A Language-Based Approach to Network Verification and Synthesis

Nate Foster
Cornell University

Microsoft Research
Faculty Summit 2015
Challenges

Networks are a critical part of our computing infrastructure...

...they have grown dramatically in size and complexity...

... and are quickly becoming unwieldy for operators to manage!
Network Management

Operators use a variety of techniques to keep networks running such as:

- Generating low-level configurations from high-level policies
- Scraping configurations using command-line interfaces
- Diagnosing errors using \textit{ping} and \textit{traceroute}
Toward Design Automation

1. Design high-level languages that model essential network features

2. Develop semantics that enables reasoning precisely about behavior

3. Build tools to synthesize low-level implementations automatically
Focus on *reachability properties* that capture the essential function of a network: moving data from one location to another.
A *machine model* describes behavior in terms of concepts like pipelines of hardware lookup tables.
A *machine model* describes behavior in terms of concepts like pipelines of hardware lookup tables.

A *programming model* describes behavior in terms of concepts like mathematical functions on packets.
What should a network programming language provide?

Two essential features:
- Packet classifiers
- Forwarding paths
NetKAT Language

pol ::= false
   | true
   | field = val
   | pol₁ + pol₂
   | pol₁ ; pol₂
   | !pol
   | pol*
   | field := val
   | S⇒T
NetKAT Language

\[
\text{pol ::= } \begin{align*}
& \text{false} \\
& \text{true} \\
& \text{field } = \text{ val} \\
& \text{pol}_1 + \text{ pol}_2 \\
& \text{pol}_1 ; \text{ pol}_2 \\
& \text{!pol} \\
& \text{pol}^* \\
& \text{field } := \text{ val} \\
& S \Rightarrow T
\end{align*}
\]

Boolean Algebra
NetKAT Language

| pol ::=  | **false**             | Boolean Algebra |
|         | **true**               | +               |
|         | field = val            | Kleene Algebra  |
|         | pol₁ + pol₂            |                |
|         | pol₁; pol₂             |                |
|         | !pol                   |                |
|         | pol*                   |                |
|         | field ::= val          |                |
|         | S⇒T                    |                |
NetKAT Language

\[
pol ::= \text{false} \mid \text{true} \mid \text{field} = \text{val} \mid pol_1 + pol_2 \mid pol_1 ; pol_2 \mid !\text{pol} \mid pol^* \mid \text{field} ::= \text{val} \mid S \Rightarrow T
\]

Boolean Algebra

+ 

Kleene Algebra

+ 

Packet Primitives
NetKAT Language

pol ::= \texttt{false} | \texttt{true} | \texttt{field} = \texttt{val} | pol\_1 + pol\_2 | pol\_1 ; pol\_2 | \texttt{!pol} | pol^* | \texttt{field} ::= \texttt{val} | S⇒T

\begin{itemize}
  \item Boolean Algebra
  \item Kleene Algebra
  \item Packet Primitives
\end{itemize}

\textbf{KAT} [Kozen '96]
### NetKAT Language

| pol ::= | **false** | **true** |
|         | field = val | pol₁ + pol₂ |
|         | pol₁ ; pol₂ | !pol |
|         | pol*        | field ::= val |
|         | S⇒T         |

- **Boolean Algebra**
- **Kleene Algebra**
- **Packet Primitives**

KAT

- **KAT** [Kozen '96]

NetKAT

- **NetKAT** [Anderson et al. '14]
NetKAT Language

pol ::= \texttt{false} | \texttt{true} | \texttt{field = val} | pol_1 + pol_2 | pol_1 ; pol_2 | \neg pol_1 | pol_1 * | field ::= val | S \Rightarrow T

\textbf{if } p_1 \textbf{ then } p_2 \textbf{ else } p_3 \triangleq (p_1 ; p_2) + (!p_1 ; p_3)

\textbf{Boolean Algebra}

\textbf{Kleene Algebra}

\textbf{Packet Primitives}

\textbf{KAT} [Kozen '96]

\textbf{POPL '14}
Sequential composition $\text{pol}_1 ; \text{pol}_2$ runs the input through $\text{pol}_1$ and then runs every output through $\text{pol}_2$. 

$\text{pol} ::= \text{false} | \text{true} | \text{field} = \text{val} | \text{pol}_1 + \text{pol}_2 | \text{pol}_1 ; \text{pol}_2 | !\text{pol} | \text{pol}^* | \text{field} := \text{val} | S \Rightarrow T$
Encodings

Switch forwarding tables and network topologies can be represented in NetKAT using simple encodings

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>dstport=22</td>
<td>Drop</td>
</tr>
<tr>
<td>srcip=10.0.0.1</td>
<td>Forward 1</td>
</tr>
<tr>
<td>*</td>
<td>Forward 2</td>
</tr>
</tbody>
</table>

```
if dstport=22 then false
else if srcip=10.0.0.1 then port := 1
else port := 2
```

```
A ⇒ B + B ⇒ A + B ⇒ C + C ⇒ B
```
Networks

The behavior of an entire network can be encoded in NetKAT by interleaving steps of processions by switches and topology.

\[
\text{policy}+\ (\text{policy}; \text{topo}); \text{policy} \\
+ \ (\text{policy}; \text{topo}; \text{policy}; \text{topo}); \text{policy} \\
\vdots \\
(\text{policy}; \text{topo})^*; \text{policy}
\]
Reachability

Given a network, want to be able to answer questions like:

“Does the network forward from ingress to egress?
Can reduce this question (and many others) to equivalence

\[
in; (policy; topo)^*; policy; out \equiv in; out
\]
Reachability

Given a network, want to be able to answer questions like:

“Does the network forward from ingress to egress?

Can reduce this question (and many others) to equivalence

\[ \text{in}; (\text{policy}; \text{topo})^*; \text{policy}; \text{out} \equiv \text{in}; \text{out} \]

Other properties:

- Access control
- Traffic Isolation
- Loop freedom
- Blackhole freedom
Kleene Algebra Axioms

\[ p + (q + r) \equiv (p + q) + r \]
\[ p + q \equiv q + p \]
\[ p + false \equiv p \]
\[ p + p \equiv p \]
\[ p ; (q ; r) \equiv (p ; q) ; r \]
\[ p ; (q + r) \equiv p ; q + p ; r \]
\[ (p + q) ; r = p ; r + q ; r \]
\[ true ; p = p \]
\[ p = p ; true \]
\[ false ; p = false \]
\[ p ; false = false \]
\[ true + p ; p^* = p^* \]
\[ true + p^* ; p = p^* \]
\[ p + q ; r + r = r \Rightarrow p^* ; q + r = r \]
\[ p + q ; r + q = q \Rightarrow p ; r^* + q = q \]

Boolean Algebra Axioms

\[ a + (b ; c) \equiv (a + b) ; (a + c) \]
\[ a + true = true \]
\[ a + ! a = true \]
\[ a ; b = b ; a \]
\[ a ; ! a \equiv false \]
\[ a ; a = a \]

Packet Axioms

\[ f := n ; f' := n' \Rightarrow n' ; f := n \quad \text{if} \quad f \neq f' \]
\[ f := n ; f' = n' \Rightarrow f' = n' ; f := n \quad \text{if} \quad f \neq f' \]
\[ f := n ; f = n = f := n \]
\[ f = n ; f := n = f = n \]
\[ f := n ; f := n' \Rightarrow f := n' \]
\[ f = n ; f = n' \Rightarrow false \quad \text{if} \quad n \neq n' \]
\[ A \Rightarrow B ; f = n = f = n ; A \Rightarrow B \quad \text{if} \quad f \neq \text{switch} \]
<table>
<thead>
<tr>
<th>Kleene Algebra Axioms</th>
<th>Boolean Algebra Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p + (q + r) = (p + q) + r )</td>
<td>( a + (b ; c) = (a + b) ; (a + c) )</td>
</tr>
<tr>
<td>( p + q = q + p )</td>
<td>( a + \text{true} = \text{true} )</td>
</tr>
<tr>
<td>( p + \text{false} = p )</td>
<td>( a + !a = \text{true} )</td>
</tr>
<tr>
<td>( p + p = p )</td>
<td>( a ; b = b ; a )</td>
</tr>
<tr>
<td>( p ; (q; r) = (p; q); r )</td>
<td>( a ; !a = \text{false} )</td>
</tr>
<tr>
<td>( p; (q + r) = p; q + p; r )</td>
<td>( a ; a = a )</td>
</tr>
<tr>
<td>( (p + q); r = p; r + q; r )</td>
<td></td>
</tr>
<tr>
<td>( \text{true}; p = p )</td>
<td></td>
</tr>
<tr>
<td>( p = p; \text{true} )</td>
<td></td>
</tr>
<tr>
<td>( \text{false}; p = \text{false} )</td>
<td></td>
</tr>
<tr>
<td>( p; \text{false} = \text{false} )</td>
<td></td>
</tr>
<tr>
<td>( \text{true} )</td>
<td></td>
</tr>
<tr>
<td>( \text{true} )</td>
<td></td>
</tr>
<tr>
<td>( p )</td>
<td></td>
</tr>
<tr>
<td>( p )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packet Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f := n; f' := n' = f' := n'; f := n ) if ( f \neq f' )</td>
</tr>
<tr>
<td>( f := n; f' = n' = f' = n'; f := n ) if ( f \neq f' )</td>
</tr>
<tr>
<td>( f := n; f = n = f := n )</td>
</tr>
<tr>
<td>( f = n; f := n = f = n )</td>
</tr>
<tr>
<td>( f := n; f := n' = f := n' ) if ( n \neq n' )</td>
</tr>
<tr>
<td>( f = n; f = n' = \text{false} ) if ( n \neq n' )</td>
</tr>
<tr>
<td>( A \Rightarrow B; f = n = f = n; A \Rightarrow B ) if ( f \neq \text{switch} )</td>
</tr>
</tbody>
</table>
NetKAT Proof System

Kleene Algebra Axioms

\[ p + (q + r) = (p + q) + r \]
\[ p + q = q + p \]
\[ p + \text{false} = p \]
\[ p + p = p \]
\[ p; (q; r) = (p; q); r \]
\[ p; (q + r) = p; q + p; r \]
\[ (p + q); r = p; r + q; r \]
\[ \text{true}; p = p \]
\[ p = p; \text{true} \]
\[ \text{false}; p = \text{false} \]
\[ p; \text{false} = \text{false} \]
\[ \text{true} \]
\[ \text{true} \]
\[ p \]
\[ p \]

Boolean Algebra Axioms

\[ a + (b ; c) = (a + b) ; (a + c) \]
\[ a + \text{true} = \text{true} \]
\[ a + ! a = \text{true} \]
\[ a ; b = b ; a \]
\[ a ; ! a = \text{false} \]
\[ a ; a = a \]

Packet Axioms

\[ f := n; f' := n' = f' := n' ; f := n \text{ if } f \neq f' \]
\[ f := n; f' = n' = f' = n' ; f := n \text{ if } f \neq f' \]
\[ f := n; f = n = f := n \]
\[ f = n; f := n = f = n \]
\[ f := n; f := n' = f := n' \]
\[ f = n; f = n' = \text{false} \text{ if } n \neq n' \]
\[ A \Rightarrow B; f = n = f = n; A \Rightarrow B \text{ if } f \neq \text{switch} \]
**NetKAT Proof System**

**Kleene Algebra Axioms**

\[
\begin{align*}
 p + (q + r) &= (p + q) + r \\
p + q &= q + p \\
p + \text{false} &= p \\
p + p &= p \\
p; (q; r) &= (p; q); r \\
p; (q + r) &= p; q + p; r \\
(p + q); r &= p; r + q; r \\
\text{true}; p &= p \\
p &= p; \text{true} \\
\text{false}; p &= \text{false} \\
p; \text{false} &= \text{false} \\
\text{true} &= \text{true} \\
p &= p \\
p &= p
\end{align*}
\]

**Boolean Algebra Axioms**

\[
\begin{align*}
a + (b; c) &= (a + b); (a + c) \\
a + \text{true} &= \text{true} \\
a + !a &= \text{true} \\
a; b &= b; a \\
a; !a &= \text{false} \\
a; a &= a
\end{align*}
\]

**Packet Axioms**

\[
\begin{align*}
f := n; f' := n' &= f' := n' ; f := n \quad \text{if } f \neq f' \\
f := n; f' := n' &= f' := n' ; f := n \quad \text{if } f \neq f' \\
f := n; f := n &= f := n \\
f = n; f := n &= f := n \\
f := n; f := n' &= f := n' \\
f = n; f := n' &= \text{false} \quad \text{if } n \neq n' \\
A \Rightarrow B; f = n &= f = n; A \Rightarrow B \quad \text{if } f \neq \text{switch}
\end{align*}
\]
NetKAT Proof System

Kleene Algebra Axioms
\[ p + (q + r) = (p + q) + r \]
\[ p + q = q + p \]
\[ p + \text{false} = p \]
\[ p + p = p \]
\[ p; (q; r) = (p; q); r \]
\[ p; (q + r) = p; q + p; r \]
\[ (p + q); r = p; r + q; r \]
\[ \text{true}; p = p \]
\[ p = p; \text{true} \]
\[ \text{false}; p = \text{false} \]
\[ p; \text{false} = \text{false} \]
\[ \text{true} \]
\[ \text{true} \]
\[ p \]
\[ p \]

Boolean Algebra Axioms
\[ a + (b ; c) = (a + b) ; (a + c) \]
\[ a + \text{true} = \text{true} \]
\[ a + ! a = \text{true} \]
\[ a ; b = b ; a \]
\[ a ; !a = \text{false} \]
\[ a ; a = a \]

Packet Axioms
\[ f := n; f' := n' = f' := n' ; f := n \quad \text{if } f \neq f' \]
\[ f := n; f' = n' = f' = n' ; f := n \quad \text{if } f \neq f' \]
\[ f := n; f = n = f := n \]
\[ f = n; f := n = f = n \]
\[ f := n; f := n' = f := n' \]
\[ f = n; f = n' = \text{false} \quad \text{if } n \neq n' \]
\[ A \Rightarrow B; f = n = f = n; A \Rightarrow B \quad \text{if } f \neq \text{switch} \]
NetKAT Proof System

Kleene Algebra Axioms

\[ p + (q + r) = (p + q) + r \]
\[ p + q = q + p \]
\[ p + \text{false} = p \]
\[ p + p = p \]
\[ p; (q; r) = (p; q); r \]
\[ p; (q + r) = p; q + p; r \]
\[ (p + q); r = p; r + q; r \]

Boolean Algebra Axioms

\[ a + (b ; c) = (a + b) ; (a + c) \]
\[ a + \text{true} = \text{true} \]
\[ a + \lnot a = \text{true} \]
\[ a ; b = b ; a \]
\[ a ; \lnot a = \text{false} \]
\[ a ; a = a \]

Packet Axioms

\[ f := n; f' := n' = f' := n' ; f := n \quad \text{if } f \neq f' \]
\[ f := n; f' = n' = f' = n' ; f := n \quad \text{if } f \neq f' \]
\[ f := n; f = n = f := n \]
\[ f = n; f := n = f = n \]
\[ f := n; f := n' = f := n' \]
\[ f = n; f = n' = \text{false} \quad \text{if } n \neq n' \]
\[ A \Rightarrow B; f = n = f = n; A \Rightarrow B \quad \text{if } f \neq \text{switch} \]
NetKAT Proof System

Kleene Algebra Axioms
\[ p + (q + r) = (p + q) + r \]
\[ p + q = q + p \]
\[ p + \text{false} = p \]
\[ p + p = p \]
\[ p; (q; r) = (p; q); r \]
\[ (p + q); r = p; r + q; r \]
\[ \text{true}; p = p \]
\[ p = p; \text{true} \]
\[ p = \text{false}; \]
\[ p = \text{false}; \]
\[ p = \text{true}; \]
\[ p = \text{true}; \]

Boolean Algebra Axioms
\[ a + (b ; c) = (a + b) ; (a + c) \]
\[ a + \text{true} = \text{true} \]
\[ a + \neg a = \text{true} \]
\[ a ; b = b ; a \]
\[ a ; \neg a = \text{false} \]
\[ a ; a = a \]

Soundness: If \( \vdash p \equiv q \), then \( \llbracket p \rrbracket = \llbracket q \rrbracket \)
Completeness: If \( \llbracket p \rrbracket = \llbracket q \rrbracket \), then \( \vdash p \equiv q \)
NetKAT Automata

Can exploit NetKAT’s regular structure to build equivalent finite automata

Automata provide a practical way to decide program equivalence

Prototype implementation performs well on Topology Zoo benchmarks

\[(x=1; x:=2; A \Rightarrow B + x=2; x:=1; B \Rightarrow A)^*\]
Regular paths have many uses:

- Network Virtualization
- Traffic Engineering
- Fault Tolerance
- Application Intent
Verified Implementation

**Question:** How can we know the NetKAT compiler is correct?

**Answer:** implement it in a proof assistant!

- Formalize source and target languages in Coq
- Prove that transformations preserve semantics
- Extract code to OCaml and execute on real hardware

[PLDI ’13]
OpenFlow Specification

42 pages...

...of informal prose

...diagrams and flow charts

...and C struct definitions
Syntax

- Models all features related to packet forwarding and all essential asynchrony
- Supports arbitrary controllers

Featherweight OpenFlow

Semantics

\[ (\text{outp}', \text{outm}') = [\text{RT}](\text{inm}) \]

\[ [\text{inm}, \text{pts}, \text{RT}, \{\text{ip}, \text{pt}\}] \rightarrow [\text{inm}, \text{pts}, \text{RT}, \{\text{ip}, \text{pt}\}] \] (Switch-PktOut)

\[ [\text{inm}, \text{pts}, \text{RT}, \{\text{ip}, \text{pt}\}] \rightarrow [\text{inm}, \text{pts}, \text{RT}, \{\text{ip}, \text{pt}\}] \] (Switch-FlowMod)

\[ \text{RT}' = \text{apply}(\Delta \text{RT}, \text{RT}) \]

\[ \text{RT}' = \text{apply}(\Delta \text{RT}, \text{RT}) \]

\[ \text{RT}' = \text{apply}(\Delta \text{RT}, \text{RT}) \]

\[ \text{RT}' = \text{apply}(\Delta \text{RT}, \text{RT}) \]
Weak Bisimulation

$$(H_1, \text{信封})$$
Weak Bisimulation

\[(H_1, \text{envelope}) \rightarrow (S_1, pt_1, \text{envelope}) \rightarrow (S_2, pt_1, \text{envelope}) \rightarrow (H_2, \text{envelope})\]
Weak Bisimulation

\[(H_1, \text{envelope}) \rightarrow (S_1, pt_1, \text{envelope}) \rightarrow (S_2, pt_1, \text{envelope}) \rightarrow (H_2, \text{envelope})\]
Theorem: NetKAT semantics is weakly bisimilar to Featherweight OpenFlow + run-time system
Network Updates

**Question:** how can we gracefully transition the network from one program to another?
Consistent Updates

**Operationally:** every packet (or flow) processed using a consistent version of the network-wide configuration

**Semantically:** guarantee preserves all safety properties

**Implementations:** many different possibilities—e.g., one option is to use a two-phase distributed protocol
Update Synthesis

$\phi$

Logical property

topology + configurations
Update Synthesis

$\phi$
Conclusion

• Programming languages and formal methods have a key role to play in next-generation networking platforms
• The NetKAT language offers expressive constructs for specifying and verifying network functionality
• Formal methods are ready for prime time!

Ongoing Work

• Probabilistic semantics
• Stateful functions
• Multi-packet properties
Thank you!

- Carolyn Anderson (UMass)
- Pavol Cerny (Colorado)
- Arjun Guha (UMass)
- Jean-Baptiste Jeannin (CMU)
- Dexter Kozen (Cornell)
- Jedidiah McClurg (Colorado)
- Matthew Milano (Cornell)
- Mark Reitblatt (Cornell)
- Jennifer Rexford (Princeton)
- Cole Schlesinger (Princeton)
- Alexandra Silva (Nijmegen/UCL)
- Steffen Smolka (Cornell)
- Laure Thompson (Cornell)
- Dave Walker (Princeton)

http://frenetic-lang.org/