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?

NetKAT: semantic foundations for networks

Full Text: PDF

see source materials below for more options

2014 Article

Authors:

Carolyn Jane Anderson Swarthmore College, Swarthmore, PA, USA
Nate Foster Cornell University, Ithaca, NY, USA
Arjun Guha University of Massachusetts Amherst, Amherst, MA, USA
Jean-Baptiste Jeannin Carnegie Mellon University, Pittsburgh, PA, USA
Dexter Kozen Cornell University, Ithaca, NY, USA
Cole Schlesinger Princeton University, Princeton, NJ, USA
David Walker Princeton University, Princeton, NJ, USA

Why a cat?
Why a network?

NetKAT: semantic foundations for networks

Full Text: PDF

see source materials below for more options

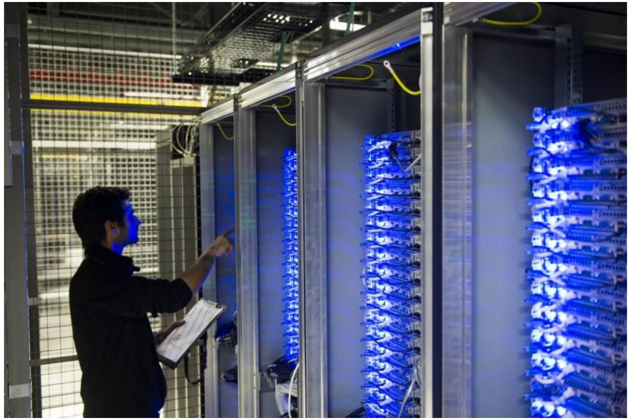
Authors:
- Carolyn Jane Anderson — Swarthmore College, Swarthmore, PA, USA
- Nate Foster — Cornell University, Ithaca, NY, USA
- Arjun Guha — University of Massachusetts Amherst, Amherst, MA, USA
- Jean-Baptiste Jeannin — Carnegie Mellon University, Pittsburgh, PA, USA
- Dexter Kozen — Cornell University, Ithaca, NY, USA
- Cole Schlesinger — Princeton University, Princeton, NJ, USA
- David Walker — Princeton University, Princeton, NJ, USA

2014 Article

Bibliometrics
- Citation Count: 35
- Downloads (cumulative): 712
- Downloads (12 Months): 225
- Downloads (6 Weeks): 28

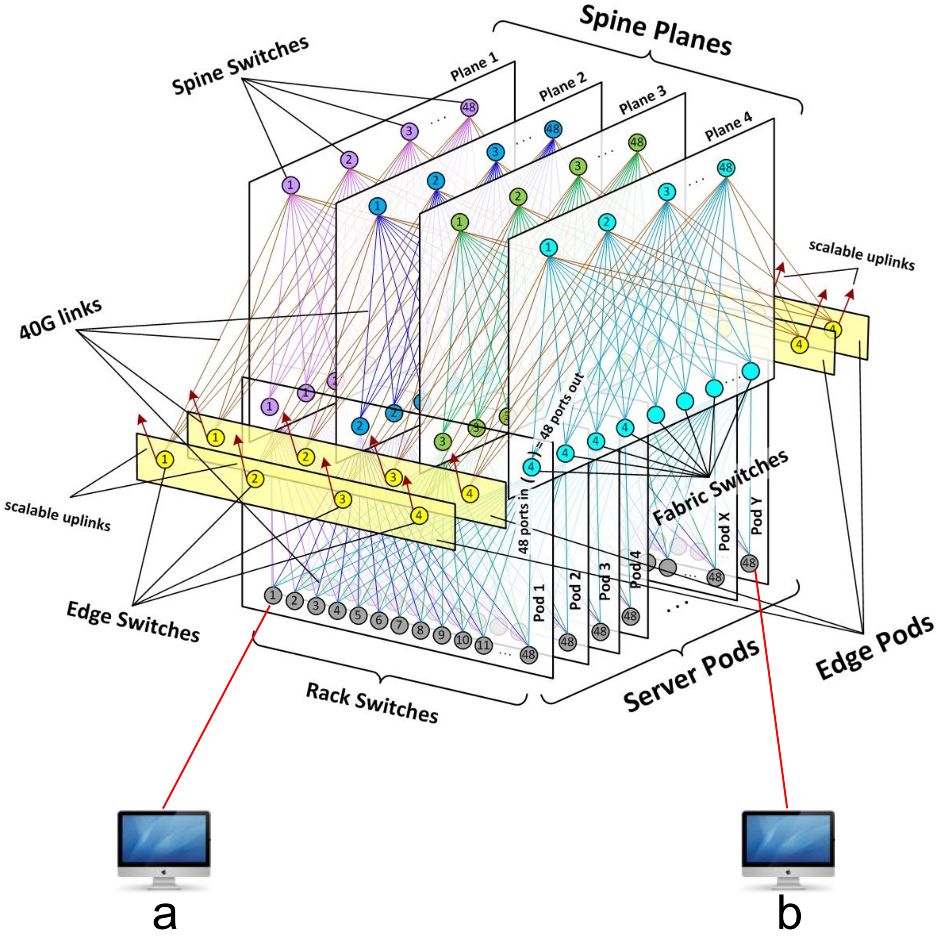
- Networks are cool :-)

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 $stars(\mathsf{g}(\mathsf{oo}*)*\mathsf{al})$
- 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?

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

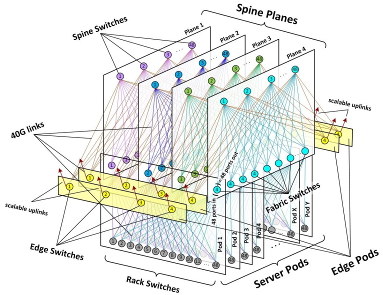
OpenFlow rule to configure network

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

Prove soundness and completeness

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

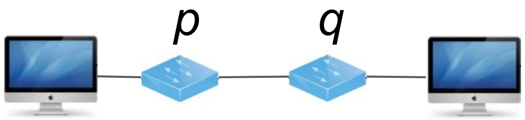


Contents

- 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

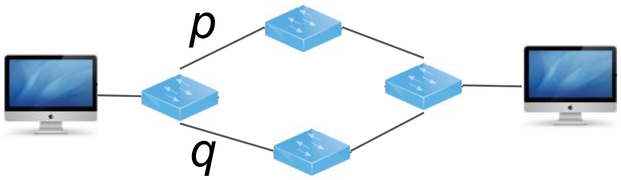
Network as an automaton to move packets

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



$$(p \cdot q \cdots)$$

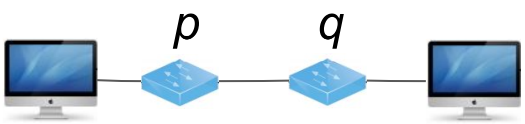
Iterative process: $(p \cdot t)^*$



$$(p + q + \cdots)$$

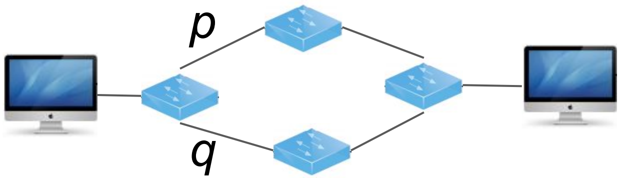
Network as an automaton to move packets

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



$$(p \cdot q \cdots)$$

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



$$(p + q + \cdots)$$

- 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.

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)?

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)?

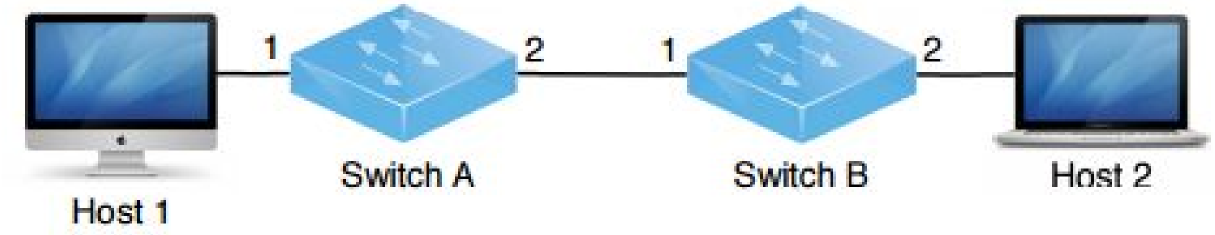
Switch

Predicate: is this SSH traffic?
Action: if yes **drop** else **forward**

- 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

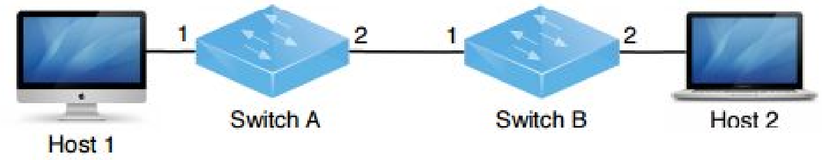


NetKAT syntax and semantics

- 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)*

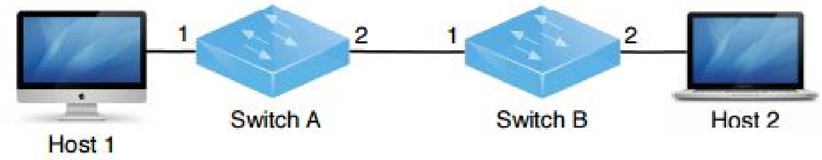


SRC	DST	TYP

SRC	DST	TYP	SW	PT

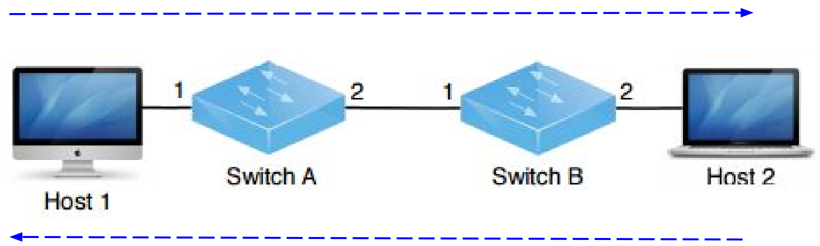
NetKAT example: forwarding

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

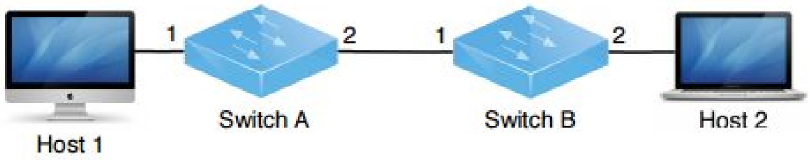


$$p \triangleq (dst = H_1 \cdot pt \leftarrow 1) + (dst = H_2 \cdot pt \leftarrow 2)$$

NetKAT example: access control (AC)

- A policy that will block SSH traffic

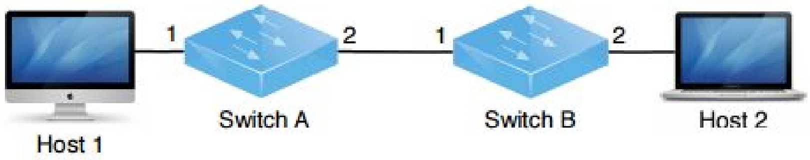
$$p_{AC} \triangleq \neg(typ = SSH) \cdot p$$



NetKAT example: access control (AC)

- A policy that will block SSH traffic

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



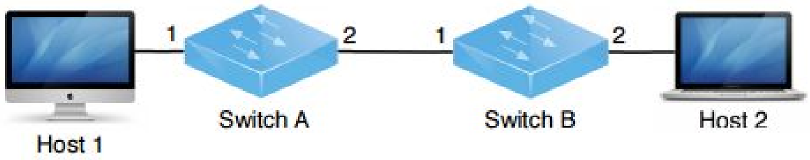
- Blocking only on Switch A

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

NetKAT example: access control (AC)

- A policy that will block SSH traffic

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



- Blocking only on Switch A

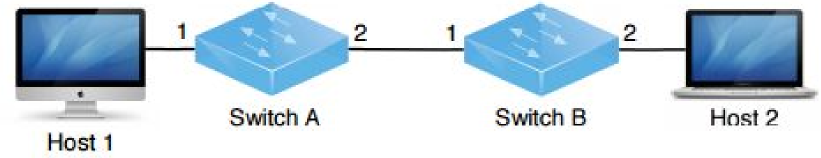
$$p_A \triangleq (sw = A \cdot \neg(typ = SSH) \cdot p) + (sw = B \cdot p)$$

- Blocking only on Switch B

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

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?



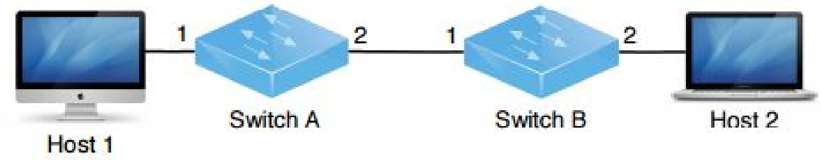
$$p_{AC} \triangleq \neg(typ = SSH) \cdot p$$

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

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

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.



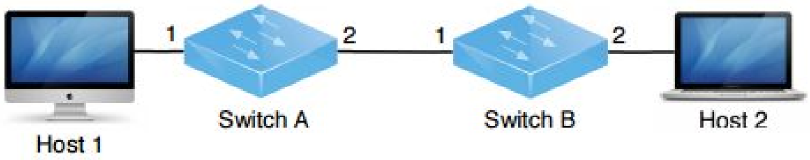
$$p_{AC} \triangleq \neg(typ = SSH) \cdot p$$

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

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

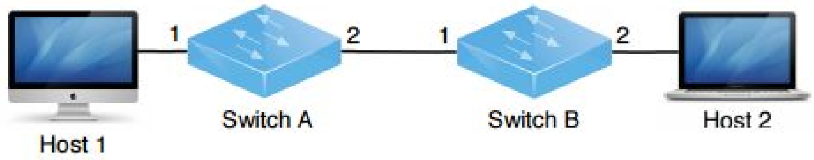
Topology in NetKAT

- 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



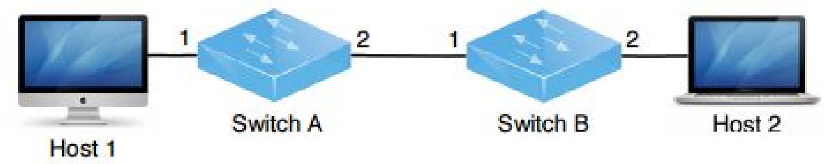
Topology in NetKAT

- 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
- The following policy models the internal links between switches A and B, and the links at the perimeter to hosts 1 and 2



$$
\begin{aligned}
t = \;& (\mathsf{sw} = A \cdot \mathsf{pt} = 2 \cdot \mathsf{sw} \leftarrow B \cdot \mathsf{pt} \leftarrow 1) + \\
& (\mathsf{sw} = B \cdot \mathsf{pt} = 1 \cdot \mathsf{sw} \leftarrow A \cdot \mathsf{pt} \leftarrow 2) + \\
& (\mathsf{sw} = A \cdot \mathsf{pt} = 1) + \\
& (\mathsf{sw} = B \cdot \mathsf{pt} = 2)
\end{aligned}
$$

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

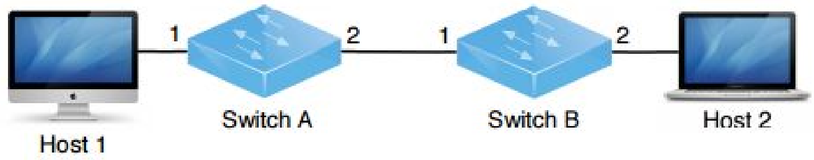
- NetKAT expression $p_{AC} \cdot t \cdot p_{AC}$

- We can generalize the global behavior by using Kleene Star

$$(p_{AC} \cdot t)^*$$

Topology in NetKAT

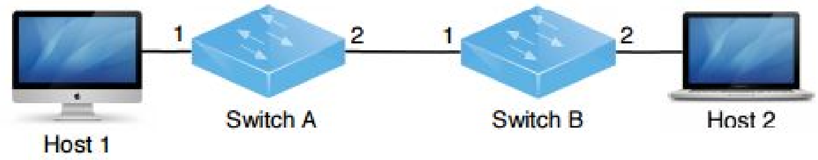
- It is often useful to restrict attention to packets that enter and exit the network at specified external locations e



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

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

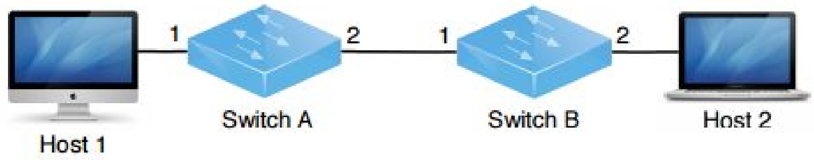
$$p_{net} \triangleq e \cdot (p_{AC} \cdot t)^* \cdot e$$

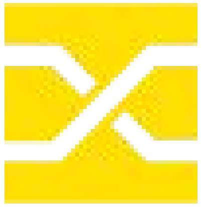
Topology in NetKAT

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

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

- We call a network modeled in this way a logical crossbar, since it encodes end-to-end processing behavior





Logical crossbar

Preliminaries: What is our notation?

Ethernet	IP	TCP

- 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?

Ethernet	IP	TCP	SW	PT	Payload

- 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.*

Preliminaries: Packet Histories

Ethernet	IP	TCP	SW	PT	Payload

- 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 $<pk_1 , . . . , pk_n >$ 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

Syntax: Predicates & Policies

Predicates

$$
\begin{array}{rlll}
a, b ::= & 1 & & \textit{Identity} \\
\mid & 0 & & \textit{Drop} \\
\mid & f = n & & \textit{Test} \\
\mid & a + b & & \textit{Disjunction} \\
\mid & a \cdot b & & \textit{Conjunction} \\
\mid & \neg a & & \textit{Negation}
\end{array}
$$

Policies

$$
\begin{array}{rlll}
p, q ::= & a & & \textit{Filter} \\
\mid & f \leftarrow n & & \textit{Modification} \\
\mid & p + q & & \textit{Union} \\
\mid & p \cdot q & & \textit{Sequential composition} \\
\mid & p^* & & \textit{Kleene star} \\
\mid & \mathsf{dup} & & \textit{Duplication}
\end{array}
$$

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

$$[\![p]\!] \in \mathsf{H} \to \mathcal{P}(\mathsf{H})$$

$$[\![1]\!]\ h \triangleq \{h\}$$

$$[\![0]\!]\ h \triangleq \{\}$$

$$[\![f = n]\!]\ (pk{::}h) \triangleq \begin{cases} \{pk{::}h\} & \text{if } pk.f = n \\ \{\} & \text{otherwise} \end{cases}$$

$$[\![\neg a]\!]\ h \triangleq \{h\} \setminus ([\![a]\!]\ h)$$

$$[\![f \leftarrow n]\!]\ (pk{::}h) \triangleq \{pk[f := n]{::}h\}$$

$$[\![p + q]\!]\ h \triangleq [\![p]\!]\ h \cup [\![q]\!]\ h$$

$$[\![p \cdot q]\!]\ h \triangleq ([\![p]\!] \bullet [\![q]\!])\ h$$

$$[\![p^*]\!]\ h \triangleq \bigcup_{i \in \mathsf{N}} F^i\ h$$

$$\text{where } F^0\ h \triangleq \{h\} \text{ and } F^{i+1}\ h \triangleq ([\![p]\!] \bullet F^i)\ h$$

$$[\![\mathsf{dup}]\!]\ (pk{::}h) \triangleq \{pk{::}(pk{::}h)\}$$

Equational Theory: Axioms

Kleene Algebra Axioms	
$p + (q + r) \equiv (p + q) + r$	KA-PLUS-ASSOC
$p + q \equiv q + p$	KA-PLUS-COMM
$p + 0 \equiv p$	KA-PLUS-ZERO
$p + p \equiv p$	KA-PLUS-IDEM
$p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$	KA-SEQ-ASSOC
$1 \cdot p \equiv p$	KA-ONE-SEQ
$p \cdot 1 \equiv p$	KA-SEQ-ONE
$p \cdot (q + r) \equiv p \cdot q + p \cdot r$	KA-SEQ-DIST-L
$(p + q) \cdot r \equiv p \cdot r + q \cdot r$	KA-SEQ-DIST-R
$0 \cdot p \equiv 0$	KA-ZERO-SEQ
$p \cdot 0 \equiv 0$	KA-SEQ-ZERO
$1 + p \cdot p^* \equiv p^*$	KA-UNROLL-L
$q + p \cdot r \leq r \Rightarrow p^* \cdot q \leq r$	KA-LFP-L
$1 + p^* \cdot p \equiv p^*$	KA-UNROLL-R
$p + q \cdot r \leq q \Rightarrow p \cdot r^* \leq q$	KA-LFP-R

Additional Boolean Algebra Axioms	
$a + (b \cdot c) \equiv (a + b) \cdot (a + c)$	BA-PLUS-DIST
$a + 1 \equiv 1$	BA-PLUS-ONE
$a + \neg a \equiv 1$	BA-EXCL-MID
$a \cdot b \equiv b \cdot a$	BA-SEQ-COMM
$a \cdot \neg a \equiv 0$	BA-CONTRA
$a \cdot a \equiv a$	BA-SEQ-IDEM

Equational Theory: Axioms

Kleene Algebra Axioms

$p + (q + r) \equiv (p + q) + r$	KA-PLUS-ASSOC
$p + q \equiv q + p$	KA-PLUS-COMM
$p + 0 \equiv p$	KA-PLUS-ZERO
$p + p \equiv p$	KA-PLUS-IDEM
$p \cdot (q \cdot r) \equiv (p \cdot q) \cdot r$	KA-SEQ-ASSOC
$1 \cdot p \equiv p$	KA-ONE-SEQ
$p \cdot 1 \equiv p$	KA-SEQ-ONE
$p \cdot (q + r) \equiv p \cdot q + p \cdot r$	KA-SEQ-DIST-L
$(p + q) \cdot r \equiv p \cdot r + q \cdot r$	KA-SEQ-DIST-R
$0 \cdot p \equiv 0$	KA-ZERO-SEQ
$p \cdot 0 \equiv 0$	KA-SEQ-ZERO
$1 + p \cdot p^* \equiv p^*$	KA-UNROLL-L
$q + p \cdot r \leq r \Rightarrow p^* \cdot q \leq r$	KA-LFP-L
$1 + p^* \cdot p \equiv p^*$	KA-UNROLL-R
$p + q \cdot r \leq q \Rightarrow p \cdot r^* \leq q$	KA-LFP-R

Additional Boolean Algebra Axioms

$a + (b \cdot c) \equiv (a + b) \cdot (a + c)$	BA-PLUS-DIST
$a + 1 \equiv 1$	BA-PLUS-ONE
$a + \neg a \equiv 1$	BA-EXCL-MID
$a \cdot b \equiv b \cdot a$	BA-SEQ-COMM
$a \cdot \neg a \equiv 0$	BA-CONTRA
$a \cdot a \equiv a$	BA-SEQ-IDEM

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

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

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

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

Equational Theory: Axioms

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER

Equational Theory: Axioms

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD

Equational Theory: Axioms

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD

Equational Theory: Axioms

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD
$f = n \cdot f = n' \equiv 0, \text{ if } n \neq n'$	PA-CONTRA

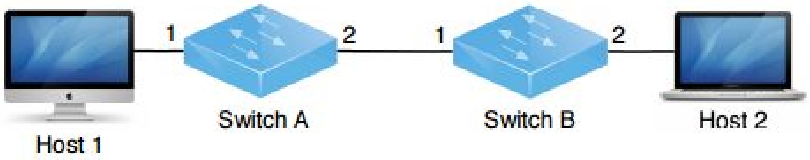
Equational Theory: Axioms

Packet Algebra Axioms

$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n, \text{ if } f \neq f'$	PA-MOD-MOD-COMM
$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n, \text{ if } f \neq f'$	PA-MOD-FILTER-COMM
$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD
$f = n \cdot f = n' \equiv 0, \text{ if } n \neq n'$	PA-CONTRA
$\sum_i f = i \equiv 1$	PA-MATCH-ALL

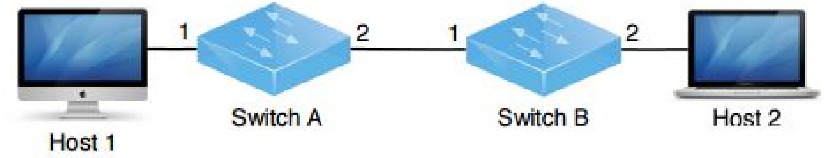
Equational Theory Example: Access Control

- 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

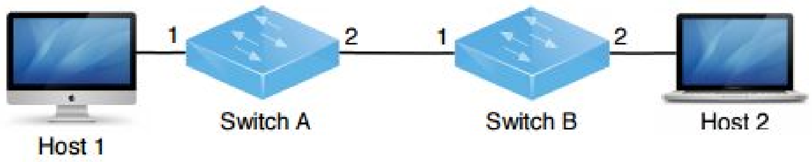


Equational Theory Example: Access Control

- We use the logical crossbar encoding with predicates

$$
\begin{aligned}
in &\triangleq (\mathsf{sw} = A \cdot \mathsf{pt} = 1) \\
out &\triangleq (\mathsf{sw} = B \cdot \mathsf{pt} = 2)
\end{aligned}
$$

$$
\begin{aligned}
a_A &\triangleq (\mathsf{sw} = A) \qquad & a_1 &\triangleq (\mathsf{pt} = 1) \\
a_B &\triangleq (\mathsf{sw} = B) \qquad & a_2 &\triangleq (\mathsf{pt} = 2)
\end{aligned}
$$



Equational Theory Example: Access Control

Lemma 1. $in \cdot a_B \cdot q \equiv 0$

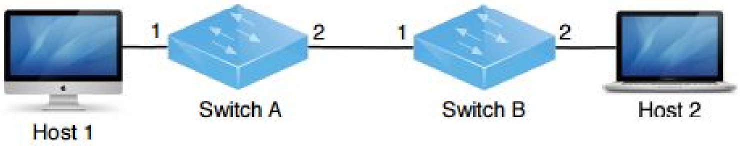
Proof.

$$
\begin{aligned}
& in \cdot a_B \cdot q \\
\equiv & \{ \text{ definition } in \text{ } \} \\
& a_A \cdot a_1 \cdot a_B \cdot q \\
\equiv & \{ \text{ KAT-COMMUTE } \} \\
& a_A \cdot a_B \cdot a_1 \cdot q \\
\equiv & \{ \text{ PA-CONTRA } \} \\
& 0 \cdot a_1 \cdot q \\
\equiv & \{ \text{ KA-ZERO-SEQ } \} \\
& 0 \qquad \square
\end{aligned}
$$

$in \triangleq (\mathsf{sw} = A \cdot \mathsf{pt} = 1)$
$out \triangleq (\mathsf{sw} = B \cdot \mathsf{pt} = 2)$

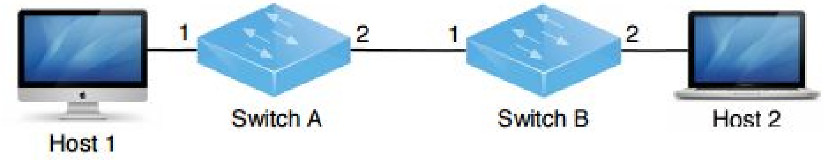
$a_A \triangleq (\mathsf{sw} = A)$
$a_B \triangleq (\mathsf{sw} = B)$

$a_1 \triangleq (\mathsf{pt} = 1)$
$a_2 \triangleq (\mathsf{pt} = 2)$



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.

$$
\begin{aligned}
& q \cdot a_A \cdot t \cdot out \\
\equiv\ & \{ \text{ definition } t \text{ } \} \\
& q \cdot a_A \cdot (a_A \cdot a_2 \cdot m_B \cdot m_1 + \\
& \qquad a_B \cdot a_1 \cdot m_A \cdot m_2 + \\
& \qquad a_A \cdot a_1 + \\
& \qquad a_B \cdot a_2) \cdot out \\
\equiv\ & \{ \text{ KA-SEQ-DIST-L, KA-SEQ-DIST-R } \} \\
& q \cdot a_A \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot out + \\
& q \cdot a_A \cdot a_B \cdot a_1 \cdot m_A \cdot m_2 \cdot out + \\
& q \cdot a_A \cdot a_A \cdot a_1 \cdot out + \\
& q \cdot a_A \cdot a_B \cdot a_2 \cdot out \\
\equiv\ & \{ \text{ definition } out \text{ } \} \\
& q \cdot a_A \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_B \cdot a_1 \cdot m_A \cdot m_2 \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_A \cdot a_1 \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_B \cdot a_2 \cdot a_B \cdot a_2 \\
\equiv\ & \{ \text{ PA-MOD-FILTER } \} \\
& q \cdot a_A \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot a_1 \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_B \cdot a_1 \cdot m_A \cdot a_A \cdot m_2 \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_A \cdot a_1 \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_B \cdot a_2 \cdot a_B \cdot a_2 \\
\equiv\ & \{ \text{ KAT-COMMUTE } \} \\
& q \cdot a_A \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot a_B \cdot a_1 \cdot a_2 + \\
& q \cdot a_A \cdot a_B \cdot a_1 \cdot m_A \cdot m_2 \cdot a_A \cdot a_B \cdot a_2 + \\
& q \cdot a_A \cdot a_A \cdot a_B \cdot a_1 \cdot a_2 + \\
& q \cdot a_A \cdot a_B \cdot a_2 \cdot a_B \cdot a_2 \\
\equiv\ & \{ \text{ PA-CONTRA } \} \\
& q \cdot a_A \cdot a_A \cdot a_2 \cdot m_B \cdot m_1 \cdot a_B \cdot 0 + \\
& q \cdot a_A \cdot a_B \cdot a_1 \cdot m_A \cdot m_2 \cdot 0 \cdot a_2 + \\
& q \cdot a_A \cdot a_A \cdot a_B \cdot 0 + \\
& q \cdot 0 \cdot a_2 \cdot a_B \cdot a_2 \\
\equiv\ & \{ \text{ KA-SEQ-ZERO, KA-ZERO-SEQ } \} \\
& 0 + 0 + 0 + 0 \\
\equiv\ & \{ \text{ KA-PLUS-IDEM } \} \\
& 0 \qquad \square
\end{aligned}
$$

$in \triangleq (\mathsf{sw} = A \cdot \mathsf{pt} = 1)$

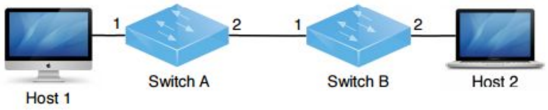
$out \triangleq (\mathsf{sw} = B \cdot \mathsf{pt} = 2)$

$a_A \triangleq (\mathsf{sw} = A)$

$a_B \triangleq (\mathsf{sw} = B)$

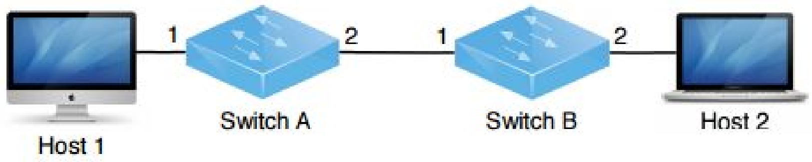
$a_1 \triangleq (\mathsf{pt} = 1)$

$a_2 \triangleq (\mathsf{pt} = 2)$



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. $in \cdot \text{SSH} \cdot (p_A \cdot t)^* \cdot out \equiv in \cdot \text{SSH} \cdot (p_B \cdot t)^* \cdot out$

Proof.

$$
\begin{aligned}
& in \cdot \text{SSH} \cdot (p_A \cdot t)^* \cdot out \\
\equiv\ & \{ \text{KAT-INVARIANT, definition } p_A \} \\
& in \cdot \text{SSH} \cdot ((a_A \cdot \neg\text{SSH} \cdot p + a_B \cdot p) \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv\ & \{ \text{KA-SEQ-DIST-R} \} \\
& 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 out \\
\equiv\ & \{ \text{KAT-COMMUTE} \} \\
& 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 out \\
\equiv\ & \{ \text{BA-CONTRA} \} \\
& in \cdot \text{SSH} \cdot (a_A \cdot 0 \cdot p \cdot t + a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv\ & \{ \text{KA-SEQ-ZERO/ZERO-SEQ, KA-PLUS-COMM, KA-PLUS-ZERO} \} \\
& in \cdot \text{SSH} \cdot (a_B \cdot p \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv\ & \{ \text{KA-UNROLL-L} \} \\
& 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 out \\
\equiv\ & \{ \text{KA-SEQ-DIST-L, KA-SEQ-DIST-R, definition } out \} \\
& in \cdot \text{SSH} \cdot a_B \cdot a_2 + \\
& 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} \} \\
& in \cdot a_B \cdot \text{SSH} \cdot a_2 + \\
& 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} \} \\
& in \cdot a_B \cdot \text{SSH} \cdot a_2 + \\
& 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 out \\
\equiv\ & \{ \text{KAT-COMMUTE, definition } out \} \\
& in \cdot \text{SSH} \cdot out + \\
& 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} \} \\
& 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} \} \\
& in \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv\ & \{ \text{KA-SEQ-ZERO/ZERO-SEQ, KA-PLUS-ZERO} \} \\
& 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} \} \\
& 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} \} \\
& in \cdot \text{SSH} \cdot (a_A \cdot p \cdot t \cdot \text{SSH} + a_B \cdot \neg\text{SSH} \cdot p \cdot t \cdot \text{SSH})^* \cdot out \\
\equiv\ & \{ \text{KA-SEQ-DIST-R} \} \\
& 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-INVARIANT, definition } p_B \} \\
& in \cdot \text{SSH} \cdot (p_B \cdot t)^* \cdot out
\end{aligned}
$$

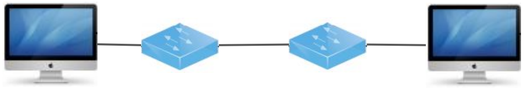
$\square$

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

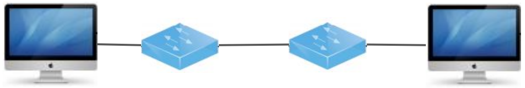
Reachability: some interesting questions

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

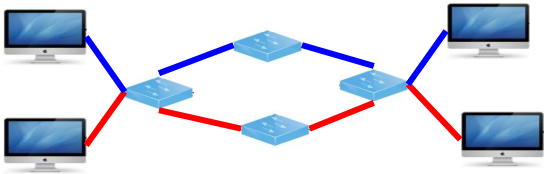


Reachability: some interesting questions

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

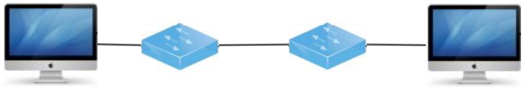


- Are managed hosts kept separate from unmanaged hosts?

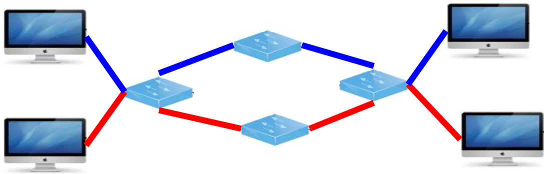


Reachability: some interesting questions

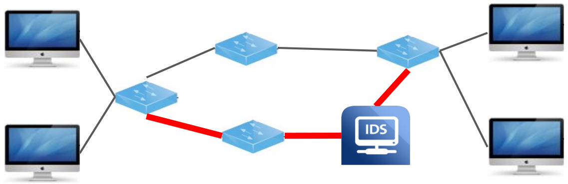
- Can host [a] send packets to host [b]?



- Are managed hosts kept separate from unmanaged hosts?

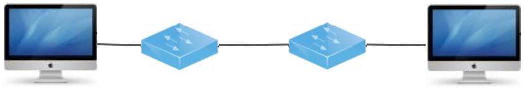


- Does all untrusted traffic traverse the intrusion detection system (IDS)?

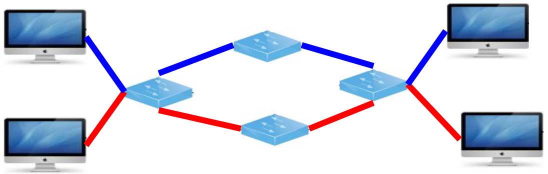


Reachability: some interesting questions

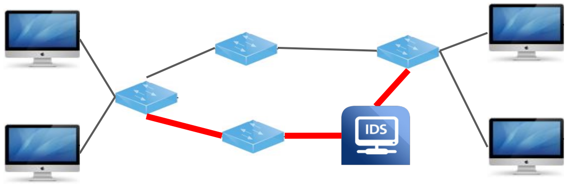
- Can host [a] send packets to host [b]?



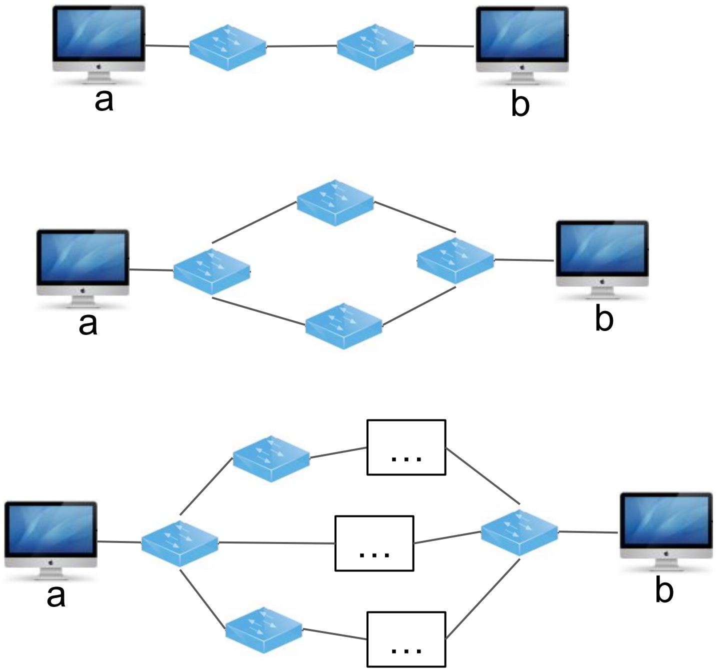
- Are managed hosts kept separate from unmanaged hosts?



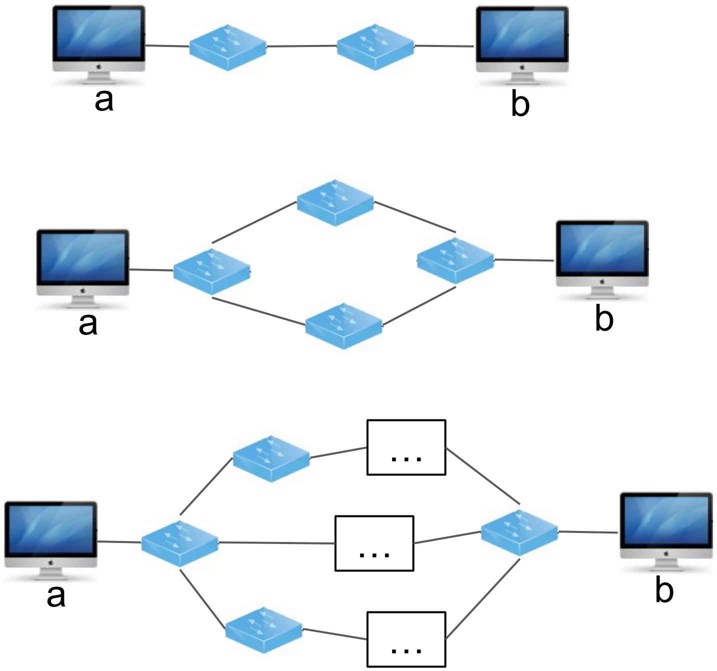
- Does all untrusted traffic traverse the intrusion detection system (IDS)?

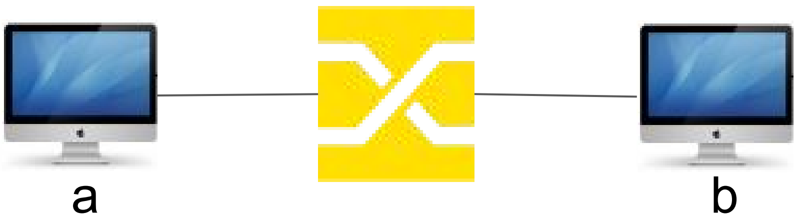


Reachability: can host [a] send packets to host [b]?

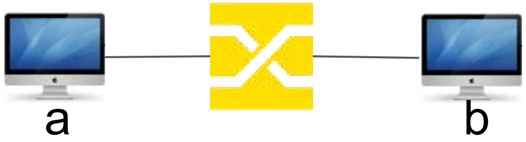


Reachability: can host [a] send packets to host [b]?

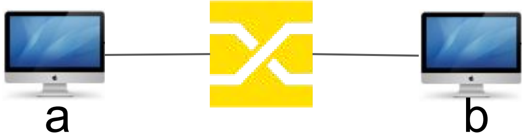




Reachability: can host [a] send packets to host [b]?



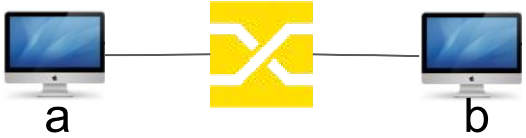
Reachability: can host [a] send packets to host [b]?



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

Behaviour of an entire network (crossbar model)

Reachability: can host [a] send packets to host [b]?



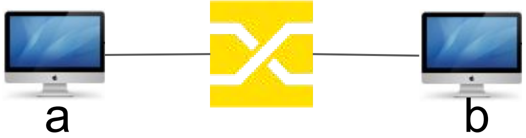
$$in \cdot (p \cdot t)^* \cdot out$$

Behaviour of an entire network (crossbar model)

$$in \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot out$$

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

Reachability: can host [a] send packets to host [b]?



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

Behaviour of an entire network (crossbar model)

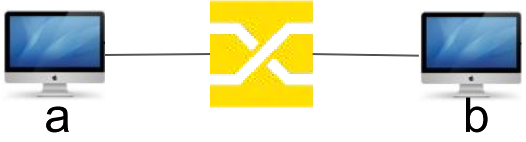
$$in \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot out$$

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

$$a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0$$

prepending *a* filters packets with source [a] and *b* filters packets with destination [b]

Reachability: can host [a] send packets to host [b]?

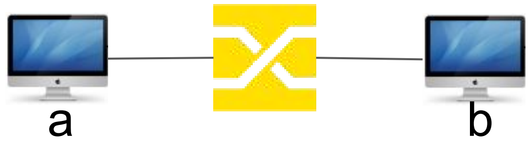


How do we know that this is correct?

$$a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0$$

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

$$a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0$$

prepending *a* filters packets with source [a] and *b* filters packets with destination [b]

NetKAT language model

Reduced NetKAT syntax

$$\boxed{\begin{array}{rrl} \text{Complete assignments} & \pi & \triangleq f_1 \leftarrow n_1 \cdots f_k \leftarrow n_k \\ \text{Complete tests} & \alpha, \beta & \triangleq f_1 = n_1 \cdots f_k = n_k \end{array}}$$

$$\begin{array}{rlll} \text{Reduced terms} & p, q ::= & \alpha & \textit{Complete test} \\ & \mid & \pi & \textit{Complete assignment} \\ & \mid & p + q & \textit{Union} \\ & \mid & p \cdot q & \textit{Sequence} \\ & \mid & p^* & \textit{Kleene star} \\ & \mid & \mathsf{dup} & \textit{Duplication} \end{array}$$

Simplified axioms for A and P

1 $\pi \equiv \pi \cdot \alpha_\pi$ 3 $\alpha \cdot \mathsf{dup} \equiv \mathsf{dup} \cdot \alpha$ 5 $\sum_\alpha \alpha \equiv 1,$

2 $\alpha \equiv \alpha \cdot \pi_\alpha$ 4 $\pi \cdot \pi' \equiv \pi'$ 6 $\alpha \cdot \beta \equiv 0,\ \alpha \neq \beta$

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

$$\begin{aligned} R(\pi) &= \{\pi\} \\ R(p+q) &= R(p) \cup R(q) \\ R(\alpha) &= \{\alpha\} \\ R(p \cdot q) &= \{xy \mid x \in R(p), y \in R(q)\} \\ R(\mathsf{dup}) &= \{\mathsf{dup}\} \\ R(p^*) &= \bigcup_{n \geq 0} R(p^n) \end{aligned}$$

Set of complete atoms (tests)

Set of complete assignments

Policies

$$\begin{array}{lll} p, q ::= & a & \textit{Filter} \\ \mid & f \leftarrow n & \textit{Modification} \\ \mid & p + q & \textit{Union} \\ \mid & p \cdot q & \textit{Sequential composition} \\ \mid & p^* & \textit{Kleene star} \\ \mid & \mathsf{dup} & \textit{Duplication} \end{array}$$

Packet Algebra Axioms

	$f \leftarrow n \cdot f' \leftarrow n' \equiv f' \leftarrow n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-MOD-COMM
	$f \leftarrow n \cdot f' = n' \equiv f' = n' \cdot f \leftarrow n$, if $f \neq f'$	PA-MOD-FILTER-COMM
3	$\mathsf{dup} \cdot f = n \equiv f = n \cdot \mathsf{dup}$	PA-DUP-FILTER-COMM
1	$f \leftarrow n \cdot f = n \equiv f \leftarrow n$	PA-MOD-FILTER
2	$f = n \cdot f \leftarrow n \equiv f = n$	PA-FILTER-MOD
4	$f \leftarrow n \cdot f \leftarrow n' \equiv f \leftarrow n'$	PA-MOD-MOD
6	$f = n \cdot f = n' \equiv 0$, if $n \neq n'$	PA-CONTRA
5	$\sum_i f = i \equiv 1$	PA-MATCH-ALL

NetKAT language model

Reduced NetKAT syntax

$$\boxed{\begin{array}{rrcl} \text{Complete assignments} & \pi & \triangleq & f_1 \leftarrow n_1 \cdots f_k \leftarrow n_k \\ \text{Complete tests} & \alpha, \beta & \triangleq & f_1 = n_1 \cdots f_k = n_k \end{array}}$$

$$\begin{array}{rrll} \text{Reduced terms} & p, q ::= & \alpha & \textit{Complete test} \\ & \mid & \pi & \textit{Complete assignment} \\ & \mid & p + q & \textit{Union} \\ & \mid & p \cdot q & \textit{Sequence} \\ & \mid & p^* & \textit{Kleene star} \\ & \mid & \mathsf{dup} & \textit{Duplication} \end{array}$$

Simplified axioms for A and P

$$\boxed{\begin{array}{lll} \pi \equiv \pi \cdot \alpha_\pi & \alpha \cdot \mathsf{dup} \equiv \mathsf{dup} \cdot \alpha & \sum_\alpha \alpha \equiv 1, \\ \alpha \equiv \alpha \cdot \pi_\alpha & \pi \cdot \pi' \equiv \pi' & \alpha \cdot \beta \equiv 0,\ \alpha \neq \beta \end{array}}$$

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

$$\begin{aligned} R(\pi) &= \{\pi\} \\ R(p+q) &= R(p) \cup R(q) \\ R(\alpha) &= \{\alpha\} \\ R(p \cdot q) &= \{xy \mid x \in R(p), y \in R(q)\} \\ R(\mathsf{dup}) &= \{\mathsf{dup}\} \\ R(p^*) &= \bigcup_{n \geq 0} R(p^n) \end{aligned}$$

I is a guarded string

NetKAT language model consists of regular subsets of a restricted class of guarded strings I.

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

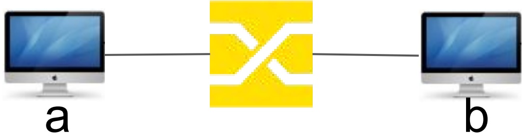
$$\begin{aligned} G(\pi) &= \{\alpha \cdot \pi \mid \alpha \in A\} \\ G(p+q) &= G(p) \cup G(q) \\ G(\alpha) &= \{\alpha \cdot \pi_\alpha\} \\ G(p \cdot q) &= G(p) \diamond G(q) \\ G(\mathsf{dup}) &= \{\alpha \cdot \pi_\alpha \cdot \mathsf{dup} \cdot \pi_\alpha \mid \alpha \in A\} \\ G(p^*) &= \bigcup_{n \geq 0} G(p^n) \end{aligned}$$

Guarded concatenation

$$\boxed{\alpha \cdot p \cdot \pi \diamond \beta \cdot q \cdot \pi' = \begin{cases} \alpha \cdot p \cdot q \cdot \pi' & \text{if } \beta = \alpha_\pi \\ \text{undefined} & \text{if } \beta \neq \alpha_\pi \end{cases}}$$

$$A \diamond B = \{p \diamond q \mid p \in A,\ q \in B\} \subseteq I$$

Reachability: can host [a] send packets to host [b]?



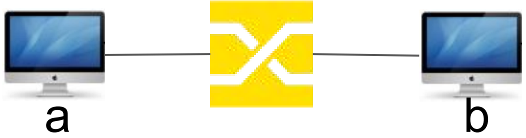
Definition 2 (Reachability). *We say b is reachable from a if and only if there exists a trace*

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

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

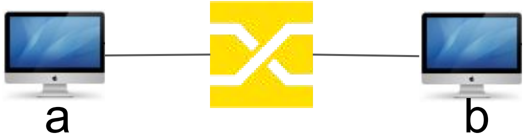
Intuition: [a] can talk to [a] if there is a trace where packet's first hop is [a] last hop is [b]

Reachability: can host [a] send packets to host [b]?



Theorem 4 (Reachability Correctness). *For predicates a and b, policy p, and topology t, $a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0$, 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, $a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0$, if and only if b is reachable from a.*

Reachability: can host [a] send packets to host [b]?

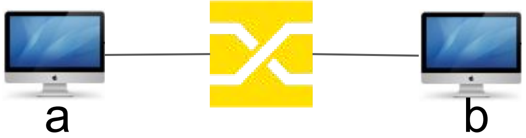
Proof. We translate the NetKAT equation into the language model:

$$\boxed{a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0}$$

$$\Rightarrow \quad \exists \alpha, \pi_n, \cdots, \pi_1 .$$
$$\alpha \cdot \pi_n \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi_1 \in G(a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b)$$

Theorem 4 (Reachability Correctness)**.** *For predicates a and b, policy p, and topology t,* $\boxed{a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0}$*, if and only if b is reachable from a.*

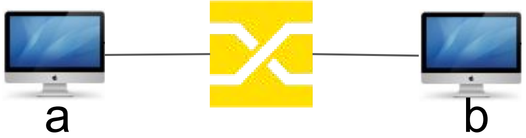
Reachability: can host [a] send packets to host [b]?



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

$$
\begin{aligned}
& a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0 \\
\Rightarrow \quad & \exists \alpha, \pi_n, \cdots, \pi_1 . \\
& \boxed{\alpha \cdot \pi_n \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi_1 \in G(a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b)}
\end{aligned}
$$

Reachability: can host [a] send packets to host [b]?



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

$$
\begin{aligned}
&\exists pk_1, \cdots, pk_n . \\
&\quad \langle pk_1, \cdots, pk_n \rangle \in \mathsf{rng}(\, [\![\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*]\!]), \\
&\quad [\![a]\!]\, \langle pk_n \rangle = \{\langle pk_n \rangle\} \text{ and} \\
&\quad [\![b]\!]\, \langle pk_1 \rangle = \{\langle pk_1 \rangle\}
\end{aligned}
$$

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

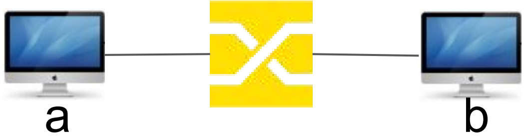
$$
\begin{aligned}
& a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0 \\
\Rightarrow\ & \exists \alpha, \pi_n, \cdots, \pi_1 . \\
& \boxed{\alpha \cdot \pi_n \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi_1 \in G(a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b)}
\end{aligned}
$$

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

$$\langle pk_1, \cdots, pk_n \rangle \in \mathit{rng}([\![\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*]\!])$$

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

Reachability: can host [a] send packets to host [b]?



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

$$
\begin{aligned}
& \exists pk_1, \cdots, pk_n. \\
& \quad \langle pk_1, \cdots, pk_n \rangle \in \mathsf{rng}(\, [\![\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*]\!]), \\
& \quad [\![a]\!] \, \langle pk_n \rangle = \{\langle pk_n \rangle\} \text{ and} \\
& \quad [\![b]\!] \, \langle pk_1 \rangle = \{\langle pk_1 \rangle\} \\
\Rightarrow \quad & \exists \pi'_1, \cdots, \pi'_m. \\
& \quad \alpha_{\pi'_m} \cdot \pi'_m \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi'_1 \in G(\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*), \\
& \quad \alpha_{\pi'_m} \cdot \pi'_m \in G(a) \text{ and} \\
& \quad \alpha_{\pi'_1} \cdot \pi'_1 \in G(b)
\end{aligned}
$$

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

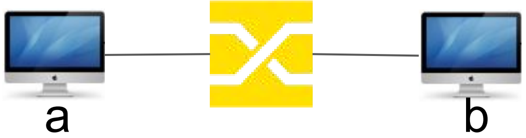
$$
\begin{aligned}
& a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0 \\
\Rightarrow \quad & \exists \alpha, \pi_n, \cdots, \pi_1. \\
& \boxed{\alpha \cdot \pi_n \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi_1 \in G(a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b)}
\end{aligned}
$$

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

$$\langle pk_1, \cdots, pk_n \rangle \in \mathit{rng}([\![\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*]\!])$$

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

Reachability: can host [a] send packets to host [b]?



$$
\begin{aligned}
& \exists pk_1, \cdots, pk_n. \\
& \quad \langle pk_1, \cdots, pk_n \rangle \in \mathsf{rng}(\llbracket \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \rrbracket), \\
& \quad \llbracket a \rrbracket \langle pk_n \rangle = \{\langle pk_n \rangle\} \text{ and} \\
& \quad \llbracket b \rrbracket \langle pk_1 \rangle = \{\langle pk_1 \rangle\} \\
\Rightarrow \; & \exists \pi'_1, \cdots, \pi'_m. \\
& \quad \alpha_{\pi'_m} \cdot \pi'_m \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi'_1 \in G(\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*), \\
& \quad \alpha_{\pi'_m} \cdot \pi'_m \in G(a) \text{ and} \\
& \quad \alpha_{\pi'_1} \cdot \pi'_1 \in G(b)
\end{aligned}
$$

To prove soundness we let $\alpha = \alpha_{\pi_n}$ and $m = n$ to show that if

$$\alpha \cdot \pi_n \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi_1 \in G(a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b)$$

then,

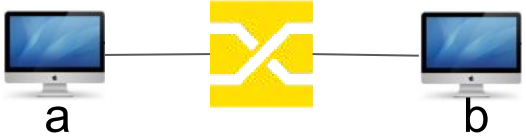
$$\alpha_{\pi'_m} \cdot \pi'_m \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi'_1 \in G(\mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^*)$$

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:

$$
\begin{aligned}
& a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0 \\
\Rightarrow \; & \exists \alpha, \pi_n, \cdots, \pi_1. \\
& \quad \boxed{\alpha \cdot \pi_n \cdot \mathsf{dup} \cdots \mathsf{dup} \cdot \pi_1 \in G(a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b)}
\end{aligned}
$$

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

$$a \cdot \mathsf{dup} \cdot (p \cdot t \cdot \mathsf{dup})^* \cdot b \not\equiv 0$$

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

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

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

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
2. NetKAT: Semantic Foundations for Networks, Technical Report, 2013, https://ecommons.cornell.edu/handle/1813/34445
3. Dexter Kozen. Kleene algebra with tests. Transactions on Programming Languages and Systems, 19(3):427–443, May 1997.