NetKAT: Semantic Foundations for Networks

Carolyn Jane Anderson[*]

[*]Swarthmore College

Symposium on Principles of Programming Languages,
January 2014

Table Of Contents

Table of Contents

Network Devices

Traditional network devices have been called "the last bastion of mainframe computing" [2].

Unlike modern computers, networks have been built the same way since the 1970s.

This design makes it difficult to extend networks with new functionality reason about their behavior.

Software-Defined Networks

A revolution has taken place with the recent rise of software-defined networking (SDN).

Two main components in a SDN:

- Controller
- Programmable switches

Results:

- Sophisticated applications (load balancing, etc.)
- OpenFlow API: low-level language for switches [1]

Table of Contents

The NetKAT Language

NetKAT is a new framework for specifying, programming, and reasoning about networks based on *Kleene algebra with tests* (KAT) [3].

For specification and reasoning, NetKAT also provides a sound and complete equation system that captures equivalences between NetKAT programs.

Kleene Algebra with Tests (KAT)

KAT has been applied successfully in a number of application areas, including compiler, device driver, and communication protocol verification [4].

equivalence checking in KAT has a PSPACE decision procedure.

NetKAT applies this theory to a new domain: networks.

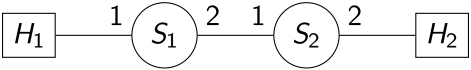

Figure 1: Sample network

We want to configure the network to implement the following policies:

- *Forwarding:* transfer packets between hosts
- *Access control:* block SSH packets

Forwarding (1/2)

We represent a packet as a record with fields for standard headers such as: source address (src), destination address (dst), and protocol type (typ).

We also use two fields, switch (sw) and port (pt), that identify the current location of the packet in the network.

Two atomic NetKAT policies:

- Filter: $f = n$
- Modify: $f \leftarrow n$

Forwarding (2/2)

We can implement the forwarding policy as follows:

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

Union of two behaviors:

- Update the pt field of all packets destined for H_1 to 1
- Update the pt field of all packets destined for H_2 to 2

Access Control

One way to do this is to compose a filter that blocks SSH traffic with the forwarding policy in sequence:

$$p_{AC} \triangleq (\mathsf{typ} \neq \mathrm{SSH}) \cdot p$$

This policy drops the input packet if its typ field is SSH and otherwise forwards it using p.

Topology

We can model the topology as the union of smaller policies that encode the behavior of each link.

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

Switches Meet Topology

A packet traverses the network in interleaved steps of processing by the switches and topology.

Using the Kleene star operator, which iterates a policy zero or more times, we can encode the overall behavior of the network.

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

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

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

Table of Contents

NetKAT Equational Axioms (1/2)

$p + (q + r) \equiv (p + q) + r$	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

Figure 2: Kleene algebra axioms

NetKAT Equational Axioms (2/2)

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

Figure 3: Packet algebra axioms

Soundness and Completeness

- **Soundness:** every equivalence provable using the NetKAT axioms also holds in the denotational model
- **Completeness:** every equivalence which holds in the denotational model is provable using the axioms

Theorem

proof based on NetKAT axioms $\Leftrightarrow$ *denotational model*

Proof: We use properties from KA and KAT to prove soundness and completeness of NetKAT.

Decidability

Theorem

The equational theory of NetKAT is PSPACE-complete.

Proof:

- **PSPACE-hardness:** KAT $\leq_P$ NetKAT
- **PSPACE-containment:** Given two NetKAT expressions e_1 and e_2, we want to show there exists a packet pk and a history h such that h can be achieved from e_1 but not e_2.
  - We will non-deterministically choose pk.
  - We follow a nondeterministically-guessed trajectory through e_1; at the same time, we trace all possible trajectories through e_2.

$$\text{NPSPACE} \xrightarrow{\text{Savitch's theorem}} \text{PSPACE}$$

Table of Contents

Practical Applications

1. **Reachability properties:** Syntactic techniques
2. **Traffic isolation:** Use of dedicated header fields
3. **Compilation to flow tables**: Prioritization with if-else equivalents

References

[1] McKeown et Al. *OpenFlow: Enabling innovation in campus*. URL: https://doi.org/10.1145/1355734.1355746.

[2] James Hamilton. *Networking: The last bastion of mainframe computing*. URL: http://tinyurl.com/y9uz64e.

[3] Dexter Kozen. *Kleene algebra with tests*. URL: https://doi.org/10.1145/256167.256195.

[4] Dexter Kozen. *Kleene algebras with tests and the static analysis of programs*. URL: https://www.cs.cornell.edu/~kozen/Papers/static.pdf.