Probabilistic Program Equivalence for NetKAT

Steffen Smolka[1], David Kahn[1], Praveen Kumar[1], Nate Foster[1], Dexter Kozen[1], Alexandra Silva[2]
[1]Cornell University [2]University College London

Key Question: decide p ≡ q for probabilistic programs p and q
Key Result: decidable for history-free Probabilistic NetKAT

Motivation

NetKAT is a formal language for **programing, modeling, and reasoning** about the behavior of packet-switched networks.

Predicates (Boolean Algebra).
`t,u ::= 0 | 1 | f=n | t + u | t ; u | ¬t`

Programs (Kleene Algebra with Tests).
`p,q ::= t | f←n | p + q | p ; q | p* | dup`

Example. `pt=1; ip←10.0.0.1; (pt←1 + pt←2)`
"For all packets coming in at port 1, rewrite the IP address to 10.0.0.1 and forward the packet out of ports 1 and 2."

Many **network properties** can be naturally phrased as questions about **program equivalence** including waypointing, reachability, isolation, loop-freedom, etc. The language has a symbolic (worst-case PSPACE) **decision procedure**.

Goal. Develop a decision procedure for ProbNetKAT—i.e. NetKAT extended with a probabilistic choice operator $p \oplus_r q$.

Applications. Randomized & resilient routing algorithms, link failures, uncertainty about network model or inputs.

Probabilistic NetKAT Semantics

Programs denote **Markov kernels** over the **uncountable space** of packet history sets (2^H, $\mathfrak{B}$): $[\![p]\!] \in 2^H \to D(2^H)$.

Histories $h \in H = Pk \cdot Pk^*$ record trajectories of packets $\pi \in Pk$.

Continuous (atomless) distributions can be encoded.

Iteration p* is defined as sup in CPO ($D(2^H)$, ⊑) [Saheb-Djahromi].

Approach

1. Restrict to history-free fragment (large but finite space)

Syntax: remove `dup` (history-extension primitive).
Consider only packet (singleton-history) inputs $a \in 2^{Pk}$.

Practical Motivation: sufficient for many properties

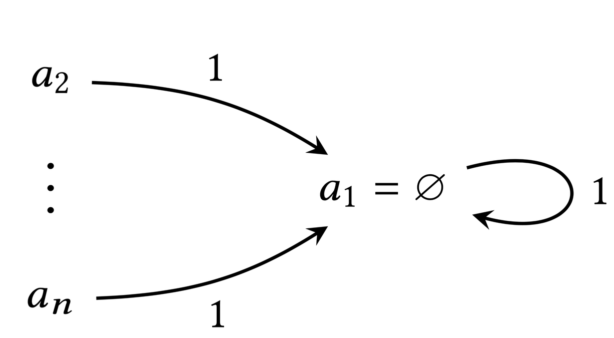
Theoretical Motivation: ingredient for full decision procedure (DP)

coalgebraic DP = derivatives + **DP for "observations"**

2. Reduce equivalence to checking equality of canonical form

"Big Step" Semantics: programs denote MCs over finite state space 2^{Pk}

$$\mathcal{B}[\![0]\!] = \begin{array}{c|cccc} & \varnothing & b_2 & \dots & b_n \\ \hline \varnothing & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & 1 & 0 & \cdots & 0 \end{array}$$



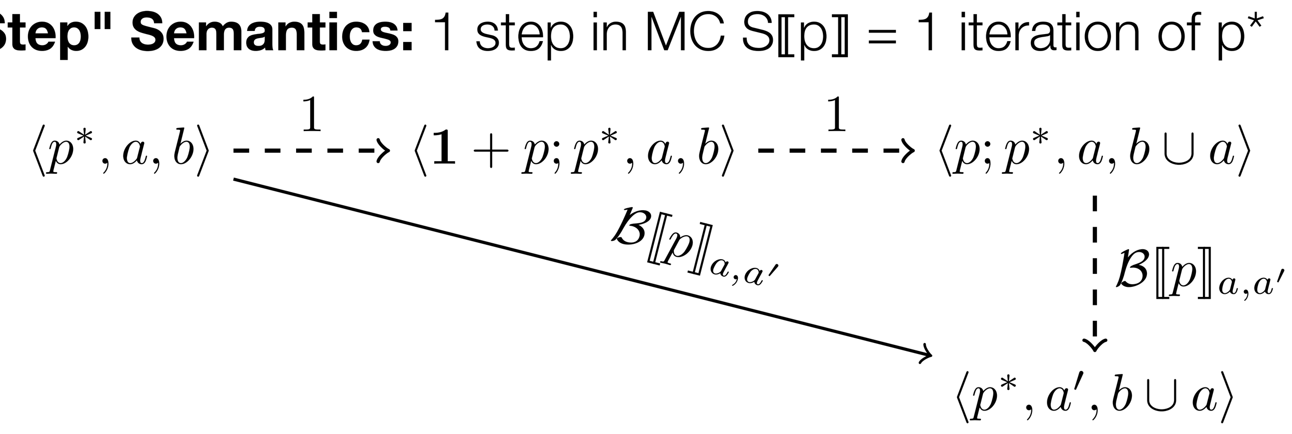
$\mathcal{B}[\![p]\!]_{a,b}$ = probability that p outputs $b \in 2^{Pk}$ on input $a \in 2^{Pk}$

Theorem (Sound & Complete). $[\![p]\!] = [\![q]\!]$ on 2^{Pk} ⟺ $\mathcal{B}[\![p]\!] = \mathcal{B}[\![q]\!]$

3. Compute canonical form using absorbing Markov chains

Challenge. How to compute $\mathcal{B}[\![p^*]\!] := \lim \mathcal{B}[\![p^{(n)}]\!]$?

"Small Step" Semantics: 1 step in MC $\mathcal{S}[\![p]\!]$ = 1 iteration of p*



States are of the form <program, input set, output accumulator>

Approach (continued)

Observation. Output accumulator is **monotonically increasing** and **eventually saturates**.

→ **Collapsing saturated** states modulo equivalent accumulators, yields an **absorbing** MC.

→ **Unique stationary distribution** exists, can be given in **closed form**.

Theorem. $\mathcal{B}[\![p^*]\!]$ = absorption probabilities for collapsed small-step MC.

Corollary. $\mathcal{B}[\![p]\!]$ is computable for all p.

Corollary. Program equivalence for history-free ProbNetKAT is decidable

Case Study: Resilient Routing

Resilient routing algorithms try to delivery packets despite links failures.
Formally, they are functions from

- the packet's destination (`dst`)
- the port at which the packet entered the switch (`pt`)
- the list of available outgoing links ($\mathtt{up}_l \in \{0,1\}$ for each link l)

to the outport through which the packet will be forwarded.

ProbNetKAT specification of desired end-to-end property:

$$\mathsf{teleport} \triangleq \sum_d \mathtt{dst}{=}d; \mathtt{sw}{\leftarrow}d$$

"Packets get delivered to their destination."

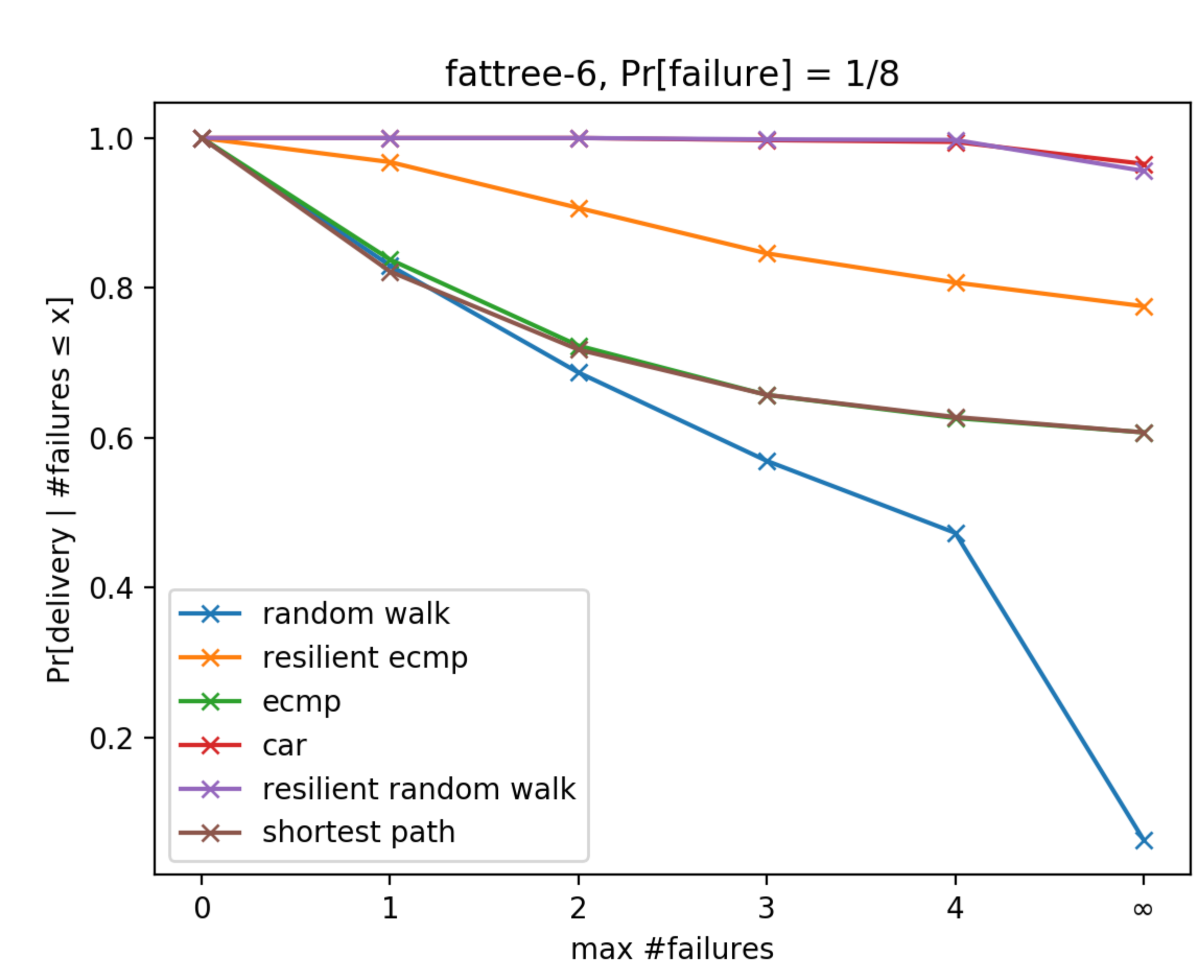
ProbNetKAT model of resilient routing algorithms:

$$\mathsf{model} \triangleq \ \mathbf{while}\ \neg\mathsf{at\_destination}\ \mathbf{do}\ \ \mathsf{initialize\_up\_bits}; \mathsf{route}; \mathsf{topology}$$

$$\mathsf{topology} \triangleq \sum_\ell \left[\begin{array}{l}\mathbf{if}\ \mathtt{up}_\ell{=}1; \mathtt{sw}{=}src\_sw(\ell); \mathtt{pt}{=}src\_pt(\ell)\ \mathbf{then} \\ \quad \mathtt{sw}{\leftarrow}dst\_sw(\ell); \mathtt{pt}{\leftarrow}dst\_pt(\ell) \\ \mathbf{else\ drop}\end{array}\right]$$

Checking Properties:

- **Correctness:** $\mathsf{model}_{\mathsf{no\_link\_failures}} \equiv \mathsf{teleport}$?
- **k-Resilience:** $\mathsf{model}_{\mathsf{at\_most\_k\_link\_failures}} \equiv \mathsf{teleport}$?



Open Questions & Future Work

1. Decision procedure for full language?
Challenges: uncountable space, continuous distributions
Have "**language model**" $L[\![p]\!] \in D(2^{Pk \cdot Pk^* \cdot Pk})$ and DC for "observations."
Exploring **derivatives** and suitable **automata model**.

2. Other practical applications?
Challenges: scalability of implementation, expressivity of language
Add Bayesian inference to determine likely sources of failures?