modern architectures

Datacenters

Data science

End-user programming
Implementation
Implementation

The SyNET system ([http://synet.ethz.ch](http://synet.ethz.ch))

- Written in **Python** (≈ 4K lines of code)
- Protocols encoded in **stratified Datalog** (≈ 100 rules)
- Uses the **Z3** constraint solver. Relies on linear integer arithmetic theories (**LIA**).
- Outputs **CISCO** configurations
- Supports **BGP**, **OSPF**, and **static routes**

Network-specific optimizations

- Partial evaluator for Datalog
- Protocol-specific constraints

Sample **CISCO** configuration output by SyNET

```plaintext
! A snippet from router A
interface f0/1
    ip address 10.0.0.2 255.255.255.254
    ip ospf cost 10
    description "To B"
interface f0/0
    ip address 10.0.0.0 255.255.255.254
    ip ospf cost 65530
    description "To C"
interface f1/0
    ip address 10.0.0.4 255.255.255.254
    ip ospf cost 65530
    description "To D"
```
Experiments
Experiment

US-based network connecting major universities and research institutes

<table>
<thead>
<tr>
<th>Protocols / # Traffic classes</th>
<th>1 class</th>
<th>5 classes</th>
<th>10 classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>1.3s</td>
<td>2.0s</td>
<td>4.0s</td>
</tr>
<tr>
<td>Static + OSPF</td>
<td>9.0s</td>
<td>21.3s</td>
<td>49.3s</td>
</tr>
<tr>
<td>Static + OSPF + BGP</td>
<td>13.3s</td>
<td>22.7s</td>
<td>1m19.7s</td>
</tr>
</tbody>
</table>

Synthesis Times
Scalability Experiment

- Grid topologies with up to 64 routers
- Requirements for 10 traffic classes

<table>
<thead>
<tr>
<th># of routers</th>
<th>Synthesis Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>&gt;1h for Static + OSPF</td>
</tr>
<tr>
<td>16</td>
<td>&lt;24h for Static + OSPF</td>
</tr>
<tr>
<td>25</td>
<td>&gt;24h for Static + OSPF + BGP and networks with &gt;36 routers</td>
</tr>
<tr>
<td>36</td>
<td>&lt;24h for Static + OSPF</td>
</tr>
<tr>
<td>49</td>
<td>&lt;1h for static routes</td>
</tr>
<tr>
<td>64</td>
<td></td>
</tr>
</tbody>
</table>
Summary: Programmable Networks with Synthesis

Global requirements vs local configurations

Network-wide configuration synthesis

Approach scales to realistic problems

For more details read this paper: https://arxiv.org/abs/1611.02537