CPSC 509: Programming Language Principles
Class presentation by David Johnson, Rodrigo Araujo and Nodir Kodirov

NetKAT: Semantic Foundations for Networks

By Carolyn Jane Anderson et al. at POPL’14
Symposium on Principles of Programming Languages

December 2, 2016
Why a cat?
Why a cat? Why a network?

- Networks are cool :-)

Facebook data center at Altoona, Iowa
Facebook data center

- Around **90K** servers
- Up to **10 Gbps** point-to-point
- **7.68 Tbps** uplink
- Non-trivial question
  - Can server [a] talk to [b]?
Why NetKAT?

- Linguistic approach to reason about end-to-end network behaviour
- Relates to class: Kleene stars from hw2 \( \text{stars}(\text{go}^*\text{al}) \)
- And many other concepts
  - denotational/axiomatic semantics
  - equational axioms/reasoning
  - properties of the program
- A grand theme
  - the structure of your definitions guides the structure of your reasoning
Big picture: how is NetKAT used?

Network admin intent:
- can host [a] send packets to host [b]?
- drop all SSH traffic from [a] to [b]

$in \cdot (p \cdot t)^* \cdot out$

Prove soundness and completeness

OpenFlow rule to configure network

Firewall
Contents

- **Rodrigo**: informal description to NetKAT and its constructs
- **David**: formal description
  - syntax, semantics, axioms, equational theory
- **Nodir**: put formal constructs to work
  - Prove soundness of NetKAT reachability equation
Network as an automaton to move packets

- Automaton: move packets from node to node along the links in topology
- PL people: use regular expressions: the language of finite automata

Iterative process: \((p \cdot t)^*\)
Network as an automaton to move packets

- Automaton: move packets from node to node along the links in topology
- PL people: use regular expressions: the language of finite automata

Iterative process: \((p \cdot t)^*\)

- This modelling allows to use Kleene Algebra (KA) to reason about network properties formally
- KA: decades-old sounds and complete equational theory of regular exp.
Now we have KA to reason about network structure (global behavior)
What about individual network components (switch)?

Switch

Predicate: is this SSH traffic?
Action: if yes drop else forward
Network (as a collection of) predicates and actions

- Now we have **KA** to reason about network structure (global behavior)
- What about individual network components (switch)?

- Hence we use
  - **Kleene Algebra**: for reasoning about network structure
  - **Boolean Algebra**: for reasoning about predicates that define switch behaviour

- These two are unified in **Kleene algebra with tests (KAT)** [3]
NetKAT syntax and semantics

- Example: suppose we want to implement two policies
  - Forwarding
  - Access Control
NetKAT syntax and semantics

- **Policies**: function from packets to sets of packets. Used to **filter and modify packets**
- **Policy combinators**
  - The **union combinator** \((p + q)\) generates the union of the sets produced by applying each of \(p\) and \(q\) to the input packet
  - The **sequential composition combinator** \((p \cdot q)\) applies \(p\) to the input packet, then applies \(q\) to each packet in the resulting set, and takes the union of all of the resulting sets
- **Armed with it, we can implement the forwarding policy**
NetKAT example: forwarding

- Packet is represented as a record with fields for standard headers such as
  - source address \((src)\)
  - destination address \((dst)\)
  - protocol type \((typ)\)

- And two fields that identify the current location of the packet in the network
  - switch \((sw)\)
  - port \((pt)\)
NetKAT example: forwarding

- A filter $f \equiv n$ takes any input packet $pk$ and yields the singleton set $\{pk\}$ if field $f$ of $pk$ equals $n$, and $\emptyset$ otherwise.

- A modification $(f \leftarrow n)$ takes any input packet $pk$ and yields the singleton set $\{pk'\}$ where $pk'$ is the packet obtained from $pk$ by setting $f$ to $n$. 
NetKAT example: forwarding

- We can define forwarding as

\[ p \triangleq (\text{dst} = H_1 \cdot pt \leftarrow 1) + (\text{dst} = H_2 \cdot pt \leftarrow 2) \]
NetKAT example: access control (AC)

- A policy that will block SSH traffic

\[ p_{AC} \triangleq \neg (typ = SSH) \cdot p \]
NetKAT example: access control (AC)

- A policy that will block SSH traffic
  \[ p_{AC} \triangleq \neg(typ = SSH) \cdot p \]

- Blocking only on Switch A
  \[ p_A \triangleq (sw = A \cdot \neg(typ = SSH) \cdot p) + (sw = B \cdot p) \]
NetKAT example: access control (AC)

- A policy that will block SSH traffic
  \[ p_{AC} \triangleq \neg (\text{typ} = \text{SSH}) \cdot p \]

- Blocking only on Switch A
  \[ p_A \triangleq (sw = A \cdot \neg (\text{typ} = \text{SSH}) \cdot p) + (sw = B \cdot p) \]

- Blocking only on Switch B
  \[ p_B \triangleq (sw = A \cdot p) + (sw = B \cdot \neg (\text{typ} = \text{SSH}) \cdot p) \]
Topography in NetKAT

- How do we answer questions about the network?
  - Are non-SSH packets forwarded?
  - Are SSH packets dropped?
  - Are $p_A$, $p_A'$, and $p_B$ equivalent?

- Is inspecting the policies enough?

$$p_{AC} \triangleq \neg (typ = SSH) \cdot p$$

$$p_A \triangleq (sw = A \cdot \neg (typ = SSH) \cdot p) + (sw = B \cdot p)$$

$$p_B \triangleq (sw = A \cdot p) + (sw = B \cdot \neg (typ = SSH) \cdot p)$$
Topology in NetKAT

- How do we answer questions about the network?
  - Are non-SSH packets forwarded?
  - Are SSH packets dropped?
  - Are $p_{AC}$, $p_A$, and $p_B$ equivalent?
- Is inspecting the policies enough?
- No! The answers depend fundamentally on the network topology.

\[
p_{AC} \triangleq \neg (typ = SSH) \cdot p
\]
\[
p_A \triangleq (sw = A \cdot \neg (typ = SSH) \cdot p) + (sw = B \cdot p)
\]
\[
p_B \triangleq (sw = A \cdot p) + (sw = B \cdot \neg (typ = SSH) \cdot p)
\]
Topology in NetKAT

- A network topology is a directed graph with hosts and switches as nodes and links as edges
- Links are unidirectional
- Bidirectional links are pair of unidirectional links
A network topology is a directed graph with hosts and switches as nodes and links as edges.

- Links are unidirectional.
- Bidirectional links are a pair of unidirectional links.
- The following policy models the internal links between switches A and B, and the links at the perimeter to hosts 1 and 2:

\[
t = (\text{sw} = A \cdot \text{pt} = 2 \cdot \text{sw} \leftarrow B \cdot \text{pt} \leftarrow 1) + \\
(\text{sw} = B \cdot \text{pt} = 1 \cdot \text{sw} \leftarrow A \cdot \text{pt} \leftarrow 2) + \\
(\text{sw} = A \cdot \text{pt} = 1) + \\
(\text{sw} = B \cdot \text{pt} = 2)
\]
Topology in NetKAT

- If host 1 sends a non-SSH packet to host 2, it is first processed by switch A, then the link between A and B, and finally by switch B.

- NetKAT expression: $p_{AC} \cdot t \cdot p_{AC}$

- We can generalize the global behavior by using Kleene Star: $(p_{AC} \cdot t)^*$
It is often useful to restrict attention to packets that enter and exit the network at specified external locations $e$

$$e \triangleq (sw = A \cdot pt = 1) + (sw = B \cdot pt = 2)$$
Topology in NetKAT

- It is often useful to restrict attention to packets that enter and exit the network at specified external locations $e$

\[
e \triangleq (sw = A \cdot pt = 1) + (sw = B \cdot pt = 2)
\]

- Restrict the policy to packets sent or received by one of the hosts

\[
p_{net} \triangleq e \cdot (p_{AC} \cdot t)^* \cdot e
\]
Topography in NetKAT

- More generally, the input and output predicates may be distinct

\[ \text{in} \cdot (p \cdot t)^* \cdot \text{out} \]

- We call a network modeled in this way a **logical crossbar**, since it encodes end-to-end processing behavior
A packet $pk$ is a record with fields $f_1 \ldots f_k$ mapping to fixed-width integers $n$.

- Assume finite set of *packet headers* including Ethernet source and destination addresses, VLAN tag, IP source and destination addresses, TCP and UDP source and destination ports
Preliminaries: What is our notation?

- A packet $pk$ is a record with fields $f_1...f_k$ mapping to fixed-width integers $n$.
- Assume finite set of packet headers including Ethernet source and destination addresses, VLAN tag, IP source and destination addresses, TCP and UDP source and destination ports
- Include special fields for switch (sw) port (pt) and payload.
- Write $pk.f$ for value in field $f$ of $pk$, and $pk [f := n]$ for the packet obtained from $pk$ by updating field $f$ to $n$. 

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>IP</th>
<th>TCP</th>
<th>SW</th>
<th>PT</th>
<th>Payload</th>
</tr>
</thead>
</table>

### Preliminaries: Packet Histories

- Packet history records the state of each packet as it travels from switch to switch.
- A packet history $h$ is a non-empty sequence of packets.
- We write $pk::<>$ to denote a history with one element, $pk::h$ to denote the history constructed by prepending $pk$ on to $h$, and $<pk_1, \ldots, pk_n>$ for the history with elements $pk_1$ to $pk_n$.
- We write $H$ for the set of all histories, and $\mathcal{P}(H)$ for the powerset of $H$.

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>IP</th>
<th>TCP</th>
<th>SW</th>
<th>PT</th>
<th>Payload</th>
</tr>
</thead>
</table>


Syntax: Predicates & Policies

Predicates
\[ a, b ::= 1 \]
\[ 0 \]
\[ f = n \]
\[ a + b \]
\[ a \cdot b \]
\[ \neg a \]

Identity
Drop
Test
Disjunction
Conjunction
Negation

Policies
\[ p, q ::= a \]
\[ f \leftarrow n \]
\[ p + q \]
\[ p \cdot q \]
\[ p^* \]
\[ \text{dup} \]

Filter
Modification
Union
Sequential composition
Kleene star
Duplication
Semantics

- Every NetKAT predicate and policy denotes a function that takes history $h$ and produces set of histories $\{h_1, \ldots, h_n\}$
- The **empty set** models **dropping the packet** (and its history)
- **Singleton** models modifying or forwarding the packet to a **single location**
- A set with **multiple histories** models modifying the packet in several ways or forwarding the packet to **multiple locations**
Equational Theory: Axioms

**Kleene Algebra Axioms**

\[
\begin{align*}
    p + (q + r) &\equiv (p + q) + r \\
    p + q &\equiv q + p \\
    p + 0 &\equiv p \\
    p + p &\equiv p \\
    p \cdot (q \cdot r) &\equiv (p \cdot q) \cdot r \\
    1 \cdot p &\equiv p \\
    p \cdot 1 &\equiv p \\
    p \cdot (q + r) &\equiv p \cdot q + p \cdot r \\
    (p + q) \cdot r &\equiv p \cdot r + q \cdot r \\
    0 \cdot p &\equiv 0 \\
    p \cdot 0 &\equiv 0 \\
    1 + p \cdot p^* &\equiv p^* \\
    q + p \cdot r &\leq r \Rightarrow p \cdot q \leq r \\
    1 + p^* \cdot p &\equiv p^* \\
    p + q \cdot r &\leq q \Rightarrow p \cdot r^* \leq q
\end{align*}
\]

**KA-PLUS-ASSOC**

**KA-PLUS-COMM**

**KA-PLUS-ZERO**

**KA-PLUS-IDEM**

**KA-SEQ-ASSOC**

**KA-ONE-SEQ**

**KA-SEQ-ONE**

**KA-SEQ-DIST-L**

**KA-SEQ-DIST-R**

**KA-ZERO-SEQ**

**KA-SEQ-ZERO**

**KA-UNROLL-L**

**KA-LFP-L**

**KA-UNROLL-R**

**KA-LFP-R**

**Additional Boolean Algebra Axioms**

\[
\begin{align*}
    a + (b \cdot c) &\equiv (a + b) \cdot (a + c) \\
    a + 1 &\equiv 1 \\
    a + \neg a &\equiv 1 \\
    a \cdot b &\equiv b \cdot a \\
    a \cdot \neg a &\equiv 0 \\
    a \cdot a &\equiv a
\end{align*}
\]

**BA-PLUS-DIST**

**BA-PLUS-ONE**

**BA-EXCL-MID**

**BA-SEQ-COMM**

**BA-CONTRA**

**BA-SEQ-IDEM**
Equational Theory: Axioms

**Kleene Algebra Axioms**

\[
\begin{align*}
    p + (q + r) & \equiv (p + q) + r & \text{KA-PLUS-ASSOC} \\
p + q & \equiv q + p & \text{KA-PLUS-COMM} \\
p + 0 & \equiv p & \text{KA-PLUS-ZERO} \\
p + p & \equiv p & \text{KA-PLUS-IDEM} \\
p \cdot (q \cdot r) & \equiv (p \cdot q) \cdot r & \text{KA-SEQ-ASSOC} \\
1 \cdot p & \equiv p & \text{KA-ONE-SEQ} \\
p \cdot 1 & \equiv p & \text{KA-SEQ-ONE} \\
p \cdot (q + r) & \equiv p \cdot q + p \cdot r & \text{KA-SEQ-DIST-L} \\
(p + q) \cdot r & \equiv p \cdot r + q \cdot r & \text{KA-SEQ-DIST-R} \\
0 \cdot p & \equiv 0 & \text{KA-ZERO-SEQ} \\
p \cdot 0 & \equiv 0 & \text{KA-SEQ-ZERO} \\
1 + p \cdot p^* & \equiv p^* & \text{KA-UNROLL-L} \\
q + p \cdot r \leq r & \Rightarrow p \cdot q \leq r & \text{KA-LFP-L} \\
1 + p^* \cdot p & \equiv p^* & \text{KA-UNROLL-R} \\
p + q \cdot r \leq q & \Rightarrow p \cdot r^* \leq q & \text{KA-LFP-R}
\end{align*}
\]

**Additional Boolean Algebra Axioms**

\[
\begin{align*}
    a + (b \cdot c) & \equiv (a + b) \cdot (a + c) & \text{BA-PLUS-DIST} \\
a + 1 & \equiv 1 & \text{BA-PLUS-ONE} \\
a + \neg a & \equiv 1 & \text{BA-EXCL-MID} \\
a \cdot b & \equiv b \cdot a & \text{BA-SEQ-COMM} \\
a \cdot \neg a & \equiv 0 & \text{BA-CONTRA} \\
a \cdot a & \equiv a & \text{BA-SEQ-IDEM}
\end{align*}
\]
Equational Theory: Axioms

**KAT Theorems**

- **KAT-INVARIANT**: If $a \cdot p \equiv p \cdot a$ then $a \cdot p^* \equiv a \cdot (p \cdot a)^*$
- **KAT-SLIDING**: $p \cdot (q \cdot p)^* \equiv (p \cdot q)^* \cdot p$
- **KAT-DENESTING**: $p^* \cdot (q \cdot p^*)^* \equiv (p + q)^*$
- **KAT-COMMUTE**: If for all atomic $x$ in $q$, $x \cdot p \equiv p \cdot x$ then $q \cdot p \equiv p \cdot q$
Equational Theory: Axioms

Packet Algebra Axioms

\[ f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad \text{PA-MOD-MOD-COMM} \]

\[ f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad \text{PA-MOD-FILTER-COMM} \]

\[ \text{dup} \cdot f = n \equiv f = n \cdot \text{dup} \quad \text{PA-DUP-FILTER-COMM} \]
Equational Theory: Axioms

Packet Algebra Axioms

\[ f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad \text{PA-MOD-MOD-COMM} \]

\[ f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad \text{PA-MOD-FILTER-COMM} \]

\[ \text{dup} \cdot f = n \equiv f = n \cdot \text{dup} \quad \text{PA-DUP-FILTER-COMM} \]

\[ f \leftarrow n \cdot f = n \equiv f \leftarrow n \quad \text{PA-MOD-FILTER} \]
Equational Theory: Axioms

Packet Algebra Axioms

\[ f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \]  
\[ f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \]  
\[ \text{dup} \cdot f = n \equiv f = n \cdot \text{dup} \]  
\[ f \leftarrow n \cdot f = n \equiv f \leftarrow n \]  
\[ f = n \cdot f \leftarrow n \equiv f = n \]  

PA-MOD-MOD-COMM
PA-MOD-FILTER-COMM
PA-DUP-FILTER-COMM
PA-MOD-FILTER
PA-FILTER-MOD
Equational Theory: Axioms

Packet Algebra Axioms

\[
\begin{align*}
    f & \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' & \text{PA-MOD-MOD-COMM} \\
    f & \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' & \text{PA-MOD-FILTER-COMM} \\
    \text{dup} \cdot f = n & \equiv f = n \cdot \text{dup} & \text{PA-DUP-FILTER-COMM} \\
    f & \leftarrow n \cdot f = n \equiv f \leftarrow n & \text{PA-MOD-FILTER} \\
    f & = n \cdot f \leftarrow n \equiv f = n & \text{PA-FILTER-MOD} \\
    f & \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n' & \text{PA-MOD-MOD}
\end{align*}
\]
Equational Theory: Axioms

Packet Algebra Axioms

\[ f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad \text{PA-MOD-MOD-COMM} \]
\[ f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \quad \text{PA-MOD-FILTER-COMM} \]
\[ \text{dup} \cdot f = n \equiv f = n \cdot \text{dup} \quad \text{PA-DUP-FILTER-COMM} \]
\[ f \leftarrow n \cdot f = n \equiv f \leftarrow n \quad \text{PA-MOD-FILTER} \]
\[ f = n \cdot f \leftarrow n \equiv f = n \quad \text{PA-FILTER-MOD} \]
\[ f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n' \quad \text{PA-MOD-MOD} \]
\[ f = n \cdot f = n' \equiv 0, \text{ if } n \neq n' \quad \text{PA-CONTRA} \]
Equational Theory: Axioms

Packet Algebra Axioms

\[
\begin{align*}
  & f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \\
  & f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f' \\
  & \text{dup} \cdot f = n \equiv f = n \cdot \text{dup} \\
  & f \leftarrow n \cdot f = n \equiv f \leftarrow n \\
  & f = n \cdot f \leftarrow n \equiv f = n \\
  & f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n' \\
  & f = n \cdot f = n' \equiv 0, \text{ if } n \neq n' \\
  & \sum_{i} f = i \equiv 1
\end{align*}
\]
Equational Theory Example: Access Control

- Policy $P_A$ filters SSH packets on switch A while $P_B$ filters SSH packets on switch B

- We can prove these are equivalent on SSH traffic going to left to right across our topology

- This is a simple form of code motion - relocating the filter from A to B
Equational Theory Example: Access Control

- Policy $P_A$ filters SSH packets on switch A while $P_B$ filters SSH packets on switch B
- We can prove these are equivalent on SSH traffic going to left to right across our topology
- This is a simple form of code motion - relocating the filter from A to B
- The first lemma of the proof shows sequencing a predicate that matches switch A with a predicate that matches switch B will drop all packets
Equational Theory Example: Access Control

- We use the logical crossbar encoding with predicates

\[
\begin{align*}
\text{in} & \triangleq (\text{sw} = A \cdot \text{pt} = 1) \\
\text{out} & \triangleq (\text{sw} = B \cdot \text{pt} = 2)
\end{align*}
\]

\[
\begin{align*}
a_A & \triangleq (\text{sw} = A) \\
\text{a}_B & \triangleq (\text{sw} = B) \\
a_1 & \triangleq (\text{pt} = 1) \\
\text{a}_2 & \triangleq (\text{pt} = 2)
\end{align*}
\]
Equational Theory Example: Access Control

\[ \text{Lemma 1. } \quad \mathit{in} \cdot \mathit{a_B} \cdot q \equiv 0 \]

\[ \text{Proof. } \]

\[ \begin{align*}
\mathit{in} \cdot \mathit{a_B} \cdot q & \equiv \{ \text{definition } \mathit{in} \} \\
& \equiv \{ \text{KAT-COMMUTE} \} \\
& \equiv \{ \text{PA-CONTRA} \} \\
& \equiv \{ \text{KA-ZERO-SEQ} \} \\
& 0 \cdot a_1 \cdot q \\
\end{align*} \]

\[ \square \]

\[ \begin{align*}
\mathit{in} & \triangleq (\mathit{sw} = A \cdot \mathit{pt} = 1) \\
\mathit{out} & \triangleq (\mathit{sw} = B \cdot \mathit{pt} = 2) \\
\mathit{a_A} & \triangleq (\mathit{sw} = A) \\
\mathit{a_B} & \triangleq (\mathit{sw} = B) \\
a_1 & \triangleq (\mathit{pt} = 1) \\
a_2 & \triangleq (\mathit{pt} = 2) \\
\end{align*} \]
Equational Theory Example: Access Control

- Next, we’ll see lemma 2 of the proof
- Lemma 2 proves sequential composition of an arbitrary policy $q$, the predicate $a_A$, topology $t$, and an output predicate is equivalent to the policy that drops all packets
Equational Theory Example: Access Control

**Lemma 2.** \( q \cdot a_A \cdot t \cdot out \equiv 0 \)

**Proof.**
\[
q \cdot a_A \cdot t \cdot out \\
\equiv \{ \text{definition } t \} \\
q \cdot a_A \cdot (a_A \cdot a_2 \cdot m_B \cdot m_1 + a_B \cdot a_1 \cdot m_A \cdot m_2 + a_A \cdot a_1 + a_B \cdot a_2) \cdot out \\
\equiv \{ \text{KA-SEQ-DIST-L, KA-SEQ-DIST-R} \} \\
q \cdot a_A \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot a_1 \cdot a_B \cdot a_2 + q \cdot a_A \cdot a_B \cdot a_1 \cdot m_B \cdot m_2 \cdot a_B \cdot a_2 + q \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot a_2 + q \cdot a_A \cdot a_B \cdot a_2 \cdot a_B \cdot a_2 \\
\equiv \{ \text{KA-SEQ-ZERO, KA-ZERO-SEQ} \} \\
0 + 0 + 0 + 0 + 0 \\
\equiv \{ \text{KA-PLUS-IDEDEM} \} \\
0
\]
Equational Theory Example: Access Control

- Finally, we’ll see lemma 3 of the proof
- Lemma 3 proves $P_A$ and $P_B$ both drop SSH traffic going from host 1 to host 2
Lemma 3. $\text{in} \cdot \text{SSH} \cdot (p_A \cdot t)^* \cdot \text{out} \equiv \text{in} \cdot \text{SSH} \cdot (p_B \cdot t)^* \cdot \text{out}$

Proof.

\[
\text{in} \cdot \text{SSH} \cdot (p_A \cdot t)^* \cdot \text{out} \\
\equiv \{ \text{KAT-INARIANT, definition } p_A \} \\
\text{in} \cdot \text{SSH} \cdot ((a_A \cdot \neg \text{SSH} \cdot p + a_B \cdot p) \cdot t \cdot \text{SSH})^* \cdot \text{out} \\
\equiv \{ \text{KA-SEQ-DIST-R} \} \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot \neg \text{SSH} \cdot p \cdot t \cdot \text{SSH} + a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot \text{out} \\
\equiv \{ \text{KAT-COMMUTE} \} \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot \neg \text{SSH} \cdot \text{SSH} \cdot p \cdot t + a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot \text{out} \\
\equiv \{ \text{BA-CONTRA} \} \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot 0 \cdot p \cdot t + a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot \text{out} \\
\equiv \{ \text{KA-SEQ-ZERO/SEQ-DIST-L, KA-PLUS-COMM, KA-PLUS-ZERO} \} \\
\text{in} \cdot \text{SSH} \cdot (a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot \text{out} \\
\equiv \{ \text{KA-UNROLL-L} \} \\
\text{in} \cdot \text{SSH} \cdot (1 + (a_B \cdot p \cdot t \cdot \text{SSH} \cdot (a_B \cdot p \cdot t \cdot \text{SSH})^*) \cdot \text{out} \\
\equiv \{ \text{KA-SEQ-DIST-L, KA-SEQ-DIST-R, definition } out \} \\
\text{in} \cdot \text{SSH} \cdot a_B \cdot a_2 + \\
\text{in} \cdot \text{SSH} \cdot a_B \cdot p \cdot t \cdot \text{SSH} \cdot (a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot a_B \cdot a_2 \\
\equiv \{ \text{KAT-COMMUTE} \} \\
\text{in} \cdot a_B \cdot \text{SSH} \cdot a_2 + \\
\text{in} \cdot a_B \cdot \text{SSH} \cdot p \cdot t \cdot \text{SSH} \cdot (a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot a_B \cdot a_2 \\
\equiv \{ \text{Lemma 1} \} \\
0 + 0 \\
\equiv \{ \text{KA-PLUS-IDEM} \} \\
0 \\
\equiv \{ \text{KA-PLUS-IDEM} \} \\
0 + 0 \\
\equiv \{ \text{Lemma 1, Lemma 2} \} \\
\text{in} \cdot a_B \cdot \text{SSH} \cdot a_2 + \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH})^* \cdot p \cdot \text{SSH} \cdot a_A \cdot t \cdot \text{SSH} \\
\equiv \{ \text{KAT-COMMUTE, definition } out \} \\
\text{in} \cdot \text{SSH} \cdot out + \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH})^* \cdot a_A \cdot p \cdot t \cdot \text{SSH} \cdot out \\
\equiv \{ \text{KA-SEQ-DIST-L, KA-SEQ-DIST-R} \} \\
\text{in} \cdot \text{SSH} \cdot (1 + (a_A \cdot p \cdot t \cdot \text{SSH})^* \cdot (a_A \cdot p \cdot t \cdot \text{SSH})) \cdot out \\
\equiv \{ \text{KA-UNROLL-R} \} \\
\text{in} \cdot \text{SSH} \cdot (a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv \{ \text{KA-SEQ-ZERO/SEQ-DIST-L, KA-PLUS-ZERO} \} \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH} + a_B \cdot 0 \cdot p \cdot t)^* \cdot out \\
\equiv \{ \text{BA-CONTRA} \} \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH} + a_B \cdot \neg \text{SSH} \cdot \text{SSH} \cdot p \cdot t)^* \cdot out \\
\equiv \{ \text{KAT-COMMUTE} \} \\
\text{in} \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH} + a_B \cdot \neg \text{SSH} \cdot \text{SSH} \cdot p \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv \{ \text{KA-SEQ-DIST-R} \} \\
\text{in} \cdot \text{SSH} \cdot ((a_A \cdot p + a_B \cdot \neg \text{SSH} \cdot p) \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv \{ \text{KAT-INARIANT, definition } p_B \} \\
\text{in} \cdot \text{SSH} \cdot (p_B \cdot t)^* \cdot out \\
\square
NetKAT at work: **useful properties**

- **Reachability properties**
  - Can host [a] send packets to host [b]?

- **Traffic isolation**
  - Policies for particular network traffic does not impact other traffic

- **Compiler correctness**
  - Ensure NetKAT policies correctly translated to network rules
Reachability: some interesting questions

- Can host [a] send packets to host [b]?
Reachability: some interesting questions

- Can host [a] send packets to host [b]?

- Are managed hosts kept separate from unmanaged hosts?
Reachability: some interesting questions

- Can host [a] send packets to host [b]?

- Are managed hosts kept separate from unmanaged hosts?

- Does all untrusted traffic traverse the intrusion detection system (IDS)?
Reachability: some interesting questions

- Can host [a] send packets to host [b]?
- Are managed hosts kept separate from unmanaged hosts?
- Does all untrusted traffic traverse the intrusion detection system (IDS)?
Reachability: can host [a] send packets to host [b]?
Reachability: can host [a] send packets to host [b]?
Reachability: can host [a] send packets to host [b]?
Reachability: can host [a] send packets to host [b]?

\[ \text{in} \cdot (p \cdot t)^* \cdot \text{out} \]

Behaviour of an entire network (crossbar model)
Reachability: can host [a] send packets to host [b]?

\[ in \cdot (p \cdot t)^* \cdot out \]

Behaviour of an entire network (crossbar model)

\[ in \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot out \]

dup records a packet and lets us reason about behaviour of each individual hop
Reachability: can host [a] send packets to host [b]?

\[
\text{in} \cdot (p \cdot t)^* \cdot \text{out}
\]

Behaviour of an entire network (crossbar model)

\[
\text{in} \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot \text{out}
\]

dup records a packet and lets us reason about behaviour of each individual hop

\[
a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0
\]

prepending a filters packets with source [a] and b filters packets with destination [b]
Reachability: can host [a] send packets to host [b]?

How do we know that this is correct?

\[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0 \]

prepending \( a \) filters packets with source \([a]\) and \( b \) filters packets with destination \([b]\)
Reachability: can host [a] send packets to host [b]?

- Prove correctness
- **Define reachability**: show semantic notion
- **Translate**
  - denotational semantics of reachability, and
  - below equation into the language model
- Equations are easily related to one another in the language model

\[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0 \]

prepending \( a \) filters packets with source [a] and \( b \) filters packets with destination [b]
NetKAT language model

Reduced NetKAT syntax

Set of complete assignments: \[ \pi \triangleq f_1 \leftarrow n_1 \cdots f_k \leftarrow n_k \]

Set of complete tests: \[ \alpha, \beta \triangleq f_1 = n_1 \cdots f_k = n_k \]

Reduced terms: \[ p, q ::= \alpha \]

Complete test: \[ \pi \]

Complete assignment: \[ p + q \]

Sequence: \[ p \cdot q \]

Kleene star: \[ p^* \]

Duplication: \[ \text{dup} \]

Policies

Filter

Modification

Union

Sequential composition

Kleene star

Duplication

Packet Algebra Axioms

\[ f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f' \]

\[ f \leftarrow n \cdot f' = n' \leftarrow f = n' \cdot f \leftarrow n, \text{ if } f \neq f' \]

\[ \text{dup} \cdot f = n \equiv f = n \cdot \text{dup} \]

\[ f \leftarrow n \cdot f = n \equiv f \leftarrow n \]

\[ f = n \cdot f \leftarrow n \equiv f = n \]

\[ f \leftarrow n \cdot f \leftarrow n' \equiv f = n' \cdot f \leftarrow n \]

\[ f = n \cdot f = n' \equiv 0, \text{ if } n \neq n' \]

\[ \sum_{i \leq 1} f = i = 1 \]

Simplified axioms for A and P

1. \[ \pi \equiv \pi \cdot \pi \]
2. \[ \alpha \equiv \alpha \cdot \pi \]
3. \[ \alpha \cdot \text{dup} \equiv \text{dup} \cdot \alpha \]
4. \[ \pi \cdot \pi' \equiv \pi' \]
5. \[ \sum_{\alpha} \alpha \equiv 1, \]
6. \[ \alpha \cdot \beta \equiv 0, \alpha \neq \beta \]

Regular interpretation:

\[ R(p) \subseteq (I + A + \text{dup})^* \]

\[ R(\pi) = \{ \pi \} \]

\[ R(p + q) = R(p) \cup R(q) \]

\[ R(\alpha) = \{ \alpha \} \]

\[ R(p \cdot q) = \{ xy \mid x \in R(p), y \in R(q) \} \]

\[ R(\text{dup}) = \{ \text{dup} \} \]

\[ R(p^*) = \bigcup_{n \geq 0} R(p^n) \]

Set of complete atoms (tests)

Set of complete assignments

63
NetKAT language model

Reduced NetKAT syntax

Complete assignments  \[ \pi \triangleq f_1 \leftarrow n_1 \cdots f_k \leftarrow n_k \]
Complete tests  \[ \alpha, \beta \triangleq f_1 = n_1 \cdots f_k = n_k \]

Reduced terms  \[ p, q ::= \alpha \]
\[ \pi \] Complete assignment
\[ p + q \] Union
\[ p \cdot q \] Sequence
\[ p^* \] Kleene star
\[ \text{dup} \] Duplication

Simplified axioms for \( A \) and \( P \)

\[ \pi \equiv \pi \cdot \pi_{\pi} \]
\[ \alpha \cdot \text{dup} \equiv \text{dup} \cdot \alpha \]
\[ \sum_{\alpha} \alpha \equiv 1, \]
\[ \alpha \equiv \alpha \cdot \pi_{\alpha} \]
\[ \pi \cdot \pi' \equiv \pi' \]
\[ \alpha \cdot \beta \equiv 0, \alpha \neq \beta \]

Regular interpretation: \( R(p) \subseteq (\Pi + A + \text{dup})^* \)

\[ R(\pi) = \{ \pi \} \]
\[ R(p + q) = R(p) \cup R(q) \]
\[ R(\alpha) = \{ \alpha \} \]
\[ R(p \cdot q) = \{ xy \mid x \in R(p), y \in R(q) \} \]
\[ R(\text{dup}) = \{ \text{dup} \} \]
\[ R(p^*) = \bigcup_{n \geq 0} R(p^n) \]

Language model: \( G(p) \subseteq I = A \cdot (\Pi \cdot \text{dup})^* \cdot \Pi \)

\[ G(\pi) = \{ \alpha \cdot \pi \mid \alpha \in A \} \]
\[ G(p + q) = G(p) \cup G(q) \]
\[ G(\alpha) = \{ \alpha \cdot \pi_{\alpha} \} \]
\[ G(p \cdot q) = G(p) \circ G(q) \]
\[ G(\text{dup}) = \{ \alpha \cdot \pi_{\alpha} \cdot \text{dup} \cdot \pi_{\alpha} \mid \alpha \in A \} \]
\[ G(p^*) = \bigcup_{n \geq 0} G(p^n) \]

Guarded concatenation

\[ \alpha \cdot p \cdot \pi \circ \beta \cdot q \cdot \pi' = \begin{cases} 
\alpha \cdot p \cdot q \cdot \pi' & \text{if } \beta = \pi_{\alpha} \\
\text{undefined} & \text{if } \beta \neq \pi_{\alpha} 
\end{cases} \]

\[ A \circ B = \{ p \circ q \mid p \in A, q \in B \} \subseteq I \]
Reachability: can host [a] send packets to host [b]?  

**Definition 2 (Reachability).** We say $b$ is reachable from $a$ if and only if there exists a trace 

$$\langle pk_1, \ldots, pk_n \rangle \in \text{rng}([\text{dup} \cdot (p \cdot t \cdot \text{dup})^*])$$  

such that $[a] \langle pk_n \rangle = \{\langle pk_n \rangle\}$ and $[b] \langle pk_1 \rangle = \{\langle pk_1 \rangle\}$.  

Intuition: [a] can talk to [a] if there is a trace where packet’s first hop is [a] last hop is [b]
Reachability: can host [a] send packets to host [b]?

Theorem 4 (Reachability Correctness). For predicates $a$ and $b$, policy $p$, and topology $t$, $a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0$, if and only if $b$ is reachable from $a$. 
Reachability: can host [a] send packets to host [b]?

**Theorem 4** (Reachability Correctness). For predicates $a$ and $b$, policy $p$, and topology $t$, $a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0$, if and only if $b$ is reachable from $a$. 
Reachability: can host [a] send packets to host [b]?

Proof. We translate the NetKAT equation into the language model:

\[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \not\equiv 0 \]

\[ \Rightarrow \exists \alpha, \pi_n, \ldots, \pi_1. \]

\[ \alpha \cdot \pi_n \cdot \text{dup} \cdots \text{dup} \cdot \pi_1 \in G(a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b) \]

Theorem 4 (Reachability Correctness). For predicates a and b, policy p, and topology t, \[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \not\equiv 0 \] if and only if b is reachable from a.
Reachability: can host [a] send packets to host [b]?

Proof. We translate the NetKAT equation into the language model:

\[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \not\equiv 0 \]

\[ \Rightarrow \exists \alpha, \pi_n, \ldots, \pi_1. 
\]

\[ \alpha \cdot \pi_n \cdot \text{dup} \cdots \text{dup} \cdot \pi_1 \in G(a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b) \]
Reachability: can host [a] send packets to host [b]?

Also translate each term in the semantic definition of reachability into the language model:

\[ \exists p_{k_1}, \ldots, p_{k_n}. \]
\[ \langle p_{k_1}, \ldots, p_{k_n} \rangle \in \text{rng}(\text{[dup \cdot (p \cdot t \cdot dup)^*]}), \]
\[ [a] \langle p_{k_n} \rangle = \{ \langle p_{k_n} \rangle \} \text{ and } [b] \langle p_1 \rangle = \{ \langle p_1 \rangle \} \]

**Proof.** We translate the NetKAT equation into the language model:

\[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0 \]
\[ \Rightarrow \exists \alpha, \pi_n, \ldots, \pi_1. \]
\[ \alpha \cdot \pi_n \cdot \text{dup} \cdot \ldots \cdot \text{dup} \cdot \pi_1 \in G(a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b) \]

**Definition 2 (Reachability).** We say b is reachable from a if and only if there exists a trace

\[ \langle p_{k_1}, \ldots, p_{k_n} \rangle \in \text{rng}(\text{[dup \cdot (p \cdot t \cdot dup)^*]}) \]
\[ \text{such that } [a] \langle p_{k_n} \rangle = \{ \langle p_{k_n} \rangle \} \text{ and } [b] \langle p_1 \rangle = \{ \langle p_1 \rangle \}. \]
Reachability: can host \([a]\) send packets to host \([b]\)?

Also translate each term in the semantic definition of reachability into the language model:

\[
\exists pk_1, \ldots, pk_n.
\langle pk_1, \ldots, pk_n \rangle \in \text{rng}(\llbracket \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \rrbracket),
\llbracket a \rrbracket \langle pk_n \rangle = \{\langle pk_n \rangle\} \text{ and }
\llbracket b \rrbracket \langle pk_1 \rangle = \{\langle pk_1 \rangle\}
\Rightarrow
\exists \pi'_1, \ldots, \pi'_m.
\alpha_{\pi'_m} \cdot \pi'_m \cdot \text{dup} \cdots \text{dup} \cdot \pi'_1 \in G(\text{dup} \cdot (p \cdot t \cdot \text{dup})^*),
\alpha_{\pi'_m} \cdot \pi'_m \in G(a) \text{ and }
\alpha_{\pi'_1} \cdot \pi'_1 \in G(b)
\]

Proof. We translate the NetKAT equation into the language model:

\[
\exists \alpha, \pi_n, \ldots, \pi_1.
\alpha \cdot \pi_n \cdot \text{dup} \cdots \text{dup} \cdot \pi_1 \in G(\alpha \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b)
\]

Definition 2 (Reachability). We say \(b\) is reachable from \(a\) if and only if there exists a trace

\[
\langle pk_1, \ldots, pk_n \rangle \in \text{rng}(\llbracket \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \rrbracket)
\]

such that \(\llbracket a \rrbracket \langle pk_n \rangle = \{\langle pk_n \rangle\}\) and \(\llbracket b \rrbracket \langle pk_1 \rangle = \{\langle pk_1 \rangle\}\).
Reachability: can host [a] send packets to host [b]?

\[
\exists p_{k_1}, \ldots, p_{k_n}.
\langle p_{k_1}, \ldots, p_{k_n} \rangle \in \text{rng}(\text{dup} \cdot (p \cdot t \cdot \text{dup})^*),
\]
\[
[a] \langle p_{k_n} \rangle = \{ \langle p_{k_n} \rangle \} \quad \text{and} \quad
[b] \langle p_{k_1} \rangle = \{ \langle p_{k_1} \rangle \}
\]

\Rightarrow
\exists \pi'_{m}, \ldots, \pi'_{m}.
\alpha_{\pi'_{m}} \cdot \pi'_{m} \cdot \text{dup} \cdot \pi'_{1} \in G(\text{dup} \cdot (p \cdot t \cdot \text{dup})^*),
\]
\[
\alpha_{\pi'_{m}} \cdot \pi'_{m} \in G(a) \quad \text{and} \quad
\alpha_{\pi'_{1}} \cdot \pi'_{1} \in G(b)
\]

To prove soundness we let \( \alpha = \alpha_{\pi_n} \) and \( m = n \) to show that if
\[
\alpha \cdot \pi_n \cdot \text{dup} \cdots \text{dup} \cdot \pi_1 \in G(a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b)
\]
then,
\[
\alpha_{\pi'_n} \cdot \pi'_m \cdot \text{dup} \cdots \text{dup} \cdot \pi'_1 \in G(\text{dup} \cdot (p \cdot t \cdot \text{dup})^*)
\]

which holds by definition of \( \diamond \). The proof of completeness follows by a similar argument.

\[\square\]

**Proof.** We translate the NetKAT equation into the language model:
\[
a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \not\equiv 0
\]

\[
\Rightarrow \exists \alpha, \pi_n, \ldots, \pi_1.
\alpha \cdot \pi_n \cdot \text{dup} \cdots \text{dup} \cdot \pi_1 \in G(a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b)
\]
To prove correctness

- **Define reachability:** show semantic notion
- **Translate**
  - denotational semantics of reachability, and
  - below equation into the language model
- Show NetKAT equation is equivalent to the reachability definition

\[ a \cdot \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \cdot b \neq 0 \]

**Definition 2** (Reachability). We say \( b \) is reachable from \( a \) if and only if there exists a trace

\[ \langle p_{k_1}, \ldots, p_{k_n} \rangle \in \text{rng}(\langle \text{dup} \cdot (p \cdot t \cdot \text{dup})^* \rangle) \]

such that \( [a] \langle p_{k_n} \rangle = \{ \langle p_{k_n} \rangle \} \) and \( [b] \langle p_{k_1} \rangle = \{ \langle p_{k_1} \rangle \} \).
Takeaways

● Showed how Kleene algebra with tests (KAT) applies to networks
● Formally described NetKAT syntax, semantics, and axioms
● Applied equational theory in NetKAT
● Gave examples of NetKAT equation to
  ○ drop SSH traffic between two nodes
  ○ check reachability between two nodes
● Formally showed correctness of the reachability equation
References

1. NetKAT: semantic foundations for networks, POPL’14 (Symposium on Principles of Programming Languages),
   http://dl.acm.org/citation.cfm?id=2535862
   https://ecommons.cornell.edu/handle/1813/34445