A Coalgebraic Decision Procedure for NetKAT

Dexter Kozen

Cornell University

MFPS XXX
June 12, 2014

NetKAT Collaborators

Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker, NetKAT: Semantic foundations for networks, POPL'14.

Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson, A coalgebraic decision procedure for NetKAT, Tech Report http://hdl.handle.net/1813/36255, Cornell University, March 2014.

Networking

“The last bastion of mainframe computing” [Hamilton 2009]

- Modern computers
  - implemented with commodity hardware
  - programmed using general-purpose languages, standard interfaces
- Networks
  - built the same way since the 1970s
  - special-purpose devices implemented on custom hardware: routers, switches, firewalls, load balancers, middle-boxes
  - programmed individually using proprietary interfaces
  - network configuration (“tuning”) largely a black art

Difficult to extend with new functionality

Effectively impossible to reason precisely about behavior

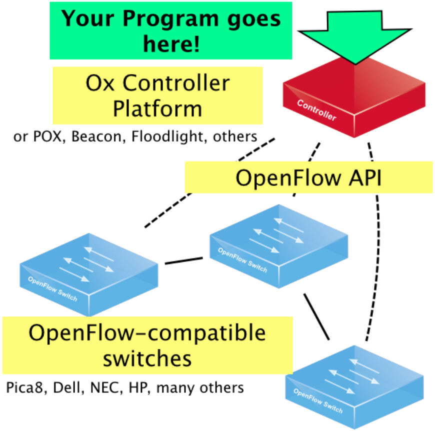
Software Defined Networks (SDN)

Main idea behind SDN

A general-purpose controller manages a collection of programmable switches

- controller can monitor and respond to network events
  - new connections from hosts
  - topology changes
  - shifts in traffic load
- controller can reprogram the switches on the fly
  - adjust routing tables
  - change packet filtering policies

SDN Network Architecture



Software Defined Networks (SDN)

Controller has a global view of the network

Enables a wide variety of applications:

- standard applications
  - shortest-path routing
  - traffic monitoring
  - access control
- more sophisticated applications
  - load balancing
  - intrusion detection
  - fault tolerance

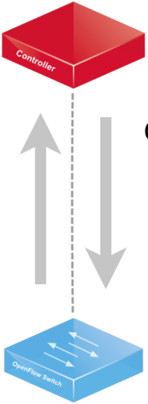
OpenFlow

A first step: the OpenFlow API [McKeown & al., SIGCOMM 08]

- specifies capabilities and behavior of switch hardware
- a language for manipulating network configurations
- very low-level: easy for hardware to implement, difficult for humans to write and reason about

Provided an open standard that any vendor could implement

OpenFlow API



Switch to controller:
- switch_connected
- switch_disconnected
- packet_in
- stats_reply

Controller to switch:
- packet_out
- flow_mod
- stats_request

A Major Trend in Industry

Backbone network

runs OpenFlow

Bought by VMware for $1.2B

Network Programming Languages & Analysis Tools

- Formally Verifiable Networking [Wang & al., HotNets 09]
- FlowChecker [Al-Shaer & Saeed Al-Haj, SafeConfig 10]
- Anteater [Mai & al., SIGCOMM 11]
- Nettle [Voellmy & Hudak, PADL 11]
- Header Space Analysis [Kazemian & al., NSDI 12]
- Frenetic [Foster & al., ICFP 11] [Reitblatt & al., SIGCOMM 12]
- NetCore [Guha & al., PLDI 13] [Monsanto & al., POPL 12]
- Pyretic [Monsanto & al., NSDI 13]
- VeriFlow [Khurshid & al., NSDI 13]
- Participatory networking [Ferguson & al., SIGCOMM 13]
- Maple [Voellmy & al., SIGCOMM 13]

Goals:

- raise the level of abstraction above hardware-based APIs (OpenFlow)
- make it easier to build sophisticated and reliable SDN applications and reason about them

Network Programming Languages & Analysis Tools

- Formally Verifiable Networking [Wang & al., HotNets 09]
- FlowChecker [Al-Shaer & Saeed Al-Haj, SafeConfig 10]
- Anteater [Mai & al., SIGCOMM 11]
- Nettle [Voellmy & Hudak, PADL 11]
- Header Space Analysis [Kazemian & al., NSDI 12]
- Frenetic [Foster & al., ICFP 11] [Reitblatt & al., SIGCOMM 12]
- NetCore [Guha & al., PLDI 13] [Monsanto & al., POPL 12]
- Pyretic [Monsanto & al., NSDI 13]
- VeriFlow [Khurshid & al., NSDI 13]
- Participatory networking [Ferguson & al., SIGCOMM 13]
- Maple [Voellmy & al., SIGCOMM 13]

Goals:

- raise the level of abstraction above hardware-based APIs (OpenFlow)
- make it easier to build sophisticated and reliable SDN applications and reason about them

NetKAT [Anderson & al. 14]

NetKAT

=

Kleene algebra with tests (KAT)

+

additional specialized constructs particular to network topology and packet switching

NetKAT [Anderson & al. 14]

NetKAT

$=$

Kleene algebra with tests (KAT)

$+$

additional specialized constructs particular to network topology and packet switching

- primitives for filtering, forwarding, duplicating, modifying packets
- parallal composition ($+$), sequential composition ($\cdot$), iteration (*)
- can specify network topology and routing, end-to-end behavior, access control
- integrated as part of the Frenetic suite of network management tools [Foster & al. 10]

NetKAT Results

[Anderson & al., POPL 14]

- NetKAT syntax and standard packet-switching semantics
- equivalent language model
- sound and complete deduction system
- (very inefficient) PSPACE algorithm & hardness proof
- practical applications: reachability analysis, non-interference, compiler correctness

[Foster & al. 14]

- coalgebraic semantics
- an efficient decision procedure based on bisimulation
- implementation and benchmarks

Axioms of Kleene Algebra (KA)

Idempotent Semiring Axioms

$$p + (q + r) = (p + q) + r \qquad p(qr) = (pq)r$$

$$p + q = q + p \qquad 1p = p1 = p$$

$$p + 0 = p \qquad p0 = 0p = 0$$

$$p + p = p$$

$$p(q + r) = pq + pr \qquad a \leq b \stackrel{\triangle}{\iff} a + b = b$$

$$(p + q)r = pr + qr$$

Axioms for *

$$1 + pp^* \leq p^* \qquad q + px \leq x \;\Rightarrow\; p^*q \leq x$$

$$1 + p^*p \leq p^* \qquad q + xp \leq x \;\Rightarrow\; qp^* \leq x$$

Standard Model

Regular sets of strings over Σ

For $A, B \subseteq \Sigma^*$,

$$A + B = A \cup B \qquad AB = \{xy \mid x \in A,\ y \in B\}$$

$$A^* = \bigcup_{n \geq 0} A^n, \text{ where } A^0 = \{\varepsilon\},\ A^{n+1} = AA^n$$

$$1 = \{\varepsilon\} \qquad 0 = \emptyset$$

This is the free KA on generators Σ

Relational Models

Binary relations on a set X

For $R, S \subseteq X \times X$,

$$R + S = R \cup S \qquad RS = \{(x,z) \mid \exists y\ (x,y) \in R \wedge (y,z) \in S\}$$

$$R^* = \bigcup_{n \geq 0} R^n \ \ \text{(reflexive transitive closure of } R\text{)}$$

$$1 = \{(x,x) \mid x \in X\} \qquad 0 = \emptyset$$

Axioms of KA are complete for the equational theory of relational models

Deciding KA

- PSPACE-complete [(1 + Stock)Meyer 74]
  - automata-based decision procedure
  - nondeterministically guess a string in $L(M_1) \oplus L(M_2)$, simulate the two automata
  - convert to deterministic using Savitch's theorem
  - inefficient—$\Omega(n^2)$ space, exponential time best-case
- coalgebraic decision procedures [Bonchi & Pous 12]
  - bisimulation-based
  - uses Brzozowski/Antimirov derivatives
  - Hopcroft–Karp union-find data structure, up-to techniques
  - implementation in OCaml
  - linear space, practical

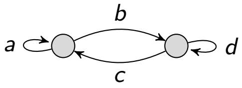
Matrices over a KA form a KA

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

$$0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad 1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a+bd^*c)^* & (a+bd^*c)^*bd^* \\ (d+ca^*b)^*ca^* & (d+ca^*b)^* \end{bmatrix}$$



Systems of Linear Inequalities

Theorem

Any system of n linear inequalities in n unknowns has a unique least solution

$$
\begin{aligned}
q_1 + p_{11}x_1 + p_{12}x_2 + \cdots p_{1n}x_n &\leq x_1 \\
&\vdots \\
q_n + p_{n1}x_1 + p_{n2}x_2 + \cdots p_{nn}x_n &\leq x_n
\end{aligned}
$$

$$
\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} + \begin{bmatrix} P = p_{ij} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
$$

Least solution is P^*q

$(K, B, +, \cdot, ^*, \bar{}, 0, 1)$, $B \subseteq K$

- $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra
- $(B, +, \cdot, \bar{}, 0, 1)$ is a Boolean algebra
- $(B, +, \cdot, 0, 1)$ is a subalgebra of $(K, +, \cdot, 0, 1)$

Encodes imperative programming constructs, subsumes Hoare logic

$p; q$	pq
if b **then** p **else** q	$bp + \bar{b}q$
while b **do** p	$(bp)^*\bar{b}$
$\{b\}\ p\ \{c\}$	$bp \leq pc,\ \ bp = bpc,\ \ bp\bar{c} = 0$
$\dfrac{\{bc\}\ p\ \{c\}}{\{c\}\ \textbf{while}\ b\ \textbf{do}\ p\ \{\bar{b}c\}}$	$bcp\bar{c} = 0 \ \Rightarrow\ (c(bp)^*\bar{b})\,\overline{\bar{b}} = 0$

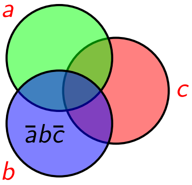
Guarded Strings [Kaplan 69]

Σ action symbols $\qquad$ T test symbols

B = free Boolean algebra generated by T
At = atoms of $B = \{\alpha, \beta, \ldots\}$

Guarded strings $GS = At \cdot (\Sigma \cdot At)^*$

$$\alpha_0 p_1 \alpha_1 p_2 \alpha_2 \cdots \alpha_{n-1} p_n \alpha_n$$



A Language Model for KAT

Regular sets of guarded strings over Σ, T

For $A, B \subseteq GS$,

$$A + B = A \cup B \qquad AB = \{x\alpha y \mid x\alpha \in A,\ \alpha y \in B\}$$

$$A^* = \bigcup_{n \geq 0} A^n \qquad 1 = At \qquad 0 = \emptyset$$

- $p \in \Sigma$ interpreted as $\{\alpha p \beta \mid \alpha, \beta \in At\}$
- $b \in T$ interpreted as $\{\alpha \mid \alpha \leq b\}$

This is the free KAT on generators Σ, T

NetKAT Primitives

- a packet π is an assignment of constant values n to fields x
- a packet history is a nonempty sequence of packets
  $\pi_1 :: \pi_2 :: \cdots :: \pi_k$
- the head packet is π_1

NetKAT Primitives

- assignments $x \leftarrow n$
  assign constant value n to field x in the head packet
- tests $x = n$
  if value of field x in the head packet is n, then pass, else drop
- dup
  duplicate the head packet

NetKAT Primitives

Example

$$sw = 6\,;\, pt = 88\,;\, sec \leftarrow true\,;\, sw \leftarrow 7\,;\, pt \leftarrow 50$$

"If this packet is at port 88 of switch 6, set the security bit to true and forward the packet to port 50 of switch 7."

Standard Model

Standard model of NetKAT is a packet-forwarding model

$$[\![e]\!] : H \to 2^H$$

where $H = \{\text{packet histories}\}$

$$
\begin{aligned}
[\![x \leftarrow n]\!](\pi_1 :: \sigma) &\triangleq \{\pi_1[n/x] :: \sigma\} \\
[\![x = n]\!](\pi_1 :: \sigma) &\triangleq \begin{cases} \{\pi_1 :: \sigma\} & \text{if } \pi_1(x) = n \\ \emptyset & \text{if } \pi_1(x) \neq n \end{cases} \\
[\![\mathrm{dup}]\!](\pi_1 :: \sigma) &\triangleq \{\pi_1 :: \pi_1 :: \sigma\}
\end{aligned}
$$

Standard Model

$$[\![p+q]\!](\sigma) \triangleq [\![p]\!](\sigma) \cup [\![q]\!](\sigma)$$

$$[\![p\,;q]\!](\sigma) \triangleq \bigcup_{\tau \in [\![p]\!](\sigma)} [\![q]\!](\tau) \quad \text{(Kleisli composition)}$$

$$[\![p^*]\!](\sigma) \triangleq \bigcup_n [\![p^n]\!](\sigma)$$

$$[\![1]\!](\sigma) \triangleq [\![\mathsf{pass}]\!](\sigma) = \{\sigma\}$$

$$[\![0]\!](\sigma) \triangleq [\![\mathsf{drop}]\!](\sigma) = \emptyset$$

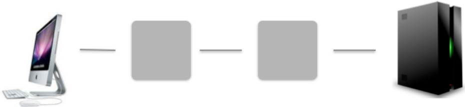
Note that $+$ is conjunctive instead of disjunctive!

NetKAT Axioms

$$
\begin{aligned}
x \leftarrow n \,;\, y \leftarrow m &\equiv y \leftarrow m \,;\, x \leftarrow n \quad \text{if } x \neq y \\
x \leftarrow n \,;\, y = m &\equiv y = m \,;\, x \leftarrow n \quad \text{if } x \neq y \\
x = n \,;\, \mathrm{dup} &\equiv \mathrm{dup} \,;\, x = n \\
x \leftarrow n \,;\, x = n &\equiv x \leftarrow n \\
x = n \,;\, x \leftarrow n &\equiv x = n \\
x \leftarrow n \,;\, x \leftarrow m &\equiv x \leftarrow m \\
x = n \,;\, x = m &\equiv 0 \quad \text{if } n \neq m \\
\left(\sum_n x = n\right) &\equiv 1
\end{aligned}
$$

Application: Rule Optimization

Given a program and a topology:



“Will my network behave the same if I put the firewall rules on A, or on switch B (or both)?”

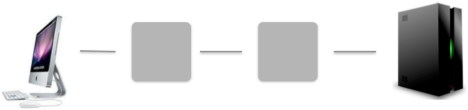
Formally, does the following equivalence hold?

(**filter** switch = A • firewall · route) + (**filter** switch = B • route)

=

(**filter** switch = A • route) + (**filter** switch = B • firewall • route)

Code Motion Proof



$in \cdot \mathrm{SSH} \cdot (p_A \cdot t)^* \cdot out$
$\equiv$ { KAT-Invariant, definition p_A }
$in \cdot \mathrm{SSH} \cdot ((a_A \cdot \neg\mathrm{SSH} \cdot p + a_B \cdot p) \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KA-Seq-Dist-R }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot \neg\mathrm{SSH} \cdot p \cdot t \cdot \mathrm{SSH} + a_B \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KAT-Commute }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot \neg\mathrm{SSH} \cdot \mathrm{SSH} \cdot p \cdot t + a_B \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { BA-Contra }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot 0 \cdot p \cdot t + a_B \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KA-Seq-Zero/Zero-Seq, KA-Plus-Comm, KA-Plus-Zero }
$in \cdot \mathrm{SSH} \cdot (a_B \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KA-Unroll-L }
$in \cdot \mathrm{SSH} \cdot (1 + (a_B \cdot p \cdot t \cdot \mathrm{SSH}) \cdot (a_B \cdot p \cdot t \cdot \mathrm{SSH})^*) \cdot out$
$\equiv$ { KA-Seq-Dist-L, KA-Seq-Dist-R, definition out }
$in \cdot \mathrm{SSH} \cdot a_B \cdot a_2 +$
$in \cdot \mathrm{SSH} \cdot a_B \cdot p \cdot t \cdot \mathrm{SSH} \cdot (a_B \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot a_B \cdot a_2$
$\equiv$ { KAT-Commute }
$in \cdot a_B \cdot \mathrm{SSH} \cdot a_2 +$
$in \cdot a_B \cdot \mathrm{SSH} \cdot p \cdot t \cdot \mathrm{SSH} \cdot (a_B \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot a_B \cdot a_2$
$\equiv$ { Lemma 1 }
$0 + 0$
$\equiv$ { KA-Plus-Idem }
0

$\equiv$ { KA-Plus-Idem }
$0 + 0$
$\equiv$ { Lemma 1, Lemma 2 }
$in \cdot a_B \cdot \mathrm{SSH} \cdot a_2 +$
$in \cdot \mathrm{SSH} \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot p \cdot \mathrm{SSH} \cdot a_A \cdot t \cdot out$
$\equiv$ { KAT-Commute, definition out }
$in \cdot \mathrm{SSH} \cdot out +$
$in \cdot \mathrm{SSH} \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot a_A \cdot p \cdot t \cdot \mathrm{SSH} \cdot out$
$\equiv$ { KA-Seq-Dist-L, KA-Seq-Dist-R }
$in \cdot \mathrm{SSH} \cdot (1 + (a_A \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH})) \cdot out$
$\equiv$ { KA-Unroll-R }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KA-Seq-Zero/Zero-Seq, KA-Plus-Zero }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH} + a_B \cdot 0 \cdot p \cdot t)^* \cdot out$
$\equiv$ { BA-Contra }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH} + a_B \cdot \neg\mathrm{SSH} \cdot \mathrm{SSH} \cdot p \cdot t)^* \cdot out$
$\equiv$ { KAT-Commute }
$in \cdot \mathrm{SSH} \cdot (a_A \cdot p \cdot t \cdot \mathrm{SSH} + a_B \cdot \neg\mathrm{SSH} \cdot p \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KA-Seq-Dist-R }
$in \cdot \mathrm{SSH} \cdot ((a_A \cdot p + a_B \cdot \neg\mathrm{SSH} \cdot p) \cdot t \cdot \mathrm{SSH})^* \cdot out$
$\equiv$ { KAT-Invariant, definition p_B }
$in \cdot \mathrm{SSH} \cdot (p_B \cdot t)^* \cdot out$

Reduced NetKAT

Let e be an expression to be analyzed and let $x_1, \ldots, x_k$ be all fields appearing in e.

- A complete assignment is a sequence $x_1 \leftarrow n_1; \cdots; x_k \leftarrow n_k$
- A complete test is a sequence $x_1 = n_1; \cdots; x_k = n_k$

Facts:

- Every test is equivalent to a sum of complete tests.
- Every assignment is equivalent to sum of complete tests and complete assignments.
- The complete tests and complete assignments are in one-to-one correspondence (one of each for each tuple $(n_1, \ldots, n_k)$)

Reduced NetKAT Axioms

Let $P = \{\text{complete assignments}\} = \{p, q, \ldots\}$ and
$At = \{\text{complete tests}\} = \{\alpha, \beta, \ldots\}$

Let α_p be the complete test corresponding to the complete assignment p

Reduced NetKAT axioms:

$$\alpha\,\mathsf{dup} = \mathsf{dup}\,\alpha \qquad \alpha\alpha = \alpha$$

$$p\alpha_p = p \qquad \alpha\beta = 0,\ \alpha \neq \beta$$

$$\alpha_p p = \alpha_p \qquad \textstyle\sum_{\alpha \in At} \alpha = 1$$

$$qp = p$$

A Language Model

Regular sets of NetKAT reduced strings

$$NS = At \cdot P \cdot (\text{dup} \cdot P)^*$$

For $A, B \subseteq NS$,

$$A + B = A \cup B \qquad AB = \{\alpha xyq \mid \alpha xp \in A,\ \alpha_p yq \in B\}$$

$$A^* = \bigcup_{n \geq 0} A^n \qquad 1 = \{\alpha_p p \mid p \in P\} \qquad 0 = \emptyset$$

- $p \in P$ interpreted as $\{\alpha p \mid \alpha \in At\}$
- $\alpha \in At$ interpreted as $\{\alpha p_\alpha\}$
- dup interpreted as $\{\alpha_p p \, \text{dup} \, \alpha_p \mid p \in P\}$

A Language Model

Lemma

Every string over P and At is equivalent to a string in $NS = At \cdot P \cdot (\mathrm{dup} \cdot P)^*$

Theorem ([Anderson & al. 14])

1. *The family of regular subsets of NS forms a NetKAT and is isomorphic to the standard packet-switching model.*
2. *This is the free NetKAT on generators P and At.*

Brzozowski Derivatives and the Coalgebraic View
[Brzozowski 64, Rutten 99, Silva 10]

A DFA over Σ is a coalgebra (S, ε, δ) for the functor $FX = 2 \times X^\Sigma$ consisting of

$$\varepsilon : S \to 2 \qquad\qquad \delta : S \to S^\Sigma$$

the observations and actions (or continuations), respectively

The final coalgebra is the semantic Brzozowski derivative

$$\varepsilon : 2^{\Sigma^*} \to 2 \qquad\qquad \delta_a : 2^{\Sigma^*} \to 2^{\Sigma^*}$$

$$\varepsilon(A) = \begin{cases} 1 & \varepsilon \in A \\ 0 & \varepsilon \notin A \end{cases} \qquad\qquad \delta_a(A) = \{x \mid ax \in A\}$$

The map

$$L : S \to 2^{\Sigma^*} \qquad\qquad L(s) = \{x \mid x \text{ accepted starting from } s\}$$

is the unique homomorphism to the final coalgebra

The Syntactic Brzozowski Derivative

Let Exp = {regular expressions over Σ}

$$E : \text{Exp} \to 2 \qquad\qquad D_a : \text{Exp} \to \text{Exp},\ a \in \Sigma$$

$$\begin{aligned} E(e_1 + e_2) &= E(e_1) + E(e_2) & D_a(e_1 + e_2) &= D_a(e_1) + D_a(e_2) \\ E(e_1 e_2) &= E(e_1) \cdot E(e_2) & D_a(e_1 e_2) &= D_a(e_1)e_2 + E(e_1)D_a(e_2) \\ E(e^*) &= 1 & D_a(e^*) &= D_a(e)e^* \\ E(1) &= 1 & D_a(1) &= D_a(0) = 0 \\ E(0) &= E(a) = 0,\ a \in \Sigma & D_a(b) &= \begin{cases} 1 & b = a \\ 0 & b \neq a \end{cases} \end{aligned}$$

The map

$$L : \text{Exp} \to 2^{\Sigma^*} \qquad\qquad L(e) = \{\text{language represented by } e\}$$

is the unique homomorphism to the final coalgebra

KAT Coalgebras

A KAT coalgebra is a coalgebra (S, ε, δ) for the functor $FX = 2^{At} \times X^{At \times \Sigma}$ consisting of

$$\varepsilon : S \to 2^{At} \qquad \delta : S \to S^{At \times \Sigma}$$

$$\varepsilon_\alpha : S \to 2 \qquad \delta_{\alpha p} : S \to S$$

for $\alpha \in At$ and $p \in \Sigma$

Viewed as a deterministic automaton, acceptance defined coinductively:

$$\text{Accept}(s, \alpha p x) \triangleq \text{Accept}(\delta_{\alpha p}(s), x) \qquad \text{Accept}(s, \alpha) \triangleq \varepsilon_\alpha(s)$$

KAT Coalgebras

The final coalgebra is $(2^{GS}, \varepsilon, \delta)$ where

$$\varepsilon : 2^{GS} \to 2^{At} \qquad \delta : 2^{GS} \to (2^{GS})^{At \times \Sigma}$$

$$\varepsilon_\alpha(A) = \begin{cases} 1 & \alpha \in A \\ 0 & \alpha \notin A \end{cases} \qquad \delta_{\alpha p}(A) = \{x \mid \alpha p x \in A\}$$

The map

$$L : S \to 2^{GS} \qquad L(s) = \{x \mid \text{Accept}(s, x)\}$$

is the unique homomorphism to the final coalgebra

KAT Coalgebras

For $p \in \Sigma$ and $\alpha \in At$,

$$E_\alpha : \mathrm{Exp} \to 2 \qquad\qquad D_{\alpha\beta} : \mathrm{Exp} \to \mathrm{Exp}$$

$$\begin{aligned}
E_\alpha(e_1 + e_2) &= E_\alpha(e_1) + E_\alpha(e_2) & D_{\alpha p}(e_1 + e_2) &= D_{\alpha p}(e_1) + D_{\alpha p}(e_2) \\
E_\alpha(e_1 e_2) &= E_\alpha(e_1) \cdot E_\alpha(e_2) & D_{\alpha p}(e_1 e_2) &= D_{\alpha p}(e_1) e_2 + E_\alpha(e_1) D_{\alpha p}(e_2) \\
E_\alpha(e^*) &= 1 & D_{\alpha p}(e^*) &= D_{\alpha p}(e) e^* \\
E_\alpha(b) &= \begin{cases} 1 & \alpha \leq b \\ 0 & \alpha \not\leq b \end{cases} & D_{\alpha p}(b) &= 0 \\
E_\alpha(p) &= 0, \ p \in \Sigma & D_{\alpha p}(q) &= \begin{cases} 1 & q = p \\ 0 & q \neq p \end{cases}
\end{aligned}$$

The unique homomorphism to the final coalgebra is

$$L : \mathrm{Exp} \to 2^{GS} \qquad\qquad L(e) = \{\text{language represented by } e\}$$

NetKAT Coalgebra [Foster & al. 14]

A NetKAT coalgebra is a coalgebra (S, ε, δ) for the functor $FX = 2^{At \times At} \times X^{At \times At}$ where

$$\varepsilon : S \to 2^{At \times At} \qquad \delta : S \to S^{At \times At}$$

As an automaton,

$$\text{Accept}(s, \alpha p_\beta \text{ dup } x) \triangleq \text{Accept}(\delta_{\alpha\beta}(s), \beta x) \quad \text{Accept}(s, \alpha p_\beta) \triangleq \varepsilon_{\alpha\beta}(s)$$

The final coalgebra is

$$\varepsilon : 2^{NS} \to 2^{At \times At} \qquad \delta : 2^{NS} \to (2^{NS})^{At \times At}$$

$$\varepsilon_{\alpha\beta}(A) = \begin{cases} 1 & \alpha p_\beta \in A \\ 0 & \alpha p_\beta \notin A \end{cases} \qquad \delta_{\alpha\beta}(A) = \{\beta x \mid \alpha p_\beta \text{ dup } x \in A\}$$

NetKAT Coalgebra [Foster & al. 14]

$$\varepsilon : S \to 2^{At \times At} \qquad \delta : S \to S^{At \times At}$$

$$\varepsilon : S \to \mathrm{Mat}(At, 2) \qquad \delta : S \to \mathrm{Mat}(At, S)$$

$$E : \mathrm{Exp} \to \mathrm{Mat}(At, 2) \qquad D : \mathrm{Exp} \to \mathrm{Mat}(At, \mathrm{Exp})$$

$$E(e_1 + e_2) = E(e_1) + E(e_2) \qquad D(e_1 + e_2) = D(e_1) + D(e_2)$$

$$E(e_1 e_2) = E(e_1) \cdot E(e_2) \qquad D(e_1 e_2) = D(e_1) \cdot I(e_2) + E(e_1) \cdot D(e_2)$$

$$E(e^*) = E(e)^* \qquad D(e^*) = E(e)^* D(e) I(e^*)$$

$$E_{\alpha\beta}(b) = \begin{cases} 1 & \alpha = \beta \leq b \\ 0 & \text{otherwise} \end{cases} \qquad D(b) = 0$$

$$E_{\alpha\beta}(p) = \begin{cases} 1 & \beta = \alpha_p \\ 0 & \text{otherwise} \end{cases} \qquad D(p) = 0$$

$$E(\mathrm{dup}) = 0 \qquad D_{\alpha\beta}(\mathrm{dup}) = \begin{cases} \alpha & \beta = \alpha \\ 0 & \text{otherwise} \end{cases}$$

Kleene's Theorem for NetKAT [Foster & al. 14]

Theorem

1. *Let M be a finite NetKAT automaton. The set of strings in NS accepted by M is $L(e)$ for some NetKAT expression e.*
2. *For every NetKAT expression e, there is a deterministic NetKAT automaton M with at most $|At| \cdot 2^{\ell}$ states accepting $L(e)$, where ℓ is the number of occurrences of* dup *in e.*

A Bisimulation-Based Algorithm

To check $e_1 = e_2$, check bisimilarity using Brzozowski/Antimirov derivatives with the matrices E and D

- use an efficient sparse matrix representation involving a compact representation of sets of indices
- compute E matrices in advance
- an efficient representation of D using spines—spines of spines are spines, so repeated derivatives can be done by lookup
- use Hopcroft-Tarjan union-find data structure to represent bisimilarity classes
- represent sums as sets (Antimirov) and products as lists—gives addition mod ACI and multiplication mod associativity for free
- algorithm is competitive with state of the art on moderately large real-life examples

Conclusion

- Programming languages have a key role to play in emerging platforms for managing software-defined networks
- NetKAT is a high-level language for programming and reasoning about network behavior in the SDN paradigm
  - based on sound mathematical principles
  - formal denotational semantics, complete deductive system
  - efficient bisimulation-based decision procedure
- Future work:
  - global compilation and optimization
  - optimizations to reduce state space
  - applications: bandwidth guarantees, fault tolerance, load-balancing
  - probabilistic semantics
  - nondeterministic NetKAT
  - verification tools: Z3-based backend, automata-based backend

Thanks!