




# Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity

Wojciech Różowski   

Department of Computer Science, University College London, UK

Tobias Kappé   

Open Universiteit, Heerlen, The Netherlands




ILLC, University of Amsterdam, The Netherlands

Dexter Kozen   

Department of Computer Science, Cornell University, Ithaca, NY, USA

Todd Schmid   

Department of Computer Science, University College London, UK

Alexandra Silva   

Department of Computer Science, Cornell University, Ithaca, NY, USA

---

## Abstract

We introduce Probabilistic Guarded Kleene Algebra with Tests (ProbGKAT), an extension of GKAT that allows reasoning about uninterpreted imperative programs with probabilistic branching. We give its operational semantics in terms of special class of probabilistic automata. We give a sound and complete Salomaa-style axiomatisation of bisimilarity of ProbGKAT expressions. Finally, we show that bisimilarity of ProbGKAT expressions can be decided in  $O(n^3 \log n)$  time via a generic partition refinement algorithm.

**2012 ACM Subject Classification** Theory of computation → Program reasoning

**Keywords and phrases** Kleene Algebra with Tests, program equivalence, completeness, coalgebra

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2023.136

**Category** Track B: Automata, Logic, Semantics, and Theory of Programming

**Related Version** *Full Version*: <https://arxiv.org/abs/2305.01755> [51]

**Funding** This work was partially supported by ERC grant Autoprobe (no. 101002697; Różowski, Schmid and Silva), the EU's Horizon 2020 research and innovation program under Marie Skłodowska-Curie grant VERLAN (no. 101027412; Kappé), and NSF grant CCF-2008083 (Kozen).

## 1 Introduction

Randomisation is an important feature in the design of efficient algorithms, cryptographic protocols, and stochastic simulation [9]. For a simple example of randomisation, imagine simulating a three-sided die [64]. There are at least two ways to do this:

- A reference implementation could use a fair coin and a biased coin with probability  $\frac{1}{3}$  of landing on **heads**: Toss the biased coin first. If it lands on **heads**, return  $\square$ , and otherwise toss the fair coin and return  $\square$  if it lands on **heads** or  $\square$  otherwise.
- Another way to do this is with two consecutive tosses of a fair coin: if the outcome is **heads-heads**, then return  $\square$ ; if it is **heads-tails**, return  $\square$ ; if it is **tails-heads**, return  $\square$ ; and if it is **tails-tails**, repeat the process [34].

These programs can be written using a function  $\text{flip}(p)$  that returns **true** (**heads**) with probability  $p$ , and **false** (**tails**) with probability  $1 - p$ , see Figure 1. If we can prove that those programs are equivalent, then we can be certain they implement the same distribution.



© Wojciech Różowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva; licensed under Creative Commons License CC-BY 4.0

50th International Colloquium on Automata, Languages, and Programming (ICALP 2023).

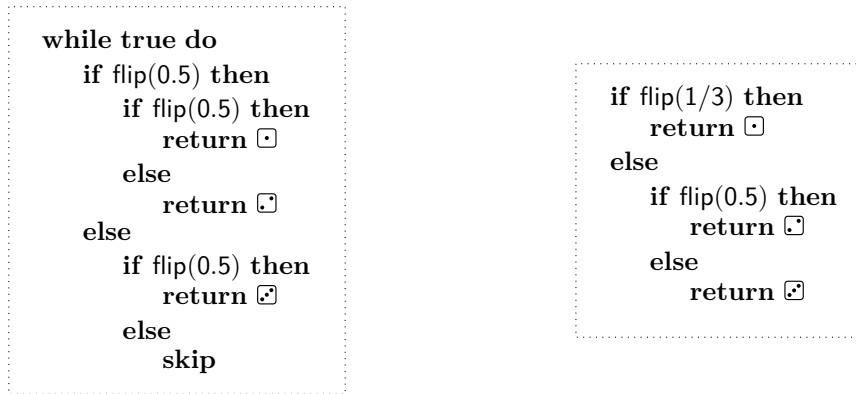
Editors: Kousha Etessami, Uriel Feige, and Gabriele Puppis; Article No. 136; pp. 136:1–136:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** On the left, the Knuth-Yao process to simulate a three-sided die with two throws of a fair coin. On the right, direct implementation of the three-sided die (note that the probability of the first else-branch is  $\frac{2}{3}$ , hence both ◻ and ◻ are returned with probability  $\frac{1}{3}$ ).

In this paper, we introduce Probabilistic GKAT (ProbGKAT), a language based on Guarded Kleene Algebra with Tests (GKAT) [39, 58, 53] augmented with extra constructs for reasoning about such randomised programs. The laws of GKAT allow reasoning about the equivalence of uninterpreted programs with deterministic control flow in the form of Boolean branching (`if-then-else`) and looping (`while-do`) constructs. GKAT comes equipped with an automata-theoretic operational semantics, a *nearly linear* decision procedure, and complete axiomatic systems for reasoning about trace equivalence [58] and bisimilarity [53] of expressions, both inspired by Salomaa’s axiomatisation of Kleene Algebra [52].

ProbGKAT extends GKAT with three new syntactic constructs: (1) a probabilistic choice operator, representing branching based on a (possibly biased) coin flip; (2) a probabilistic loop operator, representing a generalised Bernoulli process; and (3) return values, which allow a limited form of non-local control flow akin to `return` statements in imperative programming.

The main focus of this paper is the problem of axiomatising bisimilarity of ProbGKAT expressions. We build on an inference system for reasoning about bisimilarity of GKAT expressions [53], which includes a generalisation of Salomaa’s axiomatisation of the Kleene star [52] called the Uniqueness of Solutions axiom (UA), also known in the process algebra community as Recursive Specification Principle (RSP) [10]. In the presence of both Boolean guarded and probabilistic branching, axiomatisation becomes significantly more involved. Besides adding intuitive rules governing the behaviour of probabilistic choice and loops, we add axioms capturing the interaction of both kinds of branching when combined with looping constructs. Moreover, in the case of ProbGKAT, showing the soundness of UA becomes highly nontrivial. We do so by exploiting the topological structure of the operational model, namely the *behavioural pseudometric* associated with bisimilarity. Despite the jump in difficulty, our completeness proof follows a similar strategy as the one for GKAT modulo bisimilarity [53].

Our main contributions are as follows.

- We provide an operational semantics of ProbGKAT programs, which relies on a type of automata that have both Boolean guarded and probabilistic transitions (Section 3).
- We concretely characterise bisimulations for our automata using an adaption of the flow network characterisation of bisimilarity of Markov chains [30, 8] (Section 4).
- We give a sound and complete Salomaa-style axiomatisation of bisimulation equivalence of ProbGKAT expressions (Sections 5 and 6).

$e, f \in \text{Exp} ::=$	$p \in \text{Act}$	do $p$
	$b \in \text{BExp}$	assert $b$
	$e +_b f$	if $b$ then $e$ else $f$
	$e ; f$	
	$e^{(b)}$	while $b$ do $e$
	$v \in \text{Out}$	return $v$
	$e \oplus_r f$	if flip( $r$ ) then $e$ else $f$
	$e^{[r]}$	while flip( $r$ ) do $e$

■ **Figure 2** Syntax of ProbGKAT.

- We show that, when the number of tests is fixed, bisimilarity of ProbGKAT expressions can be efficiently decided in  $\mathcal{O}(n^3 \log n)$  time, where  $n$  is the total size of programs under comparison, by using *coalgebraic partition refinement* [17, 67] (Section 7).

In Section 2 we define the syntax of ProbGKAT. We survey related work in Section 8; conclusions and further work appear in Section 9. Proofs appear in the full version [51].

## 2 Syntax

ProbGKAT has a two-sorted syntax consisting of a set of *expressions*  $\text{Exp}$  that contains a set  $\text{BExp}$  of *Boolean assertions* or *tests*. For a fixed finite set  $T$  of *primitive tests*, the syntax for tests is denoted  $\text{BExp}$  and generated by the grammar

$$b, c \in \text{BExp} ::= 0 \mid 1 \mid t \in T \mid b + c \mid bc \mid \bar{b}$$

Here,  $0$  and  $1$  respectively denote false and true,  $\bar{\cdot}$  denotes negation,  $+$  is disjunction, and juxtaposition is conjunction. Let  $\equiv_{\text{BA}}$  denote Boolean equivalence in  $\text{BExp}$ . Entailment is a preorder on  $\text{BExp}$  given by  $b \leq_{\text{BA}} c \iff b + c \equiv_{\text{BA}} c$ . The quotient of  $\text{BExp}$  by  $\equiv_{\text{BA}}$  is the free Boolean algebra on the set of generators  $T$ , in which entailment  $-[b]_{\equiv_{\text{BA}}} \leq [c]_{\equiv_{\text{BA}}} \iff b \leq_{\text{BA}} c$  is a partial order, with bottom and top elements being the equivalence classes of  $0$  and  $1$  respectively. The minimal non-zero elements of that partial order are called *atoms*, and we will use  $\text{At}$  to denote the set of atoms. For fixed sets  $\text{Act}$  of *atomic actions* and  $\text{Out}$  of *return values*, the set  $\text{Exp}$  of ProbGKAT expressions is defined by the grammar in Figure 2.

The syntax of GKAT is captured by the first five cases in Figure 2, and so is a proper fragment of ProbGKAT. There are three new constructs: *return values*, *probabilistic choices*, and *probabilistic loops*. Return values behave like return statements in imperative programs, introducing a form of non-local control flow. The probabilistic choice  $e \oplus_r f$  flips a biased coin with real bias  $r \in [0, 1]$  and depending on the outcome runs  $e$  with probability  $r$  and  $f$  with probability  $1 - r$ . The probabilistic loop  $e^{[r]}$  also begins with a biased coin flip, and depending on the outcome it either executes  $e$  and starts again (probability  $r$ ) or terminates (probability  $1 - r$ ). A probabilistic loop can be regarded as a generalised Bernoulli processes.

► **Example 1.** Recall the two programs from the introduction (Figure 1): one directly implementing a 3-sided die and the other simulating a 3-sided die with a fair coin. We can express these programs using three output values,  $\square$ ,  $\square$ , and  $\square$ , to model the possible outcomes of the three-sided die. The first program is the infinite while loop  $f = g^{(1)}$ , where

the loop body is given by  $g = (\square \oplus_{\frac{1}{2}} \boxtimes) \oplus_{\frac{1}{2}} (\boxtimes \oplus_{\frac{1}{2}} \mathbf{1})$ . In  $f$ ,  $\square$  represents heads-heads,  $\boxtimes$  is heads-tails,  $\boxplus$  is tails-heads, and tails-tails prompts a rethrow. The second program encodes the ProbGKAT expression  $e = \square \oplus_{\frac{1}{3}} (\boxtimes \oplus_{\frac{1}{2}} \boxplus)$ .

### 3 Operational semantics

In this section, we formally introduce ProbGKAT automata, the operational models of ProbGKAT expressions. We associate a ProbGKAT automaton with each expression via a small-step semantics inspired by Brzozowski derivatives [13]. As we will see, the biggest hurdle is the semantics of the probabilistic loop. Before we provide our small-step semantics, we introduce the notation and operations on probability distributions that we will need.

**Preliminary definitions.** A function  $\nu : X \rightarrow [0, 1]$  is called a (*probability*) *distribution on  $X$*  if it satisfies  $\sum_{x \in X} \nu(x) = 1$ . In case  $\sum_{x \in X} \nu(x) \leq 1$  we call  $\nu$  a *subprobability distribution*, or *subdistribution*. Every (sub)distribution  $\nu$  in this paper is *finitely supported*, which means that the set  $\text{supp}(\nu) = \{x \in X \mid \nu(x) > 0\}$  is finite. Given  $A \subseteq X$ , we define  $\nu[A] = \sum_{x \in A} \nu(x)$ . This sum is well-defined because only finitely many summands have non-zero probability.

We use  $\mathcal{D}_\omega(X)$  to denote the set of finitely supported probability distributions on the set  $X$ . A function  $f : X \rightarrow Y$  can be *lifted* to a map  $\mathcal{D}_\omega(f) : \mathcal{D}_\omega(X) \rightarrow \mathcal{D}_\omega(Y)$  between distributions by setting  $\mathcal{D}_\omega(f)(\nu) = \nu[f^{-1}(y)]$ . Given  $x \in X$ , its *Dirac delta* is the distribution  $\delta_x$ ; here  $\delta_x(y)$  is equal to 1 when  $x = y$ , and 0 otherwise. Given  $f : X \rightarrow \mathcal{D}_\omega(Y)$ , there is a unique map  $\bar{f} : \mathcal{D}_\omega(X) \rightarrow \mathcal{D}_\omega(Y)$  such that  $f = \bar{f} \circ \delta$ , called the *convex extension of  $f$* , and explicitly given by  $\bar{f}(\nu)(y) = \sum_{x \in X} \nu(x)f(x)(y)$ .

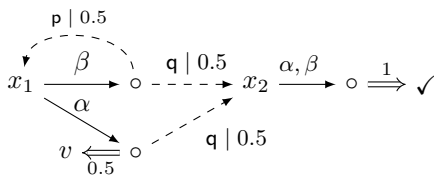
When  $\nu, \mu : X \rightarrow [0, 1]$  are probability distributions and  $r \in [0, 1]$ , we write  $r\nu + (1-r)\mu$  for the *convex combination* of  $\nu$  and  $\mu$ , which is the probability distribution given by  $(r\nu + (1-r)\mu)(x) = r\nu(x) + (1-r)\mu(x)$ ; this operation preserves finite support.

**Operational model.** Operationally, ProbGKAT expressions denote states in a transition system called a ProbGKAT *automaton*. Below, we write  $2 = \{\times, \checkmark\}$  for a two element set of symbols denoting rejection and acceptance respectively.

► **Definition 2.** A ProbGKAT automaton is a pair  $(X, \beta)$  consisting of set of states  $X$  and a transition function  $\beta : X \times \text{At} \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times X)$ .

A state in a ProbGKAT automaton associates each Boolean atom  $\alpha \in \text{At}$  (capturing the global state of the Boolean variables) with a finitely supported probability distribution over several possible outcomes. One possible outcome is *termination*, which ends execution and either signals success ( $\checkmark$ ) or failure ( $\times$ ), or returns an output value ( $v \in \text{Out}$ ). The other possible outcome is *progression*, performing an action ( $p \in \text{Act}$ ) and transitioning to a state.

► **Example 3.** Let  $X = \{x_1, x_2\}$ ,  $\text{At} = \{\alpha, \beta\}$ ,  $\text{Act} = \{p, q\}$  and  $\text{Out} = \{v\}$ . On the right, there is a definition of a transition function  $\tau : X \times \text{At} \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times X)$ , while on the left there is a visual representation of  $(X, \tau)$ . Given a state  $x \in X$  and an atom  $\alpha \in \text{At}$ , we write  $\tau(x)_\alpha$  rather than  $\tau(x)(\alpha)$ .



$$\tau(x_1)_\alpha = \frac{1}{2}\delta_{(q, x_2)} + \frac{1}{2}\delta_v$$

$$\tau(x_1)_\beta = \frac{1}{2}\delta_{(p, x_1)} + \frac{1}{2}\delta_{(q, x_2)}$$

$$\tau(x_2)_\alpha = \tau(x_2)_\beta = \delta_{\checkmark}$$

We use solid lines annotated with (sets of) atoms to denote Boolean guarded branching, dashed lines annotated with atomic actions and probabilities to denote probabilistic labelled transitions to a next state, and double bar arrows pointing at elements of  $2 + \text{Out}$  annotated with probabilities to denote probabilistic transitions that result in termination or output.

The following notions of homomorphism and bisimulation describe structure-preserving maps and relations between ProbGKAT automata.

► **Definition 4.** A homomorphism between ProbGKAT automata  $(X, \beta)$  and  $(Y, \gamma)$  is a function  $f : X \rightarrow Y$  satisfying for all  $x \in X$  and  $\alpha \in \text{At}$

1. For any  $o \in 2 + \text{Out}$ ,  $\gamma(f(x))_\alpha(o) = \beta(x)_\alpha(o)$
2. For any  $(p, y) \in \text{Act} \times Y$ ,  $\gamma(f(x))_\alpha(p, y) = \beta(x)_\alpha[\{p\} \times f^{-1}(y)]$

► **Definition 5.** Let  $(X, \beta)$  and  $(Y, \gamma)$  be ProbGKAT automata and let  $R \subseteq X \times Y$  be a relation.  $R$  is a bisimulation if there exists a transition function  $\rho : R \times \text{At} \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times R)$  such that projection maps  $\pi_1 : R \rightarrow X$  and  $\pi_2 : R \rightarrow Y$  given by  $\pi_1(x, y) = x$  and  $\pi_2(x, y) = y$  are homomorphisms from  $(R, \rho)$  to  $(X, \beta)$  and  $(Y, \gamma)$  respectively.

► **Remark 6.** Definitions 4 and 5 are direct translations from the coalgebraic theory of ProbGKAT automata [51, Appendix A]. Coalgebra plays a central role in our proofs, but for purposes of exposition it does not appear in the body of the present paper.

**Brzozowski construction.** ProbGKAT expressions can be endowed with an operational semantics in the form of a ProbGKAT automaton  $\partial : \text{Exp} \times \text{At} \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times \text{Exp})$ , which we refer to as the *Brzozowski derivative*, as it is reminiscent of the analogous construction for regular expressions and deterministic finite automata due to Brzozowski [13].

Given  $\alpha \in \text{At}$ ,  $e, f \in \text{Exp}$ ,  $\mathbf{b} \in \text{BExp}$ ,  $v \in \text{Out}$ ,  $r \in [0, 1]$ , and  $\mathbf{p} \in \text{Act}$ , we define

$$\begin{aligned} \partial(\mathbf{b})_\alpha &= \begin{cases} \delta_\checkmark & \alpha \leq_{\text{BA}} \mathbf{b} \\ \delta_X & \alpha \leq_{\text{BA}} \bar{\mathbf{b}} \end{cases} & \partial(e +_{\mathbf{b}} f)_\alpha &= \begin{cases} \partial(e)_\alpha & \alpha \leq_{\text{BA}} \mathbf{b} \\ \partial(f)_\alpha & \alpha \leq_{\text{BA}} \bar{\mathbf{b}} \end{cases} \\ \partial(v)_\alpha &= \delta_v & \partial(\mathbf{p})_\alpha &= \delta_{(\mathbf{p}, 1)} & \partial(e \oplus_r f)_\alpha &= r\partial(e)_\alpha + (1 - r)\partial(f)_\alpha \end{aligned}$$

The derivatives of sequential composition and loops are defined below. The outgoing transitions of  $\mathbf{b} \in \text{BExp}$  depend on whether or not the input atom  $\alpha \in \text{At}$  satisfies  $\mathbf{b}$ , either outputting  $\checkmark$  (success) or  $X$  (abort) with probability 1. The outgoing transitions of a guarded choice  $e +_{\mathbf{b}} f$  consist of the outgoing transitions of  $e$  labelled by atoms satisfying  $\mathbf{b}$  and the outgoing transitions of  $f$  labelled by atoms satisfying  $\bar{\mathbf{b}}$  (as in GKAT). The output value  $v \in \text{Out}$  returns the value  $v$  with probability 1 given any input atom. The atomic action  $\mathbf{p} \in \text{Act}$  emits  $\mathbf{p}$  given any input atom and transitions to the expression  $\mathbf{1}$ . The outgoing transitions of the probabilistic choice  $e \oplus_r f$  consist of the outgoing transitions of  $e$  with probabilities scaled by  $r$  and the outgoing transitions of  $f$  scaled by  $1 - r$ .

The behaviour of the sequential composition  $e ; f$  is more complicated. We need to factor in the possibility that  $e$  may accept with some probability  $t$  given an input atom  $\alpha$ , in which case the  $\alpha$ -labelled outgoing transitions of  $f$  contribute to the outgoing transitions of  $e ; f$ . Formally, we write  $\partial(e ; f)_\alpha = \partial(e)_\alpha \triangleleft_\alpha f$ , where given  $\alpha \in \text{At}$  and  $f \in \text{Exp}$  we define  $(-\triangleleft_\alpha f) : \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times \text{Exp}) \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times \text{Exp})$  to be the convex extension of  $c_{\alpha, f} : 2 + \text{Out} + \text{Act} \times \text{Exp} \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times \text{Exp})$  given below on the left.

$$c_{\alpha, f}(x) = \begin{cases} \delta_x & x \in \{X\} \cup \text{Out} \\ \partial(f)_\alpha & x = \checkmark \\ \delta_{(p, e'; f)} & x = (p, e') \end{cases} \quad \partial(f)_\alpha \times \leftarrow \begin{array}{c} e; f \\ \alpha \downarrow \\ \circ \end{array} \begin{array}{l} \xrightarrow{t} \\ \xrightarrow{p | s} \end{array} \begin{array}{c} \circ \\ \xrightarrow{s} \end{array} e'; f$$

Intuitively,  $c_{\alpha, f}$  reroutes the transitions coming out of  $e$ : acceptance (the second case) is replaced by the behaviour of  $f$ , and the probability mass of transitioning to  $e'$  (the third case) is reassigned to  $e; f$ . The branches that output the elements of  $\{X\} + \text{Out}$  are unchanged by this operation. A pictorial representation of the effect on the derivatives of  $e; f$  is given above on the right. Here, we assume that  $\partial(e)_\alpha$  can perform a  $p$ -transition to  $e'$  with probability  $s$ ; we make the same assumption in the informal descriptions of derivatives for loops, below.

For guarded loops, we consider three cases when defining  $\partial(e^{(b)})_\alpha$ . If  $\alpha \leq_{\text{BA}} \bar{b}$ , then the current state does not satisfy the loop guard and can be skipped:  $\partial(e^{(b)})_\alpha = \delta_\checkmark$ . If  $\alpha \leq_{\text{BA}} b$  and  $\partial(e)_\alpha(\checkmark) = 1$ , then the loop body is called, but the inner program  $e$  does not perform actions. We identify divergent loops with rejection and so in this case we set  $\partial(e^{(b)})_\alpha = \delta_X$ . If  $\alpha \leq_{\text{BA}} b$  and  $\partial(e)_\alpha(\checkmark) < 1$ , the program executes the loop body and starts again, having to redistribute the probability mass of immediate acceptance  $\partial(e)_\alpha(\checkmark)$  through each execution. So, for  $\alpha \leq_{\text{BA}} b$  and  $\partial(e)_\alpha(\checkmark) < 1$ , the definition of  $\partial(e^{(b)})_\alpha$  is given below on the left: it rejects or returns when  $e$  does, and transitions to  $e'; e^{(b)}$  when  $e$  transitions to  $e'$ .

$$\partial(e^{(b)})_\alpha(x) = \begin{cases} \frac{\partial(e)_\alpha(x)}{1 - \partial(e)_\alpha(\checkmark)} & x \in \{X\} \cup \text{Out} \\ \frac{\partial(e)_\alpha(p, e')}{1 - \partial(e)_\alpha(\checkmark)} & x = (p, (e'; e^{(b)})) \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{c} e^{(b)} \xrightarrow{\bar{b}} \circ \xrightarrow{1} \checkmark \\ \alpha \downarrow \\ \circ \end{array} \begin{array}{l} \xrightarrow{t} \\ \xrightarrow{p | s/(1-t)} \end{array} \begin{array}{c} \circ \\ \xrightarrow{s} \end{array} e'; e^{(b)}$$

The reweighing of probabilities used in the definition of the loops comes from defining loops as least fixpoints w.r.t. to an order on distributions, similarly to Stark and Smolka [62].

Finally, we specify the behaviour of the probabilistic loop. In the special case where  $\partial(e)_\alpha(\checkmark) = 1$  and  $r = 1$ , the loop will not terminate; hence we set  $\partial(e^{[r]})_\alpha = \delta_X$ . In all other cases, we look at  $\partial(e)_\alpha$  to build  $\partial(e^{[r]})_\alpha$  for each  $\alpha \in \text{At}$ . First, we make sure that the loop may be skipped with probability  $1 - r$ . Next, we account for the possibility that  $e$  may reject or return a value, and we modify the productive branches by adding  $e^{[r]}$  to be executed next, as was done for the guarded loop. The remaining mass is  $r\partial(e)_\alpha(\checkmark)$ , the probability that we will enter the loop with an atom that can skip over the loop body. As was the case for the guarded loop, we discard this possibility and redistribute it among the remaining branches. The resulting definition of  $\partial(e^{[r]})_\alpha$  is given below on the left.

$$\partial(e^{[r]})_\alpha(x) = \begin{cases} \frac{1-r}{1-r\partial(e)_\alpha(\checkmark)} & x = \checkmark \\ \frac{r\partial(e)_\alpha(x)}{1-r\partial(e)_\alpha(\checkmark)} & x \in \{X\} \cup \text{Out} \\ \frac{r\partial(e)_\alpha(p, e')}{1-r\partial(e)_\alpha(\checkmark)} & x = (p, (e'; e^{[r]})) \\ 0 & \text{otherwise} \end{cases} \quad \begin{array}{c} e^{[r]} \\ \alpha \downarrow \\ \circ \end{array} \begin{array}{l} \xrightarrow{\frac{1-r}{1-rt}} \\ \xrightarrow{p | rs/(1-rt)} \end{array} \begin{array}{c} \circ \\ \xrightarrow{s} \end{array} e'; e^{[r]} \\ \begin{array}{c} \downarrow \\ \circ \\ \downarrow \\ \times \end{array}$$

As before, we provide an informal visual depiction of the probabilistic loop semantics above on the right, using the same conventions.

**Reachable states.** For any ProbGKAT automaton  $(X, \beta)$  and any  $x \in X$ , we denote by  $\langle x \rangle_\beta$  the set of states reachable from  $x$  via  $\beta$ . Clearly,  $(\langle x \rangle_\beta, \beta)$  is a ProbGKAT automaton and is the smallest subautomaton of  $(X, \beta)$  containing  $x$ . The canonical inclusion map

$(\langle x \rangle_\beta, \beta) \rightarrow (X, \beta)$  is a ProbGKAT automaton homomorphism. In particular,  $(\langle e \rangle_\partial, \partial)$  is the smallest subautomaton of  $(\text{Exp}, \partial)$  containing  $e$ . We will refer to this subautomaton as the small-step semantics of  $e$ . We will often abuse notation and write  $\langle e \rangle_\partial$  for  $(\langle e \rangle_\partial, \partial)$ .

The following lemma says that every ProbGKAT expression generates a finite automaton.

► **Lemma 7.** *For all  $e \in \text{Exp}$ ,  $\langle e \rangle_\partial$  is finite. In fact, the number of states is bounded above by  $\#(e) : \text{Exp} \rightarrow \mathbb{N}$ , where  $\#(-)$  is defined recursively by*

$$\begin{aligned} \#(b) &= 1 & \#(v) &= 1 & \#(p) &= 2 & \#(e +_b f) &= \#(e) + \#(f) & \#(e \oplus_r f) &= \#(e) + \#(f) \\ \#(e ; f) &= \#(e) + \#(f) & \#(e^{(b)}) &= \#(e) & \#(e^{[r]}) &= \#(e) \end{aligned}$$

## 4 Bisimulations and their properties

Verifying that a given relation is a bisimulation (Definition 5) requires that we construct a suitable transition structure on the relation. In this section, we give necessary and sufficient conditions for the existence of such a transition structure. We also study properties of the bisimilarity relation  $\sim$ , the largest bisimulation [50].

**Concrete characterisation of bisimulation equivalence.** There is a beautiful characterisation of bisimulations between Markov chains in [30], whose proof makes use of the max-flow min-cut theorem. Adapting this work to ProbGKAT automata produces a useful characterisation of *bisimulation equivalences*, bisimulations that are also equivalence relations.

► **Lemma 8.** *Let  $(X, \beta)$  be a ProbGKAT automaton and let  $R \subseteq X \times X$  be an equivalence relation.  $R$  is a bisimulation if and only if for all  $(x, y) \in R$  and  $\alpha \in \text{At}$ ,*

1. *for all  $o \in 2 + \text{Out}$ ,  $\beta(x)_\alpha(o) = \beta(y)_\alpha(o)$ , and*
2. *for all equivalence classes  $Q \in X/R$  and all  $p \in \text{Act}$ ,  $\beta(x)_\alpha[\{p\} \times Q] = \beta(y)_\alpha[\{p\} \times Q]$*

This lemma can be seen as an extension of Larsen-Skou bisimilarity [40] to systems with outputs. Intuitively,  $R$  is a bisimulation equivalence if for any atom  $\alpha \in \text{At}$  and  $(x, y) \in R$ , the transitions assign the same probabilities to any output, and the probability of transitioning into any given equivalence class after emitting  $p$  is the same for both  $x$  and  $y$ .

**Bisimilarity and its properties.** Given a relation  $R \subseteq X \times Y$ , define  $R^{-1} = \{(y, x) \mid x R y\}$ , and given  $A \subseteq X$ , write  $R(A) = \{y \in Y \mid x R y, x \in A\}$ . The bisimilarity relation  $\sim_{\beta, \gamma} \subseteq X \times Y$  between  $(X, \beta)$  and  $(Y, \gamma)$  is the *greatest fixpoint* of the following operator.

► **Definition 9.** *Let  $(X, \beta)$  and  $(Y, \gamma)$  be ProbGKAT automata and let  $R \subseteq X \times Y$ . We define the operator  $\Phi_{\beta, \gamma} : 2^{X \times Y} \rightarrow 2^{X \times Y}$  so that  $(x, y) \in \Phi_{\beta, \gamma}(R)$  if for any given  $\alpha \in \text{At}$ ,*

- *for all  $o \in 2 + \text{Out}$ ,  $\beta(x)_\alpha(o) = \gamma(y)_\alpha(o)$ ,*
- *for all  $A \subseteq X$  and all  $p \in \text{Act}$ ,  $\beta(x)_\alpha[\{p\} \times A] \leq \gamma(y)_\alpha[\{p\} \times R(A)]$ , and*
- *for all  $B \subseteq Y$  and  $p \in \text{Act}$ ,  $\gamma(y)_\alpha[\{p\} \times B] \leq \beta(x)_\alpha[\{p\} \times R^{-1}(B)]$ .*

*From now on, we will omit the subscripts from  $\Phi$  when the automata are clear from context.*

The operator  $\Phi_{\beta, \gamma}$  can also be used to define a behavioural pseudometric. Let  $(X, \beta)$  be a ProbGKAT automaton. A *relation refinement chain* is an indexed family  $\{\sim^{(i)}\}_{i \in \mathbb{N}}$  of relations on  $X$  defined as:  $\sim^{(0)} = X \times X$ ,  $\sim^{(i+1)} = \Phi(\sim^{(i)})$ . We can intuitively think of successive elements of this chain as closer approximations of bisimilarity (see also [26]).

► **Theorem 10.** *Let  $(X, \beta)$  be a ProbGKAT automaton. For any  $x, y \in X$ ,  $x \sim y$  if and only if for all  $i \in \mathbb{N}$ , we have  $x \sim^{(i)} y$ .*



■ **Table 1** Axiomatisation of ProbGKAT. In the figure  $e, f, g \in \text{Exp}$ ,  $b, c \in \text{BExp}$ ,  $v \in \text{Out}$ ,  $p \in \text{Act}$  and  $r, s \in [0, 1]$ . Laws involving division of probabilities apply when the denominator is not zero. To simplify the notation, we write  $E(e) = 0$  to denote that for all  $\alpha \in \text{At}$  it holds that  $E(e)_\alpha = 0$ .

Guarded Choice Axioms	Probabilistic Choice Axioms
(G1) $e +_b e \equiv e$ (G2) $e +_b f \equiv b ; e +_b f$ (G3) $e +_b f \equiv f +_{\bar{b}} e$ (G4) $(e +_b f) +_c g \equiv e +_{bc} (f +_c g)$	(P1) $e \oplus_r e \equiv e$ (P2) $e \oplus_1 f \equiv e$ (P3) $e \oplus_r f \equiv f \oplus_{1-r} e$ (P4) $(e \oplus_r f) \oplus_s g \equiv e \oplus_{rs} (f \oplus_{\frac{(1-r)s}{1-rs}} g)$
Distributivity Axiom	Fixpoint Rules
(D) $e \oplus_r (f +_b g) \equiv (e \oplus_r f) +_b (e \oplus_r g)$	(F1) $\frac{g \equiv e ; g +_b f \quad E(e) = 0}{g \equiv e^{(b)} ; f}$
Sequencing Axioms	(F2) $\frac{g \equiv e ; g \oplus_r f \quad E(e) = 0}{g \equiv e^{[r]} ; f}$
(S1) $1 ; e \equiv e$ (S2) $e ; 1 \equiv e$ (S3) $(e ; f) ; g \equiv e ; (f ; g)$ (S4) $0 ; e \equiv 0$ (S5) $(e +_b f) ; g \equiv e ; g +_b f ; g$ (S6) $(e \oplus_r f) ; g \equiv e ; g \oplus_r f ; g$ (S7) $v ; e \equiv v$ (S8) $b ; c \equiv bc$	Define $E : \text{Exp} \rightarrow \text{At} \rightarrow [0, 1]$ inductively by $E(p)_\alpha = E(v)_\alpha = 0$ $E(b)_\alpha = \begin{cases} 1 & \alpha \leq_{\text{BA}} b \\ 0 & \alpha \leq_{\text{BA}} \bar{b} \end{cases}$ $E(e +_b f)_\alpha = \begin{cases} E(e)_\alpha & \alpha \leq_{\text{BA}} b \\ E(f)_\alpha & \alpha \leq_{\text{BA}} \bar{b} \end{cases}$ $E(e \oplus_r f)_\alpha = rE(e)_\alpha + (1-r)E(f)_\alpha$ $E(e ; f)_\alpha = E(e)_\alpha E(f)_\alpha$ $E(e^{(b)})_\alpha = E(\bar{b})_\alpha$ $E(e^{[r]})_\alpha = \begin{cases} 0 & r = 1 \text{ and } E(e)_\alpha = 1 \\ \frac{1-r}{1-rE(e)_\alpha} & \text{otherwise} \end{cases}$
Loop Axioms	
(L1) $e^{(b)} \equiv e ; e^{(b)} +_b 1$ (L2) $e^{[r]} \equiv e ; e^{[r]} \oplus_r 1$ (L3) $(e +_c 1)^{(b)} \equiv (c ; e)^{(b)}$ (L4) $e^{(1)} \equiv e^{[1]}$ (L5) $\frac{e \equiv (f \oplus_s 1) +_c g \quad s > 0}{c ; e^{(b)} \equiv c ; (f ; e^{(b)} +_b 1)}$	
(L6) $\frac{e \equiv (f \oplus_s 1) +_c g}{c ; e^{[r]} \equiv c ; \left( f ; e^{[r]} \oplus_{\frac{rs}{1-r(1-s)}} 1 \right)}$	

Thus, if  $x, y \in X$  are not bisimilar, then there exists a maximal  $i \in \mathbb{N}$  such that  $x \sim^{(i)} y$ . In Section 6, we use this to define a pseudometric on the states of any ProbGKAT automaton. Informally speaking, this allows us to quantify *how close* to being bisimilar two states are.

Our main goal is to axiomatise bisimilarity of ProbGKAT expressions with a set of equational laws and reason about equivalence using *equational logic*. For such an axiomatisation to exist, bisimilarity needs to be both an equivalence relation and a congruence with respect to the ProbGKAT operations. The greatest bisimulation on any ProbGKAT automaton is an equivalence [50], but being congruence requires an inductive argument.

► **Theorem 11.** *The greatest bisimulation on  $(\text{Exp}, \partial)$  is a congruence with respect to ProbGKAT operations.*



## 5 Axiomatisation

We turn our attention to *axiomatisation* of bisimilarity of ProbGKAT expressions, using an axiom system based on GKAT modulo bisimilarity [53]. First, we give an overview of the axioms, and establish their soundness. Finally, we show that our axioms are strong enough to decompose every expression into a certain syntactic normal form relating the expressions to their small-step semantics. Completeness is tackled in the next section.

**Overview of the axioms.** Table 1 contains the axioms, which are either *equational* (of the form  $e \equiv f$ ), or *quasi-equational* (of the form  $e_1 \equiv f_1, \dots, e_n \equiv f_n \implies e \equiv f$ ). It also holds the definition of the function  $E(-)$ , which is necessary to give a side condition to the fixpoint rules. We define  $\equiv \subseteq \text{Exp} \times \text{Exp}$  as the smallest congruence relation satisfying the axioms.

Axioms G1–G4 are inherited from GKAT and govern the behaviour of Boolean guarded choice. P1–P4 can be thought of as their analogues, but for the probabilistic choice. The distributivity axiom D states that guarded choice distributes over a probabilistic choice, which reflects the way our operational model resolves both types of branching.

The sequencing axioms S1–S8 are mostly inherited from GKAT. The new axioms include S6 which talks about right distributivity of sequencing over probabilistic choice and S7 which captures the intuitive property that any code executed after a `return` statement is not executed. L1 and L3 come from GKAT, while L2 is a probabilistic loop analogue of L1, which captures the semantics of the probabilistic loop in terms of recursive unrolling. L4 equates the `while(true)` and `while(flip(1))` loops. F1 and F2 are inspired by Salomaa’s axioms [52] and provide a partial converse to L1 and L2 respectively, given the loop body cannot immediately terminate. The property that a loop body has a zero probability of outputting  $\checkmark$  is formally written using the side condition  $E(e) = 0$ , which can be thought of as *empty word property* from Salomaa’s axiomatisation [52].

This leaves us with L5 and L6, which describe the behaviour of guarded and probabilistic loops where parts of the loop body may be skipped. These are quasi-equational, but can be replaced by equivalent equations – see [51, Remark 49]. L5 concerns a loop on an expression  $e$  that has probability  $1 - s$  of not performing any action, given that  $c$  holds. The rule says that, if we start the loop on  $e$  given that  $c$  holds, then either  $b$  holds and we execute  $f$ , or it does not, and the loop is skipped. The reason that we can disregard the `1` part of  $e$  is that if this branch is taken, then  $c$  still holds on the next iteration of the loop, and so the program will have to choose probabilistically between  $f$  and `1` once more. Since  $s > 0$ , it will eventually choose the probabilistic branch  $f$  with almost sure probability.

The second rule, L6, is the analogue of L5 for probabilistic choice. In this case, however, a choice for `1` also means another probabilistic experiment to determine whether the loop needs to be executed once more, with probability  $r$ . The consequence is that if the loop on  $e$  is started given that  $c$  holds, some more probability mass will shift towards skipping, as a result executing `1` some number of times before halting the loop.

**Soundness with respect to bisimilarity.** Using the characterisation from Section 4, we can show that  $\equiv$  is a bisimulation equivalence on  $(\text{Exp}, \partial)$ . The proof is available in the full version of the paper [51, Appendix D].

► **Lemma 12.**  $\equiv$  is a bisimulation equivalence on  $(\text{Exp}, \partial)$

We immediately obtain that provable equivalence is contained in bisimilarity.

► **Theorem 13 (Soundness).** For all  $e, f \in \text{Exp}$ , if  $e \equiv f$  then  $e \sim f$

## 136:10 Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity

**Example of equational reasoning.** Since our axioms are sound, we can reason about ProbGKAT expressions equationally, without constructing bisimulations by hand. Once again, we revisit the algorithm from Figure 1. To show correctness, we need to prove the equivalence of expressions  $e$  and  $g^{(1)}$  from Example 1, as follows:

$$\begin{aligned}
g^{(1)} &\equiv \left( \left( \square \oplus_{\frac{1}{2}} \square \right) \oplus_{\frac{1}{2}} \left( \square \oplus_{\frac{1}{2}} \mathbf{1} \right) \right)^{(1)} && \text{(Def. of } g) \\
&\equiv \left( \left( \left( \left( \square \oplus_{\frac{1}{2}} \square \right) \oplus_{\frac{2}{3}} \square \right) \oplus_{\frac{1}{2}} \mathbf{1} \right) \right)^{(1)} && \text{(See below)} \\
&\equiv \left( \left( \left( \left( \left( \square \oplus_{\frac{1}{2}} \square \right) \oplus_{\frac{2}{3}} \square \right) \oplus_{\frac{1}{2}} \mathbf{1} \right) +_1 \mathbf{0} \right) \right)^{(1)} && \text{(See below)} \\
&\equiv \left( \left( \square \oplus_{\frac{1}{2}} \square \right) \oplus_{\frac{2}{3}} \square \right)^{(1)} && \text{(Axioms L5 and S1)} \\
&\equiv \left( \square \oplus_{\frac{1}{3}} \left( \square \oplus_{\frac{1}{2}} \square \right) \right)^{(1)} && \text{(Axiom P4)} \\
&\equiv \left( \square \oplus_{\frac{1}{3}} \left( \square \oplus_{\frac{1}{2}} \square \right) \right); \left( \square \oplus_{\frac{1}{3}} \left( \square \oplus_{\frac{1}{2}} \square \right) \right)^{(1)} +_1 \mathbf{1} && \text{(Axiom L1)} \\
&\equiv \left( \square \oplus_{\frac{1}{3}} \left( \square \oplus_{\frac{1}{2}} \square \right) \right); e^{(1)} +_1 \mathbf{1} && \text{(Def. } e) \\
&\equiv \left( \square \oplus_{\frac{1}{3}} \left( \square \oplus_{\frac{1}{2}} \square \right) \right); e^{(1)} && \text{(See below)} \\
&\equiv \left( \square; e^{(1)} \oplus_{\frac{1}{3}} \left( \square; e^{(1)} \oplus_{\frac{1}{2}} \square; e^{(1)} \right) \right) && \text{(Axiom S6)} \\
&\equiv \left( \square \oplus_{\frac{1}{3}} \left( \square \oplus_{\frac{1}{2}} \square \right) \right) && \text{(Axiom S7)} \\
&\equiv e && \text{(Def. } e)
\end{aligned}$$

In the second step, we used that for all  $e_1, e_2, e_3 \in \mathbf{Exp}$  and  $r, s \in [0, 1]$  with  $(1-r)(1-s) > 0$ , we have  $e_1 \oplus_r (e_2 \oplus_s e_3) \equiv (e_1 \oplus_k e_2) \oplus_l e_3$ , where  $k = \frac{r}{1-(1-r)(1-s)}$  and  $l = 1 - (1-r)(1-s)$ . In the third and eighth steps, we used that for all  $e, f \in \mathbf{Exp}$ , we have that  $e +_1 f \equiv e$ . Both those equivalences follow from the other axioms; see [51, Lemma 50] in the full version of the paper for details.

**Fundamental theorem.** Every expression in the language of KA (resp. KAT, GKAT) can be reconstructed from its small-step semantics, up to  $\equiv$ . This property, often referred to as the *fundamental theorem* of (in analogy with the fundamental theorem of calculus and following the terminology of Rutten [50]) is useful in many contexts, and we will need it later on.

► **Theorem 14 (Fundamental Theorem).** *For every  $e \in \mathbf{Exp}$  it holds that*

$$e \equiv \bigoplus_{\alpha \in \mathbf{Act}} \left( \bigoplus_{d \in \text{supp}(\partial(e)_\alpha)} \partial(e)_\alpha(d) \cdot \mathbf{exp}(d) \right)$$

where  $\mathbf{exp}$  defines a function  $2 + \mathbf{Out} + \mathbf{Act} \times \mathbf{Exp} \rightarrow \mathbf{Exp}$  given by

$$\mathbf{exp}(x) = \mathbf{0} \quad \mathbf{exp}(\checkmark) = \mathbf{1} \quad \mathbf{exp}(v) = v \quad \mathbf{exp}(a, f) = a;f \quad (v \in \mathbf{Out}, a \in \mathbf{Act} \text{ and } f \in \mathbf{Exp})$$

The proof is given in the appendix. We use a generalised type of guarded and probabilistic choice ranging over indexed collections of expressions, which is defined in [51, Appendix D.2].

## 6 Completeness

Given the axioms presented in the previous section, a natural question is to ask whether they are *complete* w.r.t. bisimilarity – i.e., whether any bisimilar pair can be proved equivalent using the axioms that make up  $\equiv$ . The traditional strategy is to develop the idea of *systems of equations* within the calculus, and show that these systems have unique (least) solutions up to provable equivalence. If we can characterise the expressions of interest as solutions to a common system of equations (typically derived from the bisimulation that relates them), then uniqueness of solutions guarantees their equivalence. Unfortunately, the first step of this process, where systems of equations are shown to have unique solutions, does not transfer to ProbGKAT (nor GKAT). Indeed, some systems of equations do not have *any* solution [39, 53]; the lack of a procedure to construct solutions also encumbers a proof of uniqueness.

Instead, we follow the approach from [58] pioneered by Bergstra and Klop [10], and incorporate uniqueness of solutions into the axiomatisation. The *uniqueness axiom* (UA) that accomplishes this is an *axiom scheme*, which is to say it is a template for infinitely many axioms, one for each number of unknowns. In the case of a single unknown, one can show that F1 and F2 are special cases of UA, which moreover give a candidate solution.

With UA in hand, the traditional roadmap towards completeness works out. Before we get there, however, we must expend some energy to properly state this axiom scheme. Moreover, showing soundness of UA requires effort. Both of these take up the bulk of the development in this section; we derive the desired completeness property at the end.

**(Salomaa) systems of equations.** First, we define formally the idea of *systems of equations* for ProbGKAT automata. The constraints on each variable will be built using the following two-sorted grammar, where  $X$  is a finite set of indeterminates.

$$\begin{aligned} e_1, e_2 \in \text{Exp}(X) &::= p \mid e_1 +_{\mathbf{b}} e_2 && (p \in \text{PExp}(X), \mathbf{b} \in \text{BExp}) \\ p_1, p_2 \in \text{PExp}(X) &::= \mathbf{f} \mid \mathbf{g}x \mid p_1 \oplus_r p_2 && (\mathbf{f}, \mathbf{g} \in \text{Exp}, x \in X, r \in [0, 1]) \end{aligned}$$

► **Definition 15.** A system of equations is a pair  $(X, \tau : X \rightarrow \text{Exp}(X))$  consisting of a finite set  $X$  of indeterminates and a function  $\tau : X \rightarrow \text{Exp}(X)$ . If for all  $x \in X$ , in each of  $\tau(x)$  all subterms of the form  $\mathbf{g}x$  satisfy  $\mathbf{E}(\mathbf{g}) = 0$ , then such system is called Salomaa.<sup>1</sup>

Every finite state ProbGKAT automaton yields a Salomaa system of equations.

► **Definition 16.** Let  $(X, \beta)$  be a finite state ProbGKAT automaton. A system of equations associated with  $(X, \beta)$  is a Salomaa system  $(X, \tau)$ , with  $\tau : X \rightarrow \text{Exp}(X)$  defined by

$$\tau(x) = \bigoplus_{\alpha \in \text{Act}} \left( \bigoplus_{d \in \text{supp}(\beta(x))_{\alpha}} \beta(x)_{\alpha}(d) \cdot \text{sys}(d) \right)$$

where  $\text{sys} : 2 + \text{Out} + \text{Act} \times \text{Exp} \rightarrow \text{PExp}(X)$  is given by

$$\text{sys}(x) = 0 \quad \text{sys}(\checkmark) = 1 \quad \text{sys}(v) = v \quad \text{sys}(a, x) = ax$$

<sup>1</sup> In process algebra [46], Salomaa systems are usually called *guarded*. We avoid the latter name to prevent confusion with Boolean guarded choice present in (Prob)GKAT.

## 136:12 Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity

► **Example 17.** In the system associated with the automaton from Example 3,  $\tau$  is given by

$$x_1 \mapsto (\mathbf{q}x_2 \oplus_{\frac{1}{2}} v) +_{\alpha} \left( (\mathbf{p}x_1 \oplus_{\frac{1}{2}} \mathbf{q}x_2) +_{\beta} \mathbf{0} \right) \quad x_2 \mapsto \mathbf{1} +_{\alpha} (\mathbf{1} +_{\beta} \mathbf{0})$$

Given a function  $h : X \rightarrow \mathbf{Exp}$  that assigns a value to each indeterminate in  $X$ , we can derive a ProbGKAT expression  $h^{\#}(e)$  for each  $e \in \mathbf{Exp}(X)$  inductively, as follows:  $h^{\#}(f) = f$ ,  $h^{\#}(p_1 \oplus_r p_2) = h^{\#}(p_1) \oplus_r h^{\#}(p_2)$ ,  $h^{\#}(gx) = g; h(x)$ ,  $h^{\#}(e_1 +_{\mathbf{b}} e_2) = h^{\#}(e_1) +_{\mathbf{b}} h^{\#}(e_2)$ . We can now state the notion of a *solution* to the Salomaa system. Rather than expecting both sides of equations to be strictly equal, we require them to be related by a relation, which we leave as a parameter to instantiate later.

► **Definition 18.** Let  $R \subseteq \mathbf{Exp} \times \mathbf{Exp}$ . A solution up to  $R$  to a system  $(X, \tau)$  is a map  $h : X \rightarrow \mathbf{Exp}$  satisfying for all  $x \in X$  that  $(h(x), h^{\#}(\tau(x))) \in R$ .

► **Example 19.** A solution up to  $\equiv$  to the system from Example 17 would satisfy

$$h(x_1) \equiv (\mathbf{q}; h(x_2) \oplus_{\frac{1}{2}} v) +_{\alpha} \left( (\mathbf{p}; h(x_1) \oplus_{\frac{1}{2}} \mathbf{q}; h(x_2)) +_{\beta} \mathbf{0} \right) \quad h(x_2) \equiv \mathbf{1} +_{\alpha} (\mathbf{1} +_{\beta} \mathbf{0})$$

In this case, choosing  $h(x_1) = (\mathbf{p} +_{\beta} v)^{[\frac{1}{2}]}; \mathbf{q}$  and  $h(x_2) = \mathbf{1}$  fits these constraints.

► **Example 20.** Let  $r \in [0, 1]$ . The recursive specification on the left below describes a program `randAdd(m, r)` which takes an integer  $m$  and bias  $r$ . As long as  $m$  is strictly below 10, this program flips an  $r$ -biased coin to decide between incrementing  $m$  followed by a recursive call or termination. That recursive specification can be thought of as a Salomaa system with one unknown; the program on the right is a solution up to  $\equiv$ .

```
def randAdd(m, r):
  if m < 10 then
    if flip(r) then
      m++;
      randAdd(m, r)
    else
      skip
  else
    skip
```

```
while flip(r) do
  if m < 10 then
    m++;
    randAdd(m, r)
  else
    skip
```

Solutions up to  $\equiv$  can be characterised concretely, using Theorem 14.

► **Theorem 21.** Let  $(X, \beta)$  be a finite state ProbGKAT automaton. The map  $h : X \rightarrow \mathbf{Exp}$  is a solution up to  $\equiv$  to the system associated with  $(X, \beta)$  if and only if  $[-]_{\equiv} \circ h$  is a ProbGKAT automata homomorphism from  $(X, \beta)$  to  $(\mathbf{Exp}/\equiv, \bar{\delta})$ . We write  $\bar{\delta}$  to denote the unique transition function on  $\mathbf{Exp}/\equiv$  which makes the quotient map  $[-]_{\equiv} : \mathbf{Exp} \rightarrow \mathbf{Exp}/\equiv$  a ProbGKAT automaton homomorphism from  $(\mathbf{Exp}, \delta)$  [50, Proposition 5.8].

**Uniqueness of Solutions axiom.** Informally, UA extends  $\equiv$  by stating that solutions to Salomaa systems, if they exist, are unique. Formally, we define  $\doteq \subseteq \mathbf{Exp} \times \mathbf{Exp}$  to be the least congruence that contains  $\equiv$ , and satisfies the following (quasi-equational) axiom:

$$\frac{(X, \tau) \text{ is a Salomaa system} \quad f, g : X \rightarrow \mathbf{Exp} \text{ are solutions of } (X, \tau) \text{ up to } \doteq}{f(x) \doteq g(x) \text{ for all } x \in X} \quad (\text{UA})$$

F1 and F2 are instantiations of UA for Salomaa systems with one variable.

**Behavioural pseudometric.** We now develop the theory necessary to verify soundness of UA. First, note that for every ProbGKAT, we can define a function  $d_X : X \times X \rightarrow \mathbb{R}^+$ :

$$d_X(x, y) = \begin{cases} 2^{-n} & n \text{ is maximal such that } x \sim^{(n)} y \\ 0 & \text{if } x \sim y \end{cases}$$

The above is well-defined by Theorem 10, and is a *pseudometric*, in the following sense.

► **Definition 22.** A pseudometric space is a pair  $(X, d_X)$ , where  $d_X : X \times X \rightarrow \mathbb{R}^+$  is a pseudometric, which means that for all  $x, y, z \in X$  we have

$$d_X(x, x) = 0 \quad d_X(x, y) = d_X(y, x) \quad d_X(x, z) \leq d_X(x, y) + d_X(y, z)$$

Let  $k \in \mathbb{R}^+$ . A mapping  $f : X \rightarrow Y$  between pseudometric spaces  $(X, d_X)$  and  $(Y, d_Y)$  is called *k-Lipschitz* if for all  $x, y \in X$   $d_Y(f(x), f(y)) \leq kd_X(x, y)$ .

The behavioural pseudometric satisfies the definition above, in a strong sense.

► **Lemma 23.** For every ProbGKAT automaton,  $(X, \beta)$ ,  $(X, d_X)$  a pseudometric space that is ultra, in the sense that for all  $x, y, z \in X$  we have  $d_X(x, z) = \max\{d_X(x, y), d_X(y, z)\}$ .

Let  $(X, d_X)$  and  $(Y, d_Y)$  be pseudometric spaces. Their *product* is a pseudometric space  $(X \times Y, d_{X \times Y})$  where  $d_{X \times Y}$  is defined by  $d_{X \times Y}((x, y), (x', y')) = \max\{d_X(x, x'), d_Y(y, y')\}$ . It is easy to show that if both pseudometric spaces are ultra, then so is their product. Going forward, we will omit subscripts when they are clear from context.

**Soundness of the Uniqueness Axiom.** Every Salomaa system  $(X, \tau : X \rightarrow \text{Exp}(X))$  with  $X = \{x_1, x_2, \dots, x_n\}$  induces a mapping  $\bar{\tau} : \text{Exp}^n \rightarrow \text{Exp}^n$ . Intuitively, this mapping takes a vector  $\vec{e} = (e_1, \dots, e_n)$ , and produces a new vector where the  $i$ -th component is the evaluation of  $\tau(x_i)$  when each  $x_j$  is substituted by  $e_j$ . More formally, given this  $\vec{e}$ , we define  $e : X \rightarrow \text{Exp}$  by  $e(x_i) = e_i$ , and set  $\bar{\tau}(\vec{e}) = ((e^\# \circ \tau)(x_1), \dots, (e^\# \circ \tau)(x_n))$ .

To establish soundness of UA, we first show that  $\bar{\tau}$  is  $\frac{1}{2}$ -Lipschitz on the pseudometric space  $(\text{Exp}^n, d)$ , where  $d$  is the metric that arises from the  $n$ -fold product of  $(\text{Exp}, \partial)$ .

► **Lemma 24.** Given a Salomaa system  $(X, \tau : X \rightarrow \text{Exp}(X))$ , the map  $\bar{\tau} : \text{Exp}^n \rightarrow \text{Exp}^n$  from the pseudometric space  $(\text{Exp}^n, d)$  to itself is  $\frac{1}{2}$ -Lipschitz.

Finally, we can prove the following.

► **Lemma 25.** UA is satisfied by bisimilarity.

**Proof.** Let  $(X, \tau)$  is a Salomaa system, with  $X = \{x_1, \dots, x_n\}$ , and let  $f, g : X \rightarrow \text{Exp}(X)$  be solutions up to  $\dot{=}$  to the system. Finally, let  $\vec{f} = (f(x_1), \dots, f(x_n))$  and  $\vec{g} = (g(x_1), \dots, g(x_n))$ . Assume that the premises are satisfied by the bisimilarity,  $f(x_i) \sim (f^\# \circ \tau)(x_i)$  and  $g(x_i) \sim (g^\# \circ \tau)(x_i)$  for all  $1 \leq i \leq n$ . In other words, we have that  $d(\bar{\tau}(\vec{f}), \vec{f}) = 0$  and  $d(\bar{\tau}(\vec{g}), \vec{g}) = 0$

Let  $d(\bar{\tau}(\vec{f}), \bar{\tau}(\vec{g})) = k$  for some  $k \in \mathbb{R}^+$ . Then, since  $d$  is ultra,

$$d(\vec{f}, \vec{g}) = \max\{d(\vec{f}, \bar{\tau}(\vec{f})), d(\bar{\tau}(\vec{f}), \bar{\tau}(\vec{g}))\} = d(\bar{\tau}(\vec{f}), \bar{\tau}(\vec{g})) = \max\{d(\bar{\tau}(\vec{f}), \bar{\tau}(\vec{g})), d(\bar{\tau}(\vec{g}), \vec{g})\} = k$$

By Lemma 24, we find  $k = d(\bar{\tau}(\vec{f}), \bar{\tau}(\vec{g})) \leq \frac{1}{2}d(\vec{f}, \vec{g}) = \frac{1}{2}k$  which implies that  $d(\vec{f}, \vec{g}) = 0$ . Because of the definition of the product pseudometric on  $(\text{Exp}, \partial)$ , we have  $f(x_i) \sim g(x_i)$  for all  $1 \leq i \leq n$ . Therefore, the conclusion of the UA is satisfied by bisimilarity. ◀

Because of the above lemma and Theorem 13, both UA and the axioms of  $\equiv$  are contained in  $\sim$ , the greatest bisimulation on  $(\text{Exp}, \partial)$ . Recall that  $\dot{=}$  is the least congruence containing those rules. Since  $\sim$  on  $(\text{Exp}, \partial)$  is a congruence (Theorem 11), we have that  $\dot{=}$  is sound.

► **Theorem 26** (Soundness with UA). For all  $e, f \in \text{Exp}$  if  $e \dot{=} f$  then  $e \sim f$ .

**Completeness.** After all the hard work is done, the proof of completeness follows via the same line of reasoning as the one for GKAT [58, 53].

► **Theorem 27 (Completeness).** *For all  $e, f \in \text{Exp}$  if  $e \sim f$  then  $e \doteq f$*

**Proof.** Let  $R \subseteq \text{Exp} \times \text{Exp}$  be a bisimulation with a transition structure  $\rho : R \times \text{At} \rightarrow \mathcal{D}_\omega(2 + \text{Out} + \text{Act} \times R)$  relating ProbGKAT automata  $(\langle e \rangle_\partial, \partial)$  and  $(\langle f \rangle_\partial, \partial)$  such that  $(e, f) \in R$ . Let  $\pi_1, \pi_2$  be the projection homomorphisms from  $(R, \rho)$  to  $(\langle e \rangle_\partial, \partial)$  and  $(\langle f \rangle_\partial, \partial)$  respectively. Since both  $\langle e \rangle_\partial$  and  $\langle f \rangle_\partial$  are finite (by Lemma 7), so is  $R$ .

Let  $j, k$  be the inclusion homomorphisms of  $(\langle e \rangle_\partial, \partial)$  and  $(\langle f \rangle_\partial, \partial)$  in  $(\text{Exp}, \partial)$ . We can construct two homomorphisms  $[-]_{\equiv} \circ j \circ \pi_1$  and  $[-]_{\equiv} \circ k \circ \pi_2$  from  $(R, \rho)$  to  $(\text{Exp}/\equiv, \bar{\partial})$ . By Theorem 21,  $j \circ \pi_1$  and  $k \circ \pi_2$  are solutions up to  $\equiv$  to the Salomaa system associated with  $(R, \rho)$ . Since  $\equiv$  is contained in  $\doteq$ , those are immediately also solutions up to  $\doteq$ .

Because of UA, we have that  $(j \circ \pi_1)(g, h) \doteq (k \circ \pi_2)(g, h)$  for all  $(g, h) \in R$ . Thus,

$$e \doteq j(e) \doteq (j \circ \pi_1)(e, f) \doteq (k \circ \pi_2)(e, f) \doteq k(f) \doteq f \quad \blacktriangleleft$$

## 7 Decidability and Complexity

To decide whether  $e \doteq f$ , we need to demonstrate the existence of a bisimulation between the states  $e$  and  $f$  in  $(\text{Exp}, \partial)$ . Since bisimulations need only involve reachable states, it suffices to find this bisimulation within  $\langle e, f \rangle_\partial$ , the smallest subautomaton of  $(\text{Exp}, \partial)$  containing  $e$  and  $f$ , which is also the union of  $\langle e \rangle_\partial$  and  $\langle f \rangle_\partial$ ; this automaton is finite by Lemma 7. We thus focus on the problem of deciding bisimilarity within a single finite ProbGKAT automaton.

Our analysis in this section is facilitated by two simplifying assertions.

1. To avoid having to compare real (infinite-precision) probabilities, we limit ProbGKAT expressions to rational probabilities  $r \in [0, 1] \cap \mathbb{Q}$  in this section. This restriction is compatible with the earlier operators on probabilities, which all preserve rationality.
2. Equivalence of GKAT proper is co-NP-hard [58], simply because Boolean unsatisfiability can trivially be encoded in the language of tests. We take a fixed-parameter approach, assuming that  $\text{At}$ , the set of atoms that can appear on transitions, is fixed beforehand.

**Coalgebraic partition refinement.** We rely on *partition refinement* [31, 32, 48], which effectively computes the *largest bisimulation* on an automaton, by approximating it from above. In the *coalgebraic* presentation of partition refinement [67], which we instantiate to our setting, automata of various types are encoded as abstract graphs. More specifically, an automaton is encoded in two maps  $o : X \rightarrow O$  and  $\ell : X \rightarrow \mathcal{B}(L \times X)$ , where

- $X$  is a set of *nodes* that represent (partial) states of the automaton;
- $O$  is a set of *observable values* at each node;
- $L$  is a set representing possible *labels* of edges between nodes;
- $\mathcal{B}(L \times X)$  is a multiset of pairs representing *edges* between nodes.

Subject to a number of coherence conditions on the encoding (omitted here), coalgebraic partition refinement yields an  $\mathcal{O}(n \log |X|)$  algorithm to compute the largest bisimulation on an automaton, where  $n = \sum_{x \in X} |\ell(x)|$  is the number of *edges* of the automaton.

**Encoding ProbGKAT automata.** Coalgebraic partition refinement provides suitable encodings for well-known transition types, as well as methods to soundly obtain encodings of composite transition types [67]. The details of these techniques are beyond the scope of this paper, but the underlying idea is fairly intuitive: composite transition types are encoded

by inserting synthetic nodes that represent partially evaluated states – not unlike how our drawings contain intermediate nodes that are the target of  $\alpha$ -labelled arrows. More precisely, the nodes of an encoded ProbGKAT automaton  $(Q, t)$  are three-sorted:

1. every state of the automaton is a node; and
2. every “intermediate” state (the small circles in our drawings) is a node; and
3. every probabilistic edge gives rise to another node.

Nodes of the third kind separate the dashed arrows in our drawings (labelled with a probability as well as an action) into two arrows, each of which is labelled by one value.

Formally, we choose  $X := Q + I + \text{Act} \times Q$  as our set of nodes, where  $I := \{t(q)_\alpha : q \in Q, \alpha \in \text{At}\}$ . We also set  $L := \text{At} + \mathbb{Q} + \text{Act}$ . The map  $\ell : X \rightarrow \mathcal{B}(L \times X)$  is then defined by:<sup>2</sup>

$$\ell(x) := \begin{cases} \{(\alpha, t(q)_\alpha) \mid \alpha \in \text{At}\} & x = q \in Q \\ \{(d(\mathbf{p}, q), (\mathbf{p}, q)) \mid \mathbf{p} \in \text{Act}, q \in Q\} & x = d \in I \\ \{(\mathbf{p}, q)\} & x = (\mathbf{p}, q) \in \text{Act} \times Q \end{cases}$$

In other words,  $\ell$  labels the edges between nodes of the first and second kind with an atom, the edges between nodes of the second and third kind with a probability, and the edges between nodes of the third and first kind with an action.

Observables represent the probabilities assigned to acceptance, rejection, or a return value by nodes of the second kind. Formally,  $O := 1 + \mathbb{Q}^{2+\text{Out}}$ , where  $* \in 1$  means “no observable value”, and values from  $\mathbb{Q}^{2+\text{Out}}$  assign a probability to each  $\xi \in 2 + \text{Out}$ . We can then define  $o : X \rightarrow O$  by setting  $o(d)(\xi) := d(\xi)$  when  $d \in I$ , and  $o(x) := *$  otherwise.<sup>3</sup>

**Deciding bisimilarity.** We can now leverage the encoding given above to decide bisimilarity.

► **Theorem 28.** *If all probabilities are rational and  $\text{At}$  is fixed, then bisimilarity of states in a ProbGKAT automaton  $(Q, t)$  is decidable in time  $\mathcal{O}(|Q|^2 |\text{Act}| \log(|\text{Act} \times Q|))$ .*

**Proof.** The results from [67] ensure that our encoding of ProbGKAT automata can be equipped with an appropriate interface that allows their algorithm to decide equivalence.

As for the complexity, we instantiate their abstract complexity result by computing the parameters. The number of nodes and edges can be bound from above fairly easily, as follows:

$$\begin{aligned} |X| &= |Q| + |I| + |\text{Act} \times Q| \leq |Q| + 2 \cdot |\text{Act}| \cdot |Q| \\ n &= \sum_{x \in X} |\ell(x)| \leq |Q| \cdot |\text{At}| + |Q|^2 \cdot |\text{Act}| + |Q| \cdot |\text{Act}| \end{aligned}$$

Since  $\text{At}$  is fixed, the claimed complexity then follows. ◀

This allows us to conclude that bisimilarity of ProbGKAT expressions is also decidable.

► **Corollary 29.** *If all probabilities are rational and  $\text{At}$  is fixed, then ProbGKAT equivalence of  $e, f \in \text{Exp}$  is decidable in time  $\mathcal{O}(n^3 \log n)$ , where  $n = \#(e) + \#(f)$ .*

**Proof.** By Lemma 7,  $\langle e, f \rangle_\partial$  is of size at most  $n$ , and the number of distinct actions  $e$  or  $f$  is fixed from above by  $n$  as well. The claim then follows by Theorem 28. ◀

<sup>2</sup> Here,  $\{\{-\} - \}$  denotes multiset comprehension, where each element occurs at most once.

<sup>3</sup> If the coalgebraic approach from [67] is followed to the letter, the observable map for nodes of the third kind behaves slightly differently; we simplify our encoding here for the sake of presentation.



## 8 Related work

Our work builds on GKAT, a strictly deterministic fragment [39] of Kleene Algebra with Tests (KAT). KAT has been used in several verification tasks, such as cache control [15], compiler optimisations [37], source-to-source translations [3], and network properties [2, 22, 21, 59, 60, 66], and was generalised to include fuzzy logics [23]. GKAT admits a Salomaa-style [52] axiomatisation of trace equivalence [58] and bisimilarity [53], both relying on the Uniqueness of Solutions axiom, and completeness without it remains open, though completeness of a fragment of GKAT was recently proved by [33].

GKAT modulo bisimilarity and Milner’s interpretation of regular expressions arise as fragments of the *parametrised processes* framework [55]; this is not the case for ProbGKAT due to a different treatment of loops. The uniqueness axiom was originally introduced by Bergstra and Klop under the name Recursive Specification Principle (RSP) [10] and used in axiomatisations of process calculi [11]. The general pattern of their proofs of completeness is similar to ours, although the key challenge is the extension to the probabilistic setting.

Our paper also builds up the vast line of research on probabilistic bisimulation [40, 56, 18] and the coalgebraic approach to systems with probabilistic transitions [16, 8, 18, 61]. More concretely, we relied on relation refinement characterisation of bisimilarity [63], natural metrics on the final coalgebras for  $\omega$ -accessible endofunctors [7, 69], coalgebraic completeness theorems [27, 57, 54] and minimisation algorithms for coalgebras [67, 17, 68, 29]. Axiomatisations of probabilistic bisimulation were extensively studied in the process algebra community, including a recursion-free process algebra of Bandini and Segala [6] and recursive calculi of Stark and Smolka [62] and Mislove, Ouaknine and Worrell [47]. Aceto, Ésik and Ingólfssdóttir [1] gave an alternative axiomatisation of Stark and Smolka’s calculus by extending Iteration Theories [12, 20] with equational axioms.

Probabilistic Kleene Algebra (pKA) [43] relaxes the axioms of KA to accommodate reasoning about probabilistic predicate transformers; its axioms are complete w.r.t. simulation equivalence of NFAs [44]. pKA was also extended with a probabilistic choice operator and concurrency primitives [45], but completeness this system was not considered. ProbNetKAT [21, 60] is a domain-specific language for reasoning about probabilistic effects in networks based on KAT, which features a probabilistic choice operator, however, axiomatisation of the obtained language was not studied.

## 9 Conclusion and Future Work

We have presented ProbGKAT, a language for reasoning about uninterpreted programs with branching and loops, with both Boolean and probabilistic guards. We provided an automata-theoretic operational model and characterised bisimilarity for these automata. We gave a sound and complete axiomatisation of bisimilarity, relying on the Uniqueness of Solutions (UA) axiom, and showed bisimilarity can be efficiently decided in  $O(n^3 \log n)$  time.

A first natural direction for future work is the question whether the more traditional language semantics of GKAT can be lifted to ProbGKAT and axiomatised. More broadly, we would like to investigate notions of ProbGKAT expression equivalence more permissive than bisimilarity, including the notion of *bisimulation distance* [5] and its possible axiomatisations based on *quantitative equational logic* [42, 4].

A second direction touches on the problem of completeness without UA, which is still open for (Prob)GKAT. In light of recent completeness results for the skip-free fragment of GKAT [33] modulo bisimilarity and trace equivalence, we are interested to study the skip-free fragment of ProbGKAT. The proofs in [33] do not immediately generalise to ProbGKAT as probabilities do not obviously embed into (1-free) regular expressions.

Similarly to GKAT, ProbGKAT is strictly deterministic and thus avoids known complications of combining nondeterminism with probabilistic choice [30, 65, 24]. We are interested if the recent work on combining multisets and probabilities via distributive laws [28, 38] could be applied to extending our developments with nondeterminism.

ProbGKAT can express only uninterpreted programs, hence it cannot be used to reason about programs involving mutable state. An example of a probabilistic program with state is Pólya's urn [41]. One way of adding mutable state [25] to ProbGKAT is by adding *hypotheses* [14]. Unfortunately, adding hypotheses can lead to undecidability or incompleteness [35], although there are forms of hypotheses that retain completeness [36, 19, 49] and exploring this is as an interesting direction for future work.

---

## References

- 1 Luca Aceto, Zoltán Ésik, and Anna Ingólfssdóttir. Equational axioms for probabilistic bisimilarity. In *AMAST*, pages 239–253, 2002. doi:10.1007/3-540-45719-4\_17.
- 2 Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: semantic foundations for networks. In *POPL*, pages 113–126, 2014. doi:10.1145/2535838.2535862.
- 3 Allegra Angus and Dexter Kozen. Kleene algebra with tests and program schematology. Technical Report TR2001-1844, Cornell University, July 2001. URL: <https://hdl.handle.net/1813/5831>.
- 4 Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. Complete axiomatization for the bisimilarity distance on Markov chains. In *CONCUR*, pages 21:1–21:14, 2016. doi:10.4230/LIPIcs.CONCUR.2016.21.
- 5 Paolo Baldan, Filippo Bonchi, Henning Kerstan, and Barbara König. Coalgebraic behavioral metrics. *Log. Methods Comput. Sci.*, 14(3), 2018. doi:10.23638/LMCS-14(3:20)2018.
- 6 Emanuele Bandini and Roberto Segala. Axiomatizations for probabilistic bisimulation. In *ICALP*, pages 370–381, 2001. doi:10.1007/3-540-48224-5\_31.
- 7 Michael Barr. Terminal coalgebras in well-founded set theory. *Theor. Comput. Sci.*, 114(2):299–315, 1993. doi:10.1016/0304-3975(93)90076-6.
- 8 Falk Bartels, Ana Sokolova, and Erik P. de Vink. A hierarchy of probabilistic system types. In *CMCS*, pages 57–75, 2003. doi:10.1016/S1571-0661(04)80632-7.
- 9 Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva, editors. *Foundations of Probabilistic Programming*. Cambridge University Press, Cambridge, 2020. doi:10.1017/9781108770750.
- 10 Jan A. Bergstra and Jan Willem Klop. Verification of an alternating bit protocol by means of process algebra. In *Mathematical Methods of Specification and Synthesis of Software Systems*, volume 215 of *LNCS*, pages 9–23. Springer, 1985. doi:10.1007/3-540-16444-8\_1.
- 11 Jan A. Bergstra and Jan Willem Klop. Process theory based on bisimulation semantics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 50–122, 1988. doi:10.1007/BFb0013021.
- 12 Stephen L. Bloom and Zoltán Ésik. *Iteration Theories – The Equational Logic of Iterative Processes*. EATCS Monographs on Theoretical Computer Science. Springer, 1993. doi:10.1007/978-3-642-78034-9.
- 13 Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964. doi:10.1145/321239.321249.
- 14 Ernie Cohen. Hypotheses in Kleene algebra. Technical report, Bellcore, 1994.
- 15 Ernie Cohen. Lazy caching in Kleene algebra. Technical report, Bellcore, 1994.
- 16 Erik P. de Vink and Jan J. M. M. Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. *Theor. Comput. Sci.*, 221(1-2):271–293, 1999. doi:10.1016/S0304-3975(99)00035-3.

- 17 Hans-Peter Deifel, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Generic partition refinement and weighted tree automata. In *FM*, pages 280–297, 2019. doi:10.1007/978-3-030-30942-8\_18.
- 18 Josée Desharnais. *Labelled Markov processes*. PhD thesis, McGill University, 1999.
- 19 Amina Doumane, Denis Kuperberg, Damien Pous, and Pierre Pradic. Kleene algebra with hypotheses. In *FoSSaCS*, pages 207–223, 2019. doi:10.1007/978-3-030-17127-8\_12.
- 20 Calvin C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 175–230. Elsevier, 1975. doi:10.1016/S0049-237X(08)71949-9.
- 21 Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic NetKAT. In *ESOP*, pages 282–309, 2016. doi:10.1007/978-3-662-49498-1\_12.
- 22 Nate Foster, Dexter Kozen, Mae Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for NetKAT. In *POPL*, pages 343–355, 2015. doi:10.1145/2676726.2677011.
- 23 Leandro Gomes, Alexandre Madeira, and Luís Soares Barbosa. Generalising KAT to verify weighted computations. *Sci. Ann. Comput. Sci.*, 29(2):141–184, 2019. doi:10.7561/SACS.2019.2.141.
- 24 Alexandre Goy and Daniela Petrisan. Combining probabilistic and non-deterministic choice via weak distributive laws. In *LICS*, pages 454–464, 2020. doi:10.1145/3373718.3394795.
- 25 Niels Bjørn Bugge Grathwohl, Dexter Kozen, and Konstantinos Mamouras. KAT + B! In *CSL*, pages 44:1–44:10, 2014. doi:10.1145/2603088.2603095.
- 26 Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In *ICALP*, pages 299–309, 1980. doi:10.1007/3-540-10003-2\_79.
- 27 Bart Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, pages 375–404, 2006. doi:10.1007/11780274\_20.
- 28 Bart Jacobs. From multisets over distributions to distributions over multisets. In *LICS*, pages 1–13, 2021. doi:10.1109/LICS52264.2021.9470678.
- 29 Jules Jacobs and Thorsten Wißmann. Fast coalgebraic bisimilarity minimization. In *POPL*, pages 1514–1541, 2023. doi:10.1145/3571245.
- 30 Claire Jones. *Probabilistic non-determinism*. PhD thesis, University of Edinburgh, UK, 1990. URL: <https://hdl.handle.net/1842/413>.
- 31 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *PODC*, pages 228–240, 1983. doi:10.1145/800221.806724.
- 32 Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inf. Comput.*, 86(1):43–68, 1990. doi:10.1016/0890-5401(90)90025-D.
- 33 Tobias Kappé, Todd Schmid, and Alexandra Silva. A complete inference system for skip-free guarded Kleene algebra with tests. In *ESOP*, pages 309–336, 2023. doi:10.1007/978-3-031-30044-8\_12.
- 34 Donald E. Knuth and Andrew C. Yao. The complexity of nonuniform random number generation. In *Algorithms and Complexity: New Directions and Recent Results*, 1976.
- 35 Dexter Kozen. On the complexity of reasoning in Kleene algebra. *Inf. Comput.*, 179(2):152–162, 2002. doi:10.1006/inco.2001.2960.
- 36 Dexter Kozen and Konstantinos Mamouras. Kleene algebra with equations. In *ICALP (Part II)*, pages 280–292, 2014. doi:10.1007/978-3-662-43951-7\_24.
- 37 Dexter Kozen and Maria-Christina Patron. Certification of compiler optimizations using Kleene algebra with tests. In *CL*, pages 568–582, 2000. doi:10.1007/3-540-44957-4\_38.
- 38 Dexter Kozen and Alexandra Silva. Multisets and distributions, 2023. arXiv:2301.10812.
- 39 Dexter Kozen and Wei-Lung Dustin Tseng. The Böhm-Jacopini theorem is false, propositionally. In *MPC*, pages 177–192, 2008. doi:10.1007/978-3-540-70594-9\_11.

- 40 Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991. doi:10.1016/0890-5401(91)90030-6.
- 41 Hosam Mahmoud. *Pólya Urn Models*. Texts in Statistical Science. Chapman & Hall, 2008.
- 42 Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. Quantitative algebraic reasoning. In *LICS*, pages 700–709, 2016. doi:10.1145/2933575.2934518.
- 43 Annabelle McIver, Carlos González, Ernie Cohen, and Carroll C. Morgan. Using probabilistic Kleene algebra pKA for protocol verification. *J. Log. Algebraic Methods Program.*, 76(1):90–111, 2008. doi:10.1016/j.jlap.2007.10.005.
- 44 Annabelle McIver, Tahiry M. Rabehaja, and Georg Struth. On probabilistic Kleene algebras, automata and simulations. In *RAMICS*, pages 264–279, 2011. doi:10.1007/978-3-642-21070-9\_20.
- 45 Annabelle McIver, Tahiry M. Rabehaja, and Georg Struth. Probabilistic concurrent Kleene algebra. In *QAPL*, pages 97–115, 2013. doi:10.4204/EPTCS.117.7.
- 46 Robin Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984. doi:10.1016/0022-0000(84)90023-0.
- 47 Michael W. Mislove, Joël Ouaknine, and James Worrell. Axioms for probability and non-determinism. In *EXPRESS*, pages 7–28, 2003. doi:10.1016/j.entcs.2004.04.019.
- 48 Robert Paige and Robert Endre Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987. doi:10.1137/0216062.
- 49 Damien Pous, Jurriaan Rot, and Jana Wagemaker. On tools for completeness of kleene algebra with hypotheses, 2022. arXiv:2210.13020.
- 50 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.
- 51 Wojciech Różowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva. Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity, 2023. arXiv:2305.01755.
- 52 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966. doi:10.1145/321312.321326.
- 53 Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. In *ICALP*, pages 142:1–142:14, 2021. doi:10.4230/LIPIcs.ICALP.2021.142.
- 54 Todd Schmid, Jurriaan Rot, and Alexandra Silva. On star expressions and coalgebraic completeness theorems. In *MFPS*, pages 242–259, 2021. doi:10.4204/EPTCS.351.15.
- 55 Todd Schmid, Wojciech Różowski, Alexandra Silva, and Jurriaan Rot. Processes parametrised by an algebraic theory. In *ICALP*, 2022. doi:10.4230/LIPIcs.ICALP.2022.132.
- 56 Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR*, pages 481–496, 1994. doi:10.1007/978-3-540-48654-1\_35.
- 57 Alexandra Silva. *Kleene coalgebra*. PhD thesis, University of Nijmegen, 2010.
- 58 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: verification of uninterpreted programs in nearly linear time. In *POPL*, pages 61:1–61:28, 2020. doi:10.1145/3371129.
- 59 Steffen Smolka, Praveen Kumar, Nate Foster, Dexter Kozen, and Alexandra Silva. Cantor meets Scott: semantic foundations for probabilistic networks. In *POPL*, pages 557–571, 2017. doi:10.1145/3009837.3009843.
- 60 Steffen Smolka, Praveen Kumar, David M. Kahn, Nate Foster, Justin Hsu, Dexter Kozen, and Alexandra Silva. Scalable verification of probabilistic networks. In *PLDI*, pages 190–203, 2019. doi:10.1145/3314221.3314639.
- 61 Ana Sokolova. Probabilistic systems coalgebraically: A survey. *Theor. Comput. Sci.*, 412(38):5095–5110, 2011. doi:10.1016/j.tcs.2011.05.008.
- 62 Eugene W. Stark and Scott A. Smolka. A complete axiom system for finite-state probabilistic processes. In *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 571–596, 2000.

- 63 Sam Staton. Relating coalgebraic notions of bisimulation. *Log. Methods Comput. Sci.*, 7(1), 2011. doi:10.2168/LMCS-7(1:13)2011.
- 64 Joseph Aaron Toumanios. Three sided die, 2019. US patent 10384119. URL: <https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/10384119>.
- 65 Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006. doi:10.1017/S0960129505005074.
- 66 Jana Wagemaker, Nate Foster, Tobias Kappé, Dexter Kozen, Jurriaan Rot, and Alexandra Silva. Concurrent NetKAT – Modeling and analyzing stateful, concurrent networks. In *ESOP*, pages 575–602, 2022. doi:10.1007/978-3-030-99336-8\_21.
- 67 Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. Efficient and modular coalgebraic partition refinement. *Logical Methods in Computer Science*, 16:1:8:1–8:63, 2020. doi:10.23638/LMCS-16(1:8)2020.
- 68 Thorsten Wißmann, Stefan Milius, and Lutz Schröder. Quasilinear-time computation of generic modal witnesses for behavioural inequivalence. *Log. Methods Comput. Sci.*, 18(4), 2022. doi:10.46298/lmcs-18(4:6)2022.
- 69 James Worrell. On the final sequence of a finitary set functor. *Theor. Comput. Sci.*, 338(1-3):184–199, 2005. doi:10.1016/j.tcs.2004.12.009.