Review and applications of Kleene Algebra with Tests

Sergey Staroletov

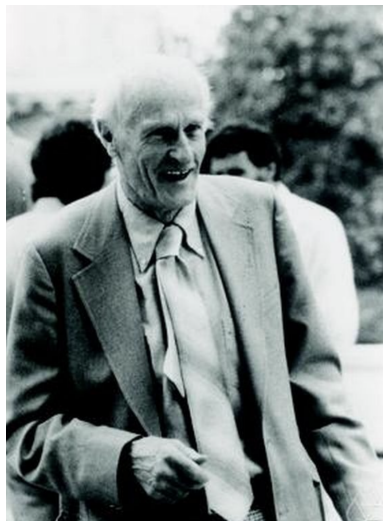
serg_soft@mail.ru

STEP seminar / online
12 April 2023

12, April[1]

[1] * = Kleene star

Stephen Cole Kleene



- January 5, 1909 – January 25, 1994
- Was awarded a Ph.D. in mathematics from Princeton University in 1934, where his thesis, entitled **A Theory of Positive Integers in Formal Logic**, was supervised by **Alonzo Church**.

Selected publications by Kleene

- 1935. Stephen Cole Kleene (Jan 1935). "A Theory of Positive Integers in Formal Logic.". American Journal of Mathematics. 57 (1): 153–173. doi:10.2307/2372027. JSTOR 2372027.
- 1935. Stephen Cole Kleene; J.B. Rosser (Jul 1935). "The Inconsistency of Certain Formal Logics". Annals of Mathematics. 2nd Series. 36 (3): 630–636. doi:10.2307/1968646. JSTOR 1968646.
- 1936. "General recursive functions of natural numbers". Mathematische Annalen (112): 727–742. 1936. 1936. "
- λ-definability and recursiveness". Duke Mathematical Journal. 2 (2): 340–352. 1936.
- 1938. "On Notations for Ordinal Numbers"(PDF). Journal of Symbolic Logic. 3 (4): 150–155. 1938. doi:10.2307/2267778. JSTOR 2267778. S2CID 34314018.
- 1943. "Recursive predicates and quantifiers". Transactions of the American Mathematical Society. 53 (1): 41–73. Jan 1943. doi:10.1090/S0002-9947-1943-0007371-8.
- 1951. Kleene, Stephen Cole (15 December 1951). "Representation of Events in Nerve Nets and Finite Automata"(PDF). No. RM-704. The RAND Corporation.

Selected publications by Kleene

- 1965 (with Richard Eugene Vesley). The Foundations of Intuitionistic Mathematics. North-Holland.
- 1967. Mathematical Logic. John Wiley & Sons. Dover reprint, 2002. ISBN 0-486-42533-9.
- 1981. "Origins of Recursive Function Theory"in Annals of the History of Computing 3, No. 1.
- 1987. "Reflections on Church's thesis". Notre Dame Journal of Formal Logic. 28 (4): 490–498. Oct 1987. doi:10.1305/ndjfl/1093637645.

A Kleene star

Given a set V. Define:

- $V^0 = \{\varepsilon\}$ (the language consisting only of the empty string),
- $V^1 = V$
- and define recursively the set $V^{i+1} = \{wv : w \in V^i \text{ and } v \in V\}$ for each $i > 0$.

The definition of Kleene star on V is:

$$V^* = \bigcup_{i \geq 0} V^i = V^0 \cup V^1 \cup V^2 \cup V^3 \cup V^4 \cup \cdots .$$

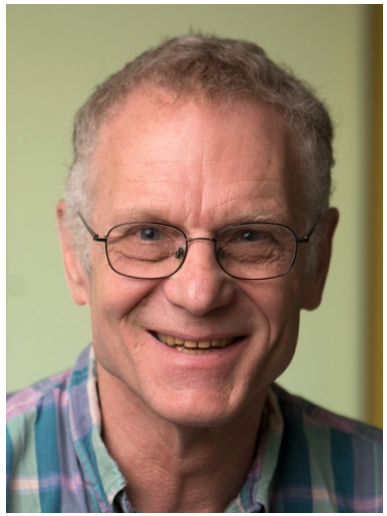
Example of Kleene star applied to set of strings:

{"ab","c"}* = { ϵ, "ab", "c"," 'abab", "abc", "cab", "cc", "abababab", "ababc", "abcab", "abcc", "cabab", "cabc", "ccab", "ccc", ...}.

Generalization:

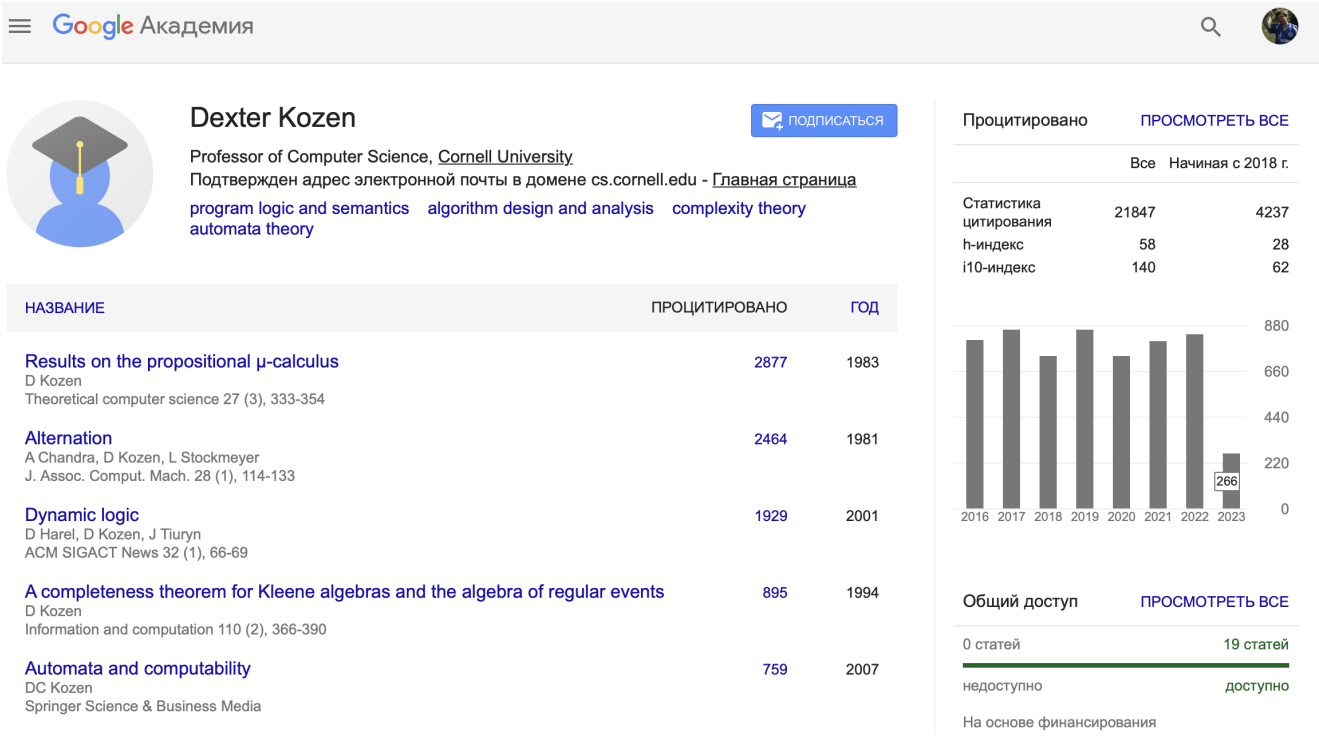
- Strings form a monoid with concatenation as the binary operation and ϵ the identity element. The Kleene star is defined for any monoid, not just strings. precisely, let (M, ·) be a monoid, and S $\subseteq$ M. Then S* is the smallest submonoid of M containing S; that is, S* contains the neutral element of M, the set S, and is such that if x,y $\in$ S*, then x · y $\in$ S*.

Dexter Kozen



- Born December 20, 1951
- He is an American theoretical computer scientist (Professor in Engineering at Cornell University)
- `https://www.cs.cornell.edu/~kozen/`
- Research interests: Logics and semantics of programming languages, algorithms and complexity, especially complexity of decision problems in logic and algebra, probabilistic computation

Dexter Kozen



Google Академия

Dexter Kozen

ПОДПИСАТЬСЯ

Professor of Computer Science, Cornell University
Подтвержден адрес электронной почты в домене cs.cornell.edu - Главная страница

program logic and semantics algorithm design and analysis complexity theory automata theory

НАЗВАНИЕ	ПРОЦИТИРОВАНО	ГОД
Results on the propositional μ-calculus D Kozen Theoretical computer science 27 (3), 333-354	2877	1983
Alternation A Chandra, D Kozen, L Stockmeyer J. Assoc. Comput. Mach. 28 (1), 114-133	2464	1981
Dynamic logic D Harel, D Kozen, J Tiuryn ACM SIGACT News 32 (1), 66-69	1929	2001
A completeness theorem for Kleene algebras and the algebra of regular events D Kozen Information and computation 110 (2), 366-390	895	1994
Automata and computability DC Kozen Springer Science & Business Media	759	2007

Процитировано ПРОСМОТРЕТЬ ВСЕ

	Все	Начиная с 2018 г.
Статистика цитирования	21847	4237
h-индекс	58	28
i10-индекс	140	62

Общий доступ ПРОСМОТРЕТЬ ВСЕ

0 статей 19 статей

недоступно доступно

На основе финансирования

Dexter Kozen

Software

- ECC Project: ECC is a simple and efficient approach to the certification of compiled code downloaded from an untrusted source. It can ensure a basic but nontrivial level of code safety, including control flow safety, memory safety, and stack safety in a way that is simple, efficient, and (most importantly) relatively painless to incorporate into existing compilers.
- KAT Interactive Theorem Prover: KAT-ML is an interactive theorem prover for Kleene algebra with tests. KAT has been applied successfully in various low-level verification tasks involving communication protocols, basic safety analysis, source-to-source program transformation, concurrency control, compiler optimization, and static analysis.
- A "lite" version of the KAT Interactive Theorem Prover (OCaml)
- The CoCaml Project CoCaml is a strongly-typed call-by-value functional language in the ML family. It is syntactically almost identical to OCaml, except that variables are mutable (can be rebound) as with the Scheme set! construct.
- Natural deduction interactive theorem prover for 1st order logic (OCaml)

Dexter Kozen

Awards:

- Outstanding Innovation Award, IBM Corporation) (1974)
- POPL Distinguished Paper Award for the paper Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time (2020)
- Alonzo Church Award, for his fundamental work on developing the theory and applications of Kleene Algebra with Tests, an equational system for reasoning about iterative programs.

Alonzo Church Award

- The Alonzo Church Award for Outstanding Contributions to Logic and Computation was established in 2015 by the ACM Special Interest Group for Logic and Computation (SIGLOG), the European Association for Theoretical Computer Science (EATCS), the European Association for Computer Science Logic (EACSL), and the Kurt Gödel Society (KGS). The award is for an outstanding contribution represented by a paper or small group of papers within the past 25 years.
- `https://www.eacsl.org/alonzo-church-award/`
- The basic eligibility criterion is that the contribution has appeared in a paper or papers within the past 25 years. When a paper has appeared in a conference and then in another refereed publication format, such as a journal or a book chapter, the date of the refereed publication will determine the cut-off date. Thus, for the first award, which was presented in summer 2016, the cut-off date was Jan. 1, 1991. In addition, the contribution should not yet have received recognition via a major award, such as the Turing Award, the Kanellakis Award, or the Goedel Prize. (The nominee(s) may, however, have received such awards for other contributions.)

Alonzo Church Award'22

The ACM Special Interest Group on Logic (SIGLOG), the European Association for Theoretical Computer Science (EATCS), the European Association for Computer Science Logic (EACSL), and the Kurt Gödel Society (KGS) are pleased to announce that the **2022 Alonzo Church Award** for Outstanding Contributions to Logic and Computation is given to **Dexter Kozen** for his fundamental work on developing the theory and applications of Kleene Algebra with Tests, an equational system for reasoning about iterative programs, published in: **Kleene Algebra with Tests. ACM Transactions on Programming Languages and Systems 19(3): 427-443 (1997)**

This work on Kleene Algebra with Tests (KAT) is one of the high points among remarkable contributions of Dexter Kozen to logics of programs. It is a culmination of a series of articles by Dexter Kozen that combines Kleene Algebra (the algebra of regular expressions) with Boolean Algebra (the tests). Together, the terms of the two algebras are capable of representing while programs, and their combined equational theory is capable of proving a wide range of important properties of programs. Although reasoning in KAT under arbitrary commuting conditions is undecidable, Kozen observes that when the commuting conditions are limited to including tests, it is decidable and in PSPACE. He illustrates the power of KAT with these decidable commuting conditions by proving a well-known folk theorem: Every while program can be simulated by a program with just one loop. KAT has been successfully applied to a variety of problems over the past 25 years, including modeling and reasoning about packet-switched networks.

Introduction

Kleene algebra is an algebraic structure [2]

$$(K, +, \cdot, *, 0, 1)$$

$$p + (q + r) = (p + q) + r \tag{1}$$

$$p + q = q + p \tag{2}$$

$$p + 0 = p \tag{3}$$

$$p + p = p \tag{4}$$

$$p(qr) = (pq)r \tag{5}$$

$$1p = p \tag{6}$$

$$p1 = p \tag{7}$$

[2] Kozen, D. C. 1994. A completeness theorem for Kleene algebras and the algebra of regular events. Infor. and Comput. 110, 2 (May), 366–390.

$$p(q + r) = pq + pr \tag{8}$$

$$(p + q)r = pr + qr \tag{9}$$

$$0p = 0 \tag{10}$$

$$p0 = 0 \tag{11}$$

$$1 + pp^* = p^* \tag{12}$$

$$1 + p^*p = p^* \tag{13}$$

$$q + pr \leq r \rightarrow p^*q \leq r \tag{14}$$

$$q + rp \leq r \rightarrow qp^* \leq r \tag{15}$$

where $\leq$ refers to the natural partial order on K: $p \leq q \leftrightarrow p + q = q$.

Instead of (14) and (15), we might take the equivalent axioms

$$pr \leq r \rightarrow p^* r \leq r \tag{16}$$

$$rp \leq r \rightarrow rp^* \leq r \tag{17}$$

Axioms (1)–(11) say that the structure is an idempotent semiring under $+$, $\cdot$, 0, and 1, and the remaining axioms (12)–(17) say essentially that * behaves like the Kleene star operator of formal language theory or the reflexive transitive closure operator of relational algebra.
Idempotence is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.

Kleene algebra with tests (KAT)

Kleene algebra with tests (KAT)

To accommodate tests, we introduce the following variant of Kleene algebra. A Kleene algebra with tests is a two-sorted algebra

$$(K, B, +, \cdot, *, 0, 1,)$$

where B $\subseteq$ K, and is a unary operator defined only on B such that (K, +, $\cdot$, *, 0, 1) is a Kleene algebra and (B,+,$\cdot$, $^-$,0,1) is a Boolean algebra.

We work with a Pascal-like programming language with:

- sequential composition $p\ ;\ q$,
- a conditional test ***if*** b ***then*** p ***else*** q,
- and a looping construct ***while*** b ***do*** p.

These constructs are modeled in Kleene algebra with tests as follows:

- p;q = pq
- **if** b **then** p **else** q = bp+b^-q
- **if** b **then** p = bp+b^-
- **while** b **do** p = $(bp)^* b^-$.

A Folk theorem[3]

Every while program can be simulated by a while program with at most one while loop, provided extra Boolean variables are allowed.
A plan for the proof:

1. An Example
2. A Normal Form Theorem
3. Conditional
4. Nested Loops
5. Eliminating Postcomputations
6. Composition

[3] Harel, D. 1980. On folk theorems. Commun. ACM 23, 7 (July), 379–389.

An Example

To illustrate this technique, consider the simple program

$$\begin{array}{l}\textbf{if } b \textbf{ then begin } p\,;q \textbf{ end}\\ \quad\textbf{else begin } p\,;r \textbf{ end}\end{array} \tag{21}$$

If the value of b were preserved by p, then we could rewrite this program more simply as

$$p\,;\textbf{if } b \textbf{ then } q \textbf{ else } r \tag{22}$$

Expressed in the language of Kleene algebra, the equivalence of (21) and (22) becomes the equation

$$bpq + \bar{b}pr \;=\; p(bq + \bar{b}r).$$

This identity can be established by simple equational reasoning:

$$\begin{aligned} p(bq + \bar{b}r) &= pbq + p\bar{b}r && \text{by (8)}\\ &= bpq + \bar{b}pr && \text{by the commutativity assumptions.}\end{aligned}$$

A Normal Form Theorem

A program is in *normal form* if it is of the form

$$p\,;\,\textbf{while}\ b\ \textbf{do}\ q \tag{24}$$

where p and q are **while** free. The subprogram p is called the *precomputation* of the normal form.

In our approach, the folk theorem takes the following form:

THEOREM 3.2.1. *Every* **while** *program, suitably augmented with finitely many new subprograms of the form* $s\,;bc+\bar{b}\bar{c}$*, is equivalent to a* **while** *program in normal form, reasoning in Kleene algebra with tests under certain commutativity assumptions* $cp = pc$.

Conditional

We first show how to deal with a conditional containing programs in normal form in its **then** and **else** clauses. Consider the program

$$\begin{array}{l}\textbf{if } b \textbf{ then begin } p_1 \textbf{ ; while } d_1 \textbf{ do } q_1 \textbf{ end}\\ \quad\textbf{else begin } p_2 \textbf{ ; while } d_2 \textbf{ do } q_2 \textbf{ end.}\end{array}$$

We introduce a new atomic program s and test c and assume that c commutes with p_1, p_2, q_1, and q_2.

Under these assumptions, we show that the programs

$$\begin{array}{l}s\,;\, bc + \bar{b}\bar{c}\,;\\ \textbf{if } b \textbf{ then begin } p_1 \textbf{ ; while } d_1 \textbf{ do } q_1 \textbf{ end}\\ \quad\textbf{else begin } p_2 \textbf{ ; while } d_2 \textbf{ do } q_2 \textbf{ end}\end{array} \tag{25}$$

and

$$\begin{array}{l}s\,;\, bc + \bar{b}\bar{c}\,;\\ \textbf{if } c \textbf{ then } p_1 \textbf{ else } p_2\,;\\ \textbf{while } cd_1 + \bar{c}d_2 \textbf{ do}\\ \quad\textbf{if } c \textbf{ then } q_1 \textbf{ else } q_2\end{array} \tag{26}$$

are equivalent.

Nested Loops

We show that the program

$$\begin{array}{l}\textbf{while } b \textbf{ do begin}\\ \qquad\qquad p\,;\\ \qquad\qquad \textbf{while } c \textbf{ do } q\\ \qquad\quad \textbf{end}\end{array} \tag{34}$$

is equivalent to the program

$$\begin{array}{l}\textbf{if } b \textbf{ then begin}\\ \qquad\qquad p\,;\\ \qquad\qquad \textbf{while } b + c \textbf{ do}\\ \qquad\qquad\quad \textbf{if } c \textbf{ then } q \textbf{ else } p\\ \qquad\quad \textbf{end.}\end{array} \tag{35}$$

This construction transforms a pair of nested **while** loops to a single **while** loop inside a conditional test. No commutativity conditions are assumed in the proof.

Eliminating Postcomputations

We wish to show that a postcomputation occurring after a **while** loop can be absorbed into the **while** loop. Consider a program of the form

$$\textbf{while } b \textbf{ do } p\,;q. \tag{39}$$

We can assume without loss of generality that b, the test of the **while** loop, commutes with the postcomputation q. If not, introduce a new test c that commutes with q along with an atomic program s that intuitively sets the value of c to b, and insert $s\,;bc+\overline{b}\overline{c}$ before the loop and in the loop after the body. We then claim that the two programs

$$s\,;bc+\overline{b}\overline{c}\,;\textbf{while } b \textbf{ do begin } p\,;s\,;bc+\overline{b}\overline{c} \textbf{ end}\,;q \tag{40}$$

$$s\,;bc+\overline{b}\overline{c}\,;\textbf{while } c \textbf{ do begin } p\,;s\,;bc+\overline{b}\overline{c} \textbf{ end}\,;q \tag{41}$$

are equivalent. This allows us to replace (40) with (41), for which the commutativity assumption holds.

Under the assumption that b and q commute, we show that (39) is equivalent to the program

$$\begin{array}{l}\textbf{if } \overline{b} \textbf{ then } q\\ \quad\textbf{else}\ \ \textbf{while } b \textbf{ do begin}\\ \qquad\qquad\qquad p\,;\\ \qquad\qquad\qquad \textbf{if } \overline{b} \textbf{ then } q\\ \qquad\qquad \textbf{end.}\end{array} \tag{42}$$

Note that if p and q are **while** free, then (42) consists of a program in normal form inside a conditional, which can be transformed to normal form using the construction of Section 3.3.

Composition

The composition of two programs in normal form

$$\begin{array}{l} p_1\,; \\ \textbf{while } b_1 \textbf{ do } q_1\,; \\ p_2\,; \\ \textbf{while } b_2 \textbf{ do } q_2 \end{array} \tag{45}$$

can be transformed to a single program in normal form. The resulting program looks like

$$\begin{array}{l} \textbf{if } \overline{b} \textbf{ then while } c \textbf{ do } q \\ \quad \textbf{else while } b \textbf{ do begin} \\ \qquad\qquad p\,; \\ \qquad\qquad \textbf{if } \overline{b} \textbf{ then while } c \textbf{ do } q \\ \qquad \textbf{end.} \end{array} \tag{47}$$

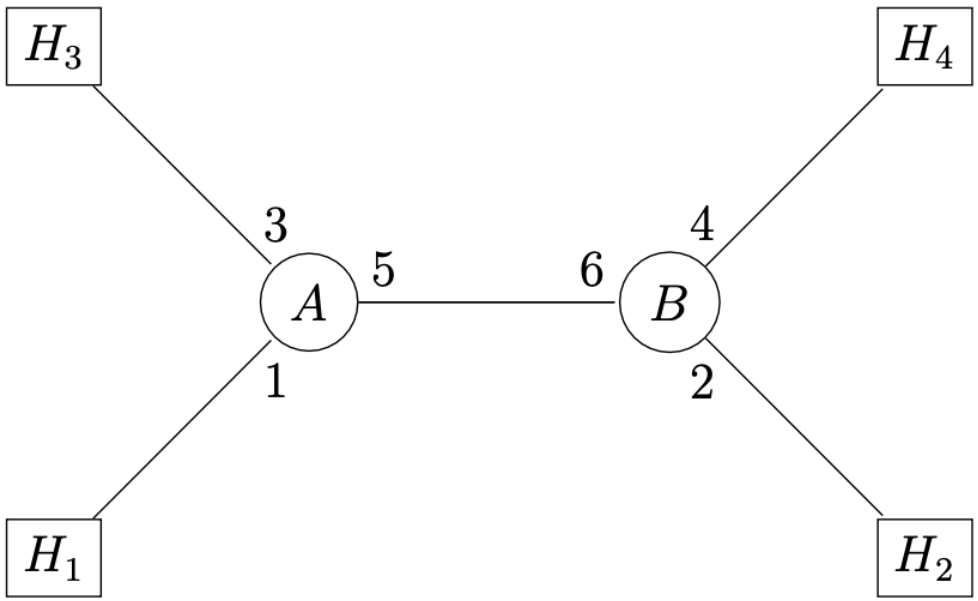
The transformations give a systematic method for moving while loops outside of any other programming construct. By applying these transformations inductively from the innermost loops outward, we can transform any program into a program in normal form.

Applications of KAT

- Caltais, G., & Tunç, H. C. (2021). Explaining safety failures in NetKAT. Journal of Logical and Algebraic Methods in Programming, 121, 100676.

  This work introduces a concept of explanations with respect to the violation of safe behaviours within software defined networks (SDNs) expressible in NetKAT. The latter is a network programming language based on a well-studied mathematical structure, namely, Kleene Algebra with Tests (KAT). Amongst others, the mathematical foundation of NetKAT gave rise to a sound and complete equational theory. In our setting, a safe behaviour is characterised by a NetKAT policy, or program, which does not enable for- warding packets from an ingress i to an undesirable egress e. We show how explanations for safety violations can be derived in an equational fashion, according to a modification of the existing NetKAT axiomatisation.

NetKAT



$$t \triangleq pt = 5 \cdot pt \leftarrow 6 + pt = 6 \cdot pt \leftarrow 5 + pt = 1 + pt = 2 + pt = 3 + pt = 4 \quad (1)$$

Applications of KAT

- Smolka, S., Kumar, P., Foster, N., Kozen, D., & Silva, A. (2017, January). Cantor meets Scott: semantic foundations for probabilistic networks. In Proceedings of the 44th
  ACM SIGPLAN Symposium on Principles of Programming Languages (pp. 557-571).

  ProbNetKAT is a probabilistic extension of NetKAT with a de- notational semantics based on Markov kernels. The language is expressive enough to generate continuous distributions, which raises the question of how to compute effectively in the language. This paper gives an new characterization of ProbNetKAT's semantics using domain theory, which provides the foundation needed to build a practical implementation. We show how to use the semantics to approximate the behavior of arbitrary ProbNetKAT programs using distributions with finite support. We develop a prototype implementation and show how to use it to solve a variety of problems including characterizing the expected congestion induced by different rout- ing schemes and reasoning probabilistically about reachability in a network.

NetKAT

Example. Consider the programs

$$p_1 \triangleq \mathsf{pt}{=}1 \,;\, (\mathsf{pt}{\leftarrow}2 \,\&\, \mathsf{pt}{\leftarrow}3)$$
$$p_2 \triangleq (\mathsf{pt}{=}2 \,\&\, \mathsf{pt}{=}3) \,;\, \mathsf{dst}{\leftarrow}10.0.0.1 \,;\, \mathsf{pt}{\leftarrow}1$$

The first program forwards packets entering at port 1 out of ports 2 and 3—a simple form of multicast—and drops all other packets. The second program matches on packets coming in on ports 2 *or* 3, modifies their destination to the IP address 10.0.0.1, and sends them out through port 1. The program $p_1 \,\&\, p_2$ acts like p_1 for packets entering at port 1, and like p_2 for packets entering at ports 2 or 3.

Applications of KAT

- Platzer A. Logical analysis of hybrid systems: proving theorems for complex dynamics. Springer Science & Business Media, 2010.

  Hybrid systems are models for complex physical systems and have become a widely used concept for understanding their behavior. Many applications are safety-critical, including car, railway, and air traffic control, robotics, physical–chemical process control, and biomedical devices. Hybrid systems analysis studies how we can build computerized controllers for physical systems which are guaranteed to meet their design goals. The author gives a unique, logic-based perspective on hybrid systems analysis.