Probabilistic Program Equivalence for NetKAT

Steffen Smolka
Cornell University

David Kahn
Cornell University

Praveen Kumar
Cornell University

Nate Foster
Cornell University

Dexter Kozen
Cornell University

Alexandra Silva
University College London

Abstract

We study the problem of deciding program equivalence in the context of Probabilistic NetKAT, a formal language for reasoning about the behavior of packet-switched networks. We show that the problem is decidable for the history-free fragment of the language, and discuss a path toward a decision procedure for the full language. The main challenge lies in reasoning about iteration, which we address by a reduction to finite-state absorbing Markov chains. We also describe an OCaml prototype that we have used to reason about probabilistic network programs.

1 Motivation

NetKAT [1] is a language based on Kleene algebra (KA) and Kleene algebra with tests (KAT) that can be used to program networks and reason about their properties. It comes with a rich theory (a denotational semantics, a sound and complete axiomatization, both language and automata models) and sophisticated tools (a fully-automatic decision procedure, an efficient compiler). Probabilistic NetKAT [4] extends the language with a probabilistic choice operator and a semantics based on Markov kernels, allowing it to model features such as faulty links, randomized routing algorithms, and uncertainty about input packets.

Why study program equivalence? Many network properties can be naturally and conveniently phrased as questions about program equivalence in NetKAT. For example, NetKAT equivalence has been used to reason about essential properties such as waypointing, reachability, isolation, and loop freedom, as well as for the validation and verification of compiler transformations. This work aims to extend this approach to the probabilistic setting.

Why is it challenging? Because ProbNetKAT has an iteration operator, it is possible to write programs that generate continuous distributions over the uncountable space of packet history sets. This makes reasoning about convergence non-trivial, and raises the issue of representing infinitary objects in an implementation. Prior work developed a domain-theoretic semantics that provides notions of approximation and continuity, which can be used to reason about programs using distributions with finite-support [4]. However, it left program equivalence as an open problem. We settle this question positively for the history-free fragment of the language, in which distributions have finite-support but iteration may still converge only in the limit. Computing the analytical value of these limits precisely is the main challenge we address.

Why not apply standard results? Although there is a lot of related work (e.g., probabilistic languages based on regular operators [3], automata [5], and model checkers [2]), ProbNetKAT's semantics seems to be sufficiently different that previous results do not apply since programs in the history-free fragment do not denote sequences in any obvious sense and correspond to one-state automata.

Our Approach. Our decision procedure for the history-free fragment of ProbNetKAT follows a general approach: we map programs to canonical representations for which checking equivalence is straightforward. Specifically, we define a *big-step* semantics that interprets each program as a finite stochastic matrix—equivalently, a Markov chain that transitions from input to output in a single step. Equivalence is trivially decidable on this representation, but some care is needed to compute the matrix in the case of iteration—intuitively, it needs to capture the result of an infinite stochastic process. We address this by embedding the system in a second Markov chain with a larger state space that models iteration in the spirit of a *small-step* semantics. This chain can be transformed to an absorbing Markov chain, which admits a closed form analytic solution using elementary matrix operations that represents the limit of the iteration. We have proved the soundness of this approach.

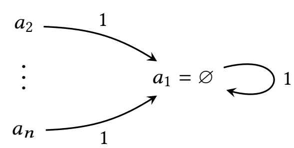
2 Results

Preliminaries. A history is a non-empty sequence of packets that encodes the trajectory of a single packet through the network: it can be thought of as a log of the packet's activity. On a semantic level, ProbNetKAT programs p denote functions of type $[\![p]\!] \in 2^{\mathsf{H}} \to \mathcal{D}(2^{\mathsf{H}})$ that map sets of input packet histories to distributions on sets of output packet histories. On a syntactic level, the language consists of a Boolean algebra (which enables using predicates to classify incoming packets based on their contents); the well-known regular operators $+$, $\cdot$, and * (which enables specifying regular forwarding paths); a probabilistic choice operator $\oplus_r$; and a modification primitive $f \leftarrow n$ (which allows rewriting the f-field of incoming packets to n). The primitive dup is a "logging" command that records the current state of the packet in the history. It is important to note that the notion of *history* is purely semantic: in a physical network, only the current packet needs to exist.

The price of history. Histories facilitate reasoning about network-wide paths, but they come at a hefty mathematical cost: there are as many histories as natural numbers, so the powerset 2^{H} is as large as the set of real numbers. As a result, ProbNetKAT's denotational semantics requires some heavy machinery including a measurable space $(2^{\mathsf{H}}, \mathcal{B})$ to deal with continuous distributions, and a CPO $(\mathcal{D}(2^{\mathsf{H}}), \sqsubseteq)$ of distributions to enable defining the semantics of iteration as a least fixpoint. To avoid this complexity, this work focuses on the *history-free fragment* of the language: it is obtained by removing the dup primitive and working only with the finite set of packets (*i.e.*, singleton histories).

Big-step semantics. In the history-free setting, we can interpret a program p as a finite stochastic matrices $\mathcal{B}[\![p]\!] \in \mathbb{S}(2^{\mathsf{Pk}})$, *i.e.* a matrix indexed by packet sets $a, b \in 2^{\mathsf{Pk}}$ such that $\mathcal{B}[\![p]\!]_{a,b}$ denotes the probability that the program produces output b on input a. At a high-level, deterministic program primitives map to simple $(0,1)$-matrices, and program operators map to operations on matrices. For example, the program primitive drop is interpreted as the matrix

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



that puts all probability mass in the $\varnothing$-column; the primitive skip is the identity matrix; sequential composition is given by matrix product; and probabilistic choice is given by convex sum. Such a matrix represents a Markov chain over the state space 2^{Pk} that, intuitively, transitions from an initial state a corresponding to the set of input packets in a single step straight to the set of output packets b—thus the name *big step* semantics.

We define the big-step semantics of iteration as the limit

$$\mathcal{B}[\![p^*]\!]_{a,b} \triangleq \lim_{n\to\infty} \mathcal{B}[\![p^{(n)}]\!]_{a,b} \tag{1}$$

where $p^{(n)}$ denotes the n-th unrolling of p^*. We then prove this semantics is sound with respect to the denotational semantics: $\mathcal{B}[\![p]\!]_{a,b} = [\![p]\!](a)(\{b\})$. The result is nontrivial because the sequence of measures $\mu_n = [\![p^{(n)}]\!](a)$ does not in general converge pointwise to $\mu = [\![p^*]\!](a)$. This crucially relies on the assumption that p is dup-free, and requires some additional results about the denotational semantics. As a corollary, we reduce the equivalence problem on dup-free programs to checking equality of finite-dimensional matrices. The remaining problem is *how* to compute these matrices, which requires computing a limit in general.

Example. Suppose packets have a single binary field f, and consider the program $p^* \triangleq (\mathsf{skip} \oplus_r f \leftarrow 1)^*$. The program describes an infinite stochastic process that repeatedly flips a biased coin and either outputs the input packets unmodified with probability r, or outputs the input packets after setting their f-field to 1 with probability $1-r$. Recalling (1), we can approximate this infinite process using its finite unrollings and then take the limit:

$$\mathcal{B}[\![p^{(n)}]\!] = \begin{array}{c} \\ \varnothing \\ 0 \\ 1 \\ 0,1 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} \varnothing & 0 & 1 & 0,1 \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & r^n & 0 & 1-r^n \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \end{array} \Rightarrow \mathcal{B}[\![p^*]\!] = \begin{array}{c} \\ \varnothing \\ 0 \\ 1 \\ 0,1 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} \varnothing & 0 & 1 & 0,1 \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

The problem is how to compute these limits in general.

Small-step semantics. To solve this problem, we define a *small-step* semantics $\mathcal{S}[\![p]\!] \in \mathbb{S}(2^{\mathsf{Pk}} \times 2^{\mathsf{Pk}})$ in terms of $\mathcal{B}[\![p]\!]$. Intuitively, one step in the Markov chain $\mathcal{S}[\![p]\!]$ models one iteration of p^*. States of the chain are pairs $\langle a, b\rangle$, where a is the current set of packets, and b is an accumulator tracking the packets output so far. We prove that $\mathcal{B}[\![p^{(n)}]\!]_{a,b} = \sum_{a'} \mathcal{S}[\![p]\!]^{n+1}_{(a,\varnothing),(a',b)}$. With care, we can transform $\mathcal{S}[\![p]\!]$ into and *absorbing Markov chain*, whose unique stationary distribution can be given in closed-form. This allows us compute the required limit analytically.

Language model. While the equivalence problem for the full language remains open, we show that ProbNetKAT programs are fully characterized by a language model $\mathcal{L}[\![p]\!] \in \mathcal{D}(2^{\mathsf{Pk}\cdot\mathsf{Pk}^*\cdot\mathsf{Pk}})$. That is, programs can be interpreted as distributions over languages (*i.e.*, packet sequence sets). This gives hope that we may be able to interpret ProbNetKAT programs as suitable automata, much in the way deterministic NetKAT programs can be interpreted as NetKAT automata.

3 Discussion

We are pursuing several directions in ongoing work. On the practical side, we are exploring applications of ProbNetKAT for reasoning about probabilistic network properties, such as correct load balancing and robustness to failures. On the theoretical side, we are investigating a decision procedure for the full language. The decision procedure for deterministic NetKAT follows a coalgebraic approach: it translates programs to automata using derivatives, and then checks whether the automata are bisimilar, where states are bisimilar iff they have the same "observations" and transition to bisimilar states. NetKAT "observations" are isomorphic to dup-free programs (up to program equivalence) so this requires deciding equivalence for dup-free programs. Hence, the only thing missing for a decision procedure for the full language is a suitable definition of the derivative.

References

[1] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. 2014. NetKAT: Semantic Foundations for Networks. In *POPL*. 113–126.

[2] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (LNCS)*, G. Gopalakrishnan and S. Qadeer (Eds.), Vol. 6806. Springer, 585–591. https://doi.org/10.1007/978-3-642-22110-1_47

[3] A. K. McIver, E. Cohen, C. Morgan, and C. Gonzalia. 2008. Using Probabilistic Kleene Algebra pKA for Protocol Verification. *J. Logic and Algebraic Programming* 76, 1 (2008), 90–111.

[4] Steffen Smolka, Praveen Kumar, Nate Foster, Dexter Kozen, and Alexandra Silva. 2017. Cantor Meets Scott: Semantic Foundations for Probabilistic Networks. In *POPL 2017*. https://doi.org/10.1145/3009837.3009843

[5] Wen-Guey Tzeng. 1992. A Polynomial-Time Algorithm for the Equivalence of Probabilistic Automata. *SIAM J. Comput.* 21, 2 (1992), 216–227. https://doi.org/10.1137/0221017