A Kleene algebra with tests for union bound reasoning on probabilistic programs

Leandro Gomes, Patrick Baillot, Marco Gaboardi

A Kleene algebra with tests for union bound reasoning on probabilistic programs

Leandro Gomes[a1], Patrick Baillot[b1], and Marco Gaboardi[c2]

[1]Université de Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2]Boston University, USA

[a]leandrogomes.moreiragomes@univ-lille.fr
[b]patrick.baillot@univ-lille.fr
[c]gaboardi@bu.edu

Abstract

Kleene Algebra with Tests (KAT) provides a framework for algebraic equational reasoning on imperative programs. The recent variant Guarded KAT (GKAT) allows to reason on non-probabilistic properties of probabilistic programs. Here we introduce an extension of this framework called aGKAT (approximate GKAT), a form of graded GKAT over a partially ordered monoid (real numbers) which enables to express satisfaction of (deterministic) properties *except* with a probability up to a certain bound. This allows to represent in equational reasoning 'à la KAT' proofs of probabilistic programs based on the union bound, a technique from basic probability theory. We show how a propositional variant of approximate Hoare Logic (aHL), a program logic for union bound, can be soundly encoded in our system aGKAT. We then illustrate the use of aGKAT on an example of accuracy analysis from the field of differential privacy.

Keywords: Kleene algebras with tests, Hoare logic, equational reasoning, probabilistic programs, union bound, formal verification

1 Introduction

Kleene algebra with tests (KAT) have been introduced by Kozen [12] as an algebraic framework for program verification. A KAT consists in a Kleene algebra containing a Boolean algebra of tests. The Kleene algebra part accounts for sequential composition, branching and iteration, and the Boolean algebra part accounts for the conditions in the if-then-else instructions, while loops, assertions, as well as, considering KAT a tool able to subsume propositional Hoare logic [13], for the pre and post-conditions. This framework allowed to give algebraic proofs corresponding to several approaches in program verification, see e.g. [13, 1, 14]. It has been implemented as a library for the Coq proof assistant [17]. It has also been followed by several variants, like for instance NetKAT [2] which allows to reason about software defined networks, Concurrent NetKAT [20] for concurrent networks, or TopKAT for reasoning about incorrectness [21].

Recently the variant Guarded KAT (GKAT) [19] has been proposed. This variant consists in a restriction of KAT where all sums and iterations are guarded by tests. It offers several advantages over KAT, including the fact that the complexity of its equational theory is lower (almost linear time) and the fact that it admits a probabilistic model. This latter property paves the way for using GKAT for reasoning about probabilistic programs. However an important feature of this system is that the tests of GKAT remain the same as those of KAT, namely they express deterministic (boolean) properties on states. Therefore the framework of GKAT allows to reason about probabilistic programs, but only for establishing deterministic (i.e. non probabilistic) properties.

In this paper our goal is to extend the GKAT approach to reason about probabilistic programs by allowing to handle not only deterministic properties, but also satisfaction of a property *with a given probability bound* β. Our objective is not to design an expressive framework for advanced probabilistic proofs, but instead to allow for simple probabilistic reasoning with a low technical overhead. Namely we target proofs based on the *union bound principle*, a property from basic probability theory. It can be stated as follows: given some properties $A_1, \ldots, A_n$, one has:

$$Pr[\cup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} Pr[A_i]$$

This principle is ubiquitus when reasoning about properties of randomized algorithms [16] and in their application in security, privacy [6], learning theory [11], etc.

A previous approach for reasoning about probabilistic imperative programs using the union bound principle had been provided by the *union bound program logic* aHL of [4]. This is a Hoare logic for reasoning about probabilistic programs with non-probabilistic assertions but with judgements carrying a numeric index for tracking the failure probability. That is, judgments have the form $\vdash_\beta c : \phi \Rightarrow \psi$ where β is an upper bound on the probability that $\neg\psi$ is true after executing c starting from a memory satisfying ϕ. The authors illustrated how this logic could be used for the verification of accuracy of both non-interactive and interactive algorithms, in particular in the setting of differential privacy.

A natural approach is trying to adapt this idea to the GKAT framework. To do this and capture the union bound reasoning in an algebraic framework we extend GKAT with an additional relation, denoted $\triangleleft$, relating GKAT expressions with elements of a partially ordered monoid, typically real numbers. We call this new system *approximate* GKAT (aGKAT). An important feature is that we want the new setting to subsume standard GKAT, and so for that we require aGKAT to satisfy GKAT's theory. A second feature is that we want the probabilistic model of sub-Markov kernels to be a model of our new structure, when we consider the monoid of real numbers. For this particular instantiation of our structure, the meaning of the new relation $\triangleleft$ will be that $c \triangleleft \beta$ holds if *the program c's probability of successful execution is bounded by* β. The theory of aGKAT extends the one of GKAT, by a small set of axioms characterizing the properties of the new relation $\triangleleft$. We illustrate how this theory allows for a concise form of equational reasoning for establishing probability bounds on some GKAT programs. Moreover in order to demonstrate the expressivity of our system aGKAT, we show how propositional aHL can be encoded in it. This is inspired by the classical result of Kozen [13] showing that propositional Hoare logic can be encoded in KAT. Thus aGKAT can be seen as an algebraic counterpart of propositional aHL, similarly as KAT is the algebraic counterpart of propositional Hoare logic.

Outline. In Sect. 2 we will recall the background on GKAT and its probabilistic model, and in Sect. 3 we will recall the Hoare logic aHL. Then in Sect 4 we will define our system, aGKAT, its theory and its semantics. After that in Sect. 5 we will provide an encoding of the logic aHL in the algebra aGKAT and prove its soundness. Finally Sect. 7 will be devoted to an example, the analysis in aGKAT of the accuracy of the probabilistic algorithm Report-noisy-max.

2 Guarded Kleene algebra with tests

This section recalls the language and the semantics of *Guarded Kleene Algebra with Tests (GKAT)* [19], an abstraction of imperative programs where conditionals ($c_1 +_b c_2$) and loops ($c^{(b)}$) are guarded by Boolean predicates b. The structure is a restriction of KAT in which we are not allowed to freely use operators $+$ and * to build terms, i.e. GKAT does not allow nondeterminism. Although less expressive that KAT, GKAT offers two advantages: decidability in (almost) linear time (compared to PSPACE complexity of decidability in KAT), and better foundation for probabilistic applications. Although the first one was the main motivation to introduce the structure [19], we are more interested in the second advantage for the purpose of this paper.

2.1 Syntax

The syntax of GKAT is defined with a set of *actions* Σ and a finite set of primitive tests T, which are disjoint. We denote actions by a and primitive tests by p. The classes of boolean expressions BExp (also called tests) and GKAT expressions Exp (also called programs) are then defined by the following grammars:

$b, b_1, b_2 \in \text{BExp} ::=$			$c, c_1, c_2 \in \text{Exp} ::=$	
|	0	**false**	| 1	**skip**
|	1	**true**	| $a \in \Sigma$	**do** a
|	$p \in T$	p	| $b \in \text{BExp}$	**assert** b
|	$b_1 \cdot b_2$	b_1 **and** b_2	| $c_1 \cdot c_2$	$c_1;\ c_2$
|	$b_1 + b_2$	b_1 **or** b_2	| $c_1 +_b c_2$	**if** b **then** c_1 **else** c_2
|	$\neg\ b$	**not** b	| $c^{(b)}$	**while** b **do** c

where, for any $b, b_1, b_2 \in \text{BExp}$, operators $\cdot$, $+$ and $\neg$ denote conjunction, disjunction and negation, respectively, and, for any $c, c_1, c_2 \in \text{Exp}$, operator $\cdot$ denotes sequential composition. We will also write $\overline{b}$ for $\neg\ b$. The notations on the right-hand-side are given to help intuition and will sometimes be used when writing programs. The Boolean expression 1, as it is also an element of Exp, encodes command **skip**.
The precedence of the operators is the usual one. To simplify the writing, we often omit the operator $\cdot$ by writing $c_1 c_2$ for the expression $c_1 \cdot c_2$, for any $c_1, c_2 \in \text{Exp}$.

For defining actions and primitive tests we will now introduce some terms:

$$t \in \text{Terms} ::= x \in \mathit{Var} \mid r \in \mathbb{R} \mid t_1 + t_2 \mid t_1 - t_2 \mid t_1 \times t_2$$

The set of primitive tests is defined by $T = \{t_1 < t_2,\ t_1 = t_2 \mid t_1, t_2 \in \text{Terms}\}$.

In this paper we are interested in using GKAT for representing probabilistic programs. Let us for that first fix a few definitions. Given a set S, $\mathcal{D}(S)$ is the set of *probability sub-distributions* over S with discrete support, i.e. the set of functions $f : S \to [0,1]$ such that $\mathit{Supp}(f) = \{x \in [0,1] \mid f(x) > 0\}$ is discrete and f sums up to at most 1, i.e. $\sum_{s \in S} f(s) \leq 1$. In particular, the *Dirac* distribution $\delta_s \in \mathcal{D}(S)$ is the map $w \to [w = s] = \begin{cases} 1, \text{ if w=s} \\ 0, \text{ otherwise} \end{cases}$

We will thus consider a set Distr of sub-distributions over $\mathbb{R}$ with discrete support and two kinds of actions, assignment of value to a variable and sampling from a distribution. The set of actions is thus:

$$\Sigma = \{x \leftarrow t,\ x \overset{\$}{\leftarrow} d \mid x \in \mathit{Var}, t \in \text{Terms}, d \in \text{Distr}\}.$$

The first action evaluates term t and assigns the result to x and the second one samples from d and assigns the result to x. Some examples of distributions are the tossing of a fair coin, with probability 0.5 for 0 and 1 only, that we will denote *Coin*, and the (discrete version of) Laplacian distribution $\mathcal{L}_p(a)$ centered in a with parameter p. The density function of $\mathcal{L}_p(a)$ is given by $\frac{1}{2p} \exp(\frac{|x-a|}{p})$.

Observe that while actions and thus GKAT expressions, that is to say programs, are probabilistic, because of samplings, the tests themselves are deterministic and have values true or false. In particular the conditional branching is only done on deterministic tests.

2.2 Semantics

We now present the semantic interpretation of GKAT that we will be using, the *Probabilistic model* [19]. Note that more interpretations of GKAT are presented in [19], namely a relational model and a language model. We first revise some basic concepts needed for the semantics. The *Iverson bracket* $[p]$, for p a predicate term, is the function taking value 1 if p is true and 0 otherwise. Typical models of probabilistic imperative programming languages interpret programs as *Markov kernels*, i.e. maps from S to probability distributions. The semantic model defined below interprets programs as *sub-Markov* kernels, i.e. Markov kernels over sub-distributions.

Definition 2.1 (Probabilistic interpretation)**.** *Let $i = (\mathit{State}, \mathit{eval}, \mathit{sat})$ be a triple where:*

- *State is a set of states,*
- *for each action $a \in \Sigma$, $\mathit{eval}(a) : \mathit{State} \to \mathcal{D}(\mathit{State})$ is a sub-Markov kernel,*
- *for each primitive test $p \in T$, $\mathit{sat}(p) \subseteq \mathit{State}$ is a set of states.*

The *probabilistic interpretation* of an expression c with relation to i is the sub-Markov kernel $\mathcal{P}_i[\![c]\!] : State \to \mathcal{D}(State)$ defined as follows:

1. $\mathcal{P}_i[\![a]\!] := eval(a)$
2. $\mathcal{P}_i[\![b]\!](\sigma) := [\sigma \in sat^\dagger(b)] \times \delta_\sigma$
3. $\mathcal{P}_i[\![c_1 \cdot c_2]\!](\sigma)(\sigma') := \sum_{\sigma''} \mathcal{P}_i[\![c_1]\!](\sigma)(\sigma'') \times \mathcal{P}_i[\![c_2]\!](\sigma'')(\sigma')$
4. $\mathcal{P}_i[\![c_1 +_b c_2]\!](\sigma) := [\sigma \in sat^\dagger(b)] \times \mathcal{P}_i[\![c_1]\!](\sigma) + [\sigma \in sat^\dagger(\neg\, b)] \times \mathcal{P}_i[\![c_2]\!](\sigma)$
5. $\mathcal{P}_i[\![c^{(b)}]\!](\sigma)(\sigma') := \lim_{n\to\infty} \mathcal{P}_i[\![(c +_b 1)^n \cdot \neg\, b]\!](\sigma)(\sigma')$

where $\mathrm{sat}^\dagger : \mathrm{BExp} \to 2^{State}$ is the lifting of $\mathrm{sat} : T \to 2^{State}$ to arbitrary Boolean expressions over T.

Remark 2.1 (Finite state case)**.** *In the case where State is a finite set of size n, say $\{s_1, \ldots, s_n\}$ then a sub-Markov kernel f can be represented as an $n \times n$ matrix $\mathcal{M} = (a_{i,j})_{i,j\in[1,n]}$. Each coefficient $a_{i,j}$ is defined as $a_{i,j} = f(s_i)(s_j)$. So in particular the sum over each line is inferior or equal to 1. We denote abusively: $f = \mathcal{M}$. In the case of a test b the matrix $\mathcal{P}_i[\![b]\!]$ has only diagonal coefficients, with value $a_{i,i} = 1$ if $b(s_i) = 1$, $a_{i,i} = 0$ if $b(s_i) = 0$. In the case of $c_1 \cdot c_2$, the matrix $\mathcal{P}_i[\![c_1 \cdot c_2]\!]$ is obtained by the matrix product of $\mathcal{P}_i[\![c_1]\!]$ and $\mathcal{P}_i[\![c_2]\!]$.*

In the following we will consider programs over a finite set of variables *Var* and the set of states will be the set of *memories*, that is to say functions in $Var \to D$ where D is the domain of variables (we can take for instance $D = \mathbb{Q}$, the rational numbers). If $x \in Var$ and σ is a memory, then $\sigma[x \leftarrow t]$ is the memory identical to σ except that it maps x to the evaluation of t in memory σ. The interpretation of actions $a \in \Sigma$ as sub-Markov Kernels is then given by:
$eval(x \leftarrow t)(\sigma) := \delta_{\sigma[x\leftarrow t]}$ and $eval(x \xleftarrow{\$} d)(\sigma) := \sum_{t\in Supp(d)} d(t) \cdot \delta_{\sigma[x\leftarrow t]}$.

In the sequel memories will often be denoted as m.

2.3 Axioms

The theory of GKAT introduced in [19] is given by the axioms from Fig. 1. Note in particular the fixpoint

$$
\begin{array}{rcll}
c +_b c & = & c & (1)\\
c_1 +_b c_2 & = & c_2 +_{\neg\, b} c_1 & (2)\\
(c_1 +_{b_1} c_2) +_{b_2} c_3 & = & c_1 +_{b_1 b_2} (c_2 +_{b_3} c_3) & (3)\\
c_1 +_b c_2 & = & bc_1 +_b c_2 & (4)\\
c_1 c_3 +_b c_2 c_3 & = & (c_1 +_b c_2) \cdot c_3 & (5)\\
(c_1 \cdot c_2) \cdot c_3 & = & c_1 \cdot (c_2 \cdot c_3) & (6)\\
0 \cdot c & = & 0 & (7)
\end{array}
\qquad
\begin{array}{rcll}
c \cdot 0 & = & 0 & (8)\\
1 \cdot c & = & c & (9)\\
c \cdot 1 & = & c & (10)\\
c^{(b)} & = & cc^{(b)} +_b 1 & (11)\\
(c +_{b_2} 1)^{(b_1)} & = & (b_2 c)^{(b_1)} & (12)\\
\dfrac{c_3 = c_1 c_3 +_b c_2}{c_3 = c_1^{(b)} c_2} & \text{if} & E(c_1) = 0 & (13)
\end{array}
$$

Figure 1: Axiomatisation of Guarded Kleene algebra with tests

axiom (13). Intuitively, it says that if expression c_3 chooses (using guard b) between executing c_1 and looping again, and executing c_2, then c_3 is a b-guarded loop followed by c_2. However, the rule is not sound in general (see [19] for more details). In order to overcome such limitation, the side condition $E(c_1) = 0$ is introduced, ensuring that command c_1 is *productive*, i.e. that it performs some action. To this end, the function E is inductively defined as follows: $E(b) := b$, $E(a) := 0$, $E(c_1 +_b c_2) := b \cdot E(c_1) + \neg\, b \cdot E(c_2)$, $E(c_1 \cdot c_2) := E(c_1) \cdot E(c_2)$, $E(c^{(b)}) := \neg\, b$. We can see $E(c)$ as the weakest test that guarantees that command c terminates successfully but does not perform any action.

Moreover, note particularly the following observation: in KAT the encoding $c_1 \cdot (b \cdot c_2 + \neg\, b \cdot c_3) = c_1 \cdot b \cdot c_2 + c_1 \cdot \neg\, b \cdot c_3$ is not an **if-then-else** statement; it is rather a nondeterministic choice between executing c_1, then testing b and executing c_2, and executing c_1, then testing $\neg\, b$ and executing c_3. That is why left distributivity does not hold in GKAT for any $c \in \mathrm{Exp}$; it only holds for the particular case of $c_1 \in \mathrm{BExp}$, i.e. if c_1 is a test.

Note that $+$ is defined on tests and represents disjunction. We define the relation $\leq$ on tests as: $b_1 \leq b_2$ iff $b_1 + b_2 = b_2$. Contrarily to KAT [12], the relation $\leq$ is not defined on any GKAT expression, only on tests.

In the Appendix A we list additional derivable equations in GKAT, which were given in [19].

Since any test is a program ($\texttt{Bexp} \subseteq \texttt{Exp}$), the grammar also allows to write expressions as $b_1 +_b b_2$, for any $b \in \mathrm{BExp}$. We thus establish the following proposition[1] which expresses the guarded sum $+_b$, for any $b \in \mathrm{BExp}$, in terms of the disjunction $+$ on tests.

Proposition 2.1.

$$b_1 +_b b_2 = b \cdot b_1 + \neg\, b \cdot b_2 \tag{14}$$

By Boolean reasoning, we can observe that $b \cdot b + \neg\, b \cdot \neg\, b = 1$. Such observation will be useful later to prove the soundness of some aHL rules in aGKAT.

We also have the following property (see Appendix C for the proof):

Proposition 2.2.

$$b_1 + b_2 = b_1 +_{b_1} b_2 \tag{15}$$

3 Union bound logic - Approximate Hoare logic

Observe that GKAT allows to reason on probabilistic programs but only for establishing *deterministic* properties. We want to extend this kind of reasoning.

For that we start in this section by recalling *Aproximate Hoare logic (aHL)* [4], a logic based on the union bound, a tool from probability theory for analyzing randomised algorithms. A judgment in aHL is of the form:

$$\vdash_\beta c : \phi \Rightarrow \psi$$

where ϕ, ψ are first-order formulas representing pre- and post-conditions, respectively. Note that ϕ and ψ are non-probabilistic assertions, i.e. they represent properties of memories rather that properties of distributions over memories. This means that $\models \phi$ states that ϕ is valid in memory m. The value β belongs to $[0, 1]$ and is an upper bound on the probability that the postcondition ψ does *not* hold on the output distribution, assuming that ϕ holds on the initial memory. The validity of the judgment is thus stated as follows:

Definition 3.1 (Validity of aHL judgment)**.** *A judgment $\vdash_\beta c : \phi \Rightarrow \psi$ is valid if for every memory m such that $m \models \phi$, we have:*

$$\mathcal{P}_i[\![c]\!](m)[\neg\, \psi] \leq \beta$$

Figure 2 presents the deduction rules of aHL. In rule (*Weak*) the premise $\models \phi' \Rightarrow \phi$ means that, in any model, ϕ' implies ϕ. Observe that the (*While*) rule is slightly more restrictive than the usual one of Hoare logic. Its side conditions ensure that the loop terminates in at most k iterations except with probability $k \cdot \beta$.

4 Approximate Guarded Kleene algebra with tests (aGKAT)

4.1 Definition and theory of aGKAT

We want to define an extension of GKAT which would allow to express the fact that a probabilistic program c satisfies a deterministic postcondition, *except* with a probability up to a certain bound, in the spirit of the logic aHL that we have just recalled. For that we will extend GKAT with a possibility to relate a GKAT expression with a value β from a partially ordered set. Let us start by giving some definitions.

We call *preordered double monoid* (pod-monoid) a structure $\mathcal{M} = (M, \leq, \cdot, 1, +, 0)$ where:

- $\leq$ is a preorder on M,
- $(M, \cdot, 1)$ and $(M, +, 0)$ are two monoid structures, whose laws are compatible with $\leq$.

[1] We thank the anonymous reviewer of another paper for pointing out to us the fact that this property is derivable in GKAT.

- *Skip*:

$$\frac{}{\vdash_0 \texttt{skip} : \phi \Rightarrow \phi}$$

- *Assn*:

$$\frac{}{\vdash_0 \textbf{do } x \leftarrow t : \phi[t/x] \Rightarrow \phi}$$

- *Rand*:

$$\frac{\forall\, m \cdot m \models \phi \Rightarrow \mathcal{P}_i[\![x \xleftarrow{\$} d]\!](m)[\neg\, \psi] \leq \beta}{\vdash_\beta \textbf{do } x \xleftarrow{\$} d : \phi \Rightarrow \psi}$$

- *Seq*:

$$\frac{\vdash_\beta c : \phi \Rightarrow \phi' \quad \vdash_{\beta'} c' : \phi' \Rightarrow \phi''}{\vdash_{\beta+\beta'} c;\ c' : \phi \Rightarrow \phi''}$$

- *Cond*:

$$\frac{\vdash_\beta c : \phi \wedge b \Rightarrow \psi \quad c' : \phi \wedge \neg\, b \Rightarrow \psi}{\vdash_\beta \textbf{if } b \textbf{ then } c \textbf{ else } c' : \phi \Rightarrow \psi}$$

- *Weak*:

$$\frac{\models \phi' \Rightarrow \phi \quad \vdash_\beta c : \phi \Rightarrow \psi \quad \models \psi \Rightarrow \psi' \quad \beta \leq \beta'}{\vdash_{\beta'} c : \phi' \Rightarrow \psi'}$$

- *Or*:

$$\frac{\vdash_\beta c : \phi \Rightarrow \psi \quad \vdash_\beta c : \phi' \Rightarrow \psi}{\vdash_\beta c : \phi \vee \phi' \Rightarrow \psi}$$

- *And*:

$$\frac{\vdash_\beta c : \phi \Rightarrow \psi \quad \vdash_{\beta'} c : \phi \Rightarrow \psi'}{\vdash_{\beta+\beta'} c : \phi \Rightarrow \psi \wedge \psi'}$$

- *False*:

$$\frac{}{\vdash_1 c : \phi \Rightarrow \bot}$$

- *While*:

$$\frac{b_v : \mathbb{N} \quad \models \phi \wedge b_v \leq 0 \rightarrow \neg\, b \quad \vdash_\beta c : \phi \Rightarrow \phi \quad \forall_{\eta>0} \cdot \vdash_0 c : \phi \wedge b \wedge (b_v = \eta) \Rightarrow (b_v < \eta)}{\vdash_{k\cdot\beta} \textbf{while } b \textbf{ do } c : \phi \wedge (b_v \leq k) \Rightarrow \phi \wedge \neg\, b}$$

Figure 2: Approximate Hoare Logic rules (aHL)

Note that we do not request here any property relating $\cdot$ and $+$. In the sequel we will consider the pod-monoid consisting of the real unit interval $[0,1]$ equipped with multiplication and addition truncated to 1, that is to say $min\,((\beta_1+\beta_2),1)$, where $+$ is the ordinary addition.

Definition 4.1. *An approximate GKAT, denoted aGKAT, is a triple $(\mathbb{A},\mathcal{M},\triangleleft)$ where $\mathbb{A}$ is a GKAT, $\mathcal{M}$ is a pod-monoid and $\triangleleft$ is a relation on $\mathbb{A}\times\mathcal{M}$ satisfying the equations of Fig. 3.*

The following axioms are quantified universally for any GKAT expressions c, c_1, c_2, tests e and monoid elements β, β_1, β_2,

$$(c_1 = c_2 \ \wedge\ c_1 \triangleleft \beta) \ \Rightarrow\ c_2 \triangleleft \beta \tag{16}$$
$$(c \triangleleft \beta_1 \ \wedge\ \beta_1 \leq \beta_2) \ \Rightarrow\ c \triangleleft \beta_2 \tag{17}$$
$$(c_1 \triangleleft \beta_1 \ \wedge\ c_2 \triangleleft \beta_2) \ \Rightarrow\ c_1 \cdot c_2 \triangleleft \beta_1 \cdot \beta_2 \tag{18}$$
$$(c_1 \triangleleft \beta \ \wedge\ c_2 \triangleleft \beta) \ \Rightarrow\ c_1 +_b c_2 \triangleleft \beta \tag{19}$$
$$(c \cdot c_1 \triangleleft \beta_1 \ \wedge\ c \cdot c_2 \triangleleft \beta_2) \ \Rightarrow\ c \cdot (c_1 +_b c_2) \triangleleft \beta_1 + \beta_2 \tag{20}$$
$$c \triangleleft 1 \tag{21}$$
$$0 \triangleleft 0 \tag{22}$$

Figure 3: Axioms on the relation $\triangleleft$

We define a semantic interpretation of aGKAT as follows:

Definition 4.2. *The probabilistic semantic interpretation of an aGKAT $(\mathbb{A},\mathcal{M},\triangleleft)$ is obtained by extending the probabilistic interpretation $\mathcal{P}_i$ of GKAT recalled in Sect. 2.2 in the following way:*

- *we consider the triple $i = (State, eval, sat)$ of Def. 2.1 which gives an interpretation of the GKAT $\mathbb{A}$,*
- *the pod-monoid $\mathcal{M}$ is interpreted as indicated above by $([0,1],\leq,\cdot,1,+,0)$ where $\cdot$ is the product and $+$ the truncated sum,*
- *the relation $\triangleleft$ is interpreted by the relation between sub-Markov kernels f and $[0,1]$-reals β consisting in the pairs (f,β) satisfying:*

$$\forall\, s \in State,\ \Sigma_{s'\in State} f(s)(s') \leq \beta \tag{23}$$

that is to say, for any state s, the total mass of the sub-distribution $f(s)$ is inferior or equal to β.

We still denote this enlarged interpretation as $\mathcal{P}_i$.

If a program c satisfies property (23) w.r.t. β, that is to say if:

$$\forall_{s\in State},\ \sum_{s'\in State} \mathcal{P}_i[\![c]\!](s)(s') \leq \beta$$

then we will write $c \blacktriangleleft \beta$.

Proposition 4.1. *The probabilistic interpretation of an aGKAT $(\mathbb{A},\mathcal{M},\triangleleft)$ from Def. 4.2 gives a model of $(\mathbb{A},\mathcal{M},\triangleleft)$, that is to say:*

1. *the interpretation of $\mathbb{A}$ satisfies the axioms of GKAT (Fig. 1) and that of $([0,1],\leq,\cdot,1,+,0)$ satisfies the axioms of pod-monoid,*
2. *the axioms of Fig.3 (axioms (16) to (22)) are satisfied.*

So Proposition 4.1 implies that if i is a probabilistic interpretation and if a statement $c \triangleleft \beta$ is derivable with the aGKAT axioms, then the semantic property $c \blacktriangleleft \beta$ holds.

Proof. (Prop. 4.1)

The fact that the interpretation of $\mathbb{A}$ satisfies the axioms of GKAT is known from [19].

We give here the proof that the interpretation of $\mathbb{A}$ satisfies Axiom (18). The proofs for the other axioms are given in Appendix D.

Consider a set of states S. By assumption we know that:

$$\forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') \leq \beta_1, \ \ \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_2]\!](s)(s') \leq \beta_2$$

$$\begin{aligned}
\text{So we have } \ \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1 \cdot c_2]\!](s)(s') &= \sum_{s'\in S}\sum_{s''\in S} \mathcal{P}_i[\![c_1]\!](s)(s'') \cdot \mathcal{P}_i[\![c_2]\!](s'')(s') \\
&= \sum_{s'\in S} (\mathcal{P}_i[\![c_1]\!](s)(s') \cdot \sum_{s''\in S} \mathcal{P}_i[\![c_2]\!](s'')(s')) \text{ commutativity of } + \\
&= \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') \cdot \beta_2 \text{ by assumptions} \\
&= \beta_1 \cdot \beta_2
\end{aligned}$$

□

Remark 4.1. *Note that by analogy with Axiom (19), one could have expected an axiom stronger than Axiom (20), namely that if* $(c \cdot c_1 \triangleleft \beta \ \wedge \ c \cdot c_2 \triangleleft \beta)$ *then one would have* $c \cdot (c_1 +_b c_2) \triangleleft \beta$ *(this would then generalize Axiom (19) when taking* $c = 1$*). However it turns out that this candidate additional axiom is not valid in the probabilistic model. A counter-example is given in the Appendix E.*

Proposition 4.2. *The following implication holds in any aGKAT:*

$$(c \cdot b_1 \triangleleft \beta_1 \ \wedge \ c \cdot b_2 \triangleleft \beta_2) \Rightarrow c \cdot (b_1 + b_2) \triangleleft \beta_1 + \beta_2$$

Proof. Observe that $b_1 + b_2 = b_1 +_{b_1} b_2$ by Prop. 2.2 and use Axiom (20). □

The following proposition refines in some sense Axiom (20).

Proposition 4.3. *The following implication holds in any aGKAT:*

$$(c \cdot b \cdot c_1 \triangleleft \beta_1 \ \wedge \ c \cdot \overline{b} \cdot c_2 \triangleleft \beta_2) \Rightarrow c \cdot (c_1 +_b c_2) \triangleleft \beta_1 + \beta_2$$

Proof. Observe that $c_1 +_b c_2 = b \cdot c_1 +_b \overline{b} \cdot c_2$ by Axiom 4, Axiom 2, applied two times .Then apply Axiom (20) to c, $c_1' = bc_1$ and $c_2' = bc_2$. □

Remark 4.2 (Axiom (20) and left distributivity)**.** *Recall that KAT has an axiom of left distributivity* $c \cdot (c_1 + c_2) = c \cdot c_1 + c \cdot c_2$*. It does not hold in GKAT with the guarded sum* $+_b$ *though. In some sense axiom (20) (or its refinement Prop. 4.3) can be seen as a kind of compensation for this lack of left distributivity because it allows when one is reasoning on an expression* $c \cdot (c_1 +_b c_2)$ *(in order to establish a bound* β*) to continue the proof with two branches, respectively on* $c \cdot c_1$ *and on* $c \cdot c_2$ *.*

4.2 Semantic reasoning

When reasoning on examples of programs we want to establish properties on their semantic interpretations. For that, beside the axioms of aGKAT we will also need to use some semantic properties, for instance that some actions can be commuted without changing the semantics of the program. For this we introduce some notations:

Definition 4.3. *Given two GKAT program* c *and* c' *and a probabilistic interpretation* i*, we write* $c \equiv c'$ *if* $\mathcal{P}_i[\![c]\!] = \mathcal{P}_i[\![c']\!]$.

Then we have the following properties:

Proposition 4.4. *Consider GKAT programs* c *and* c'*, and a probabilistic interpretation* i*.*

1. *If* b *is a test which only depends on the values of some variables* $x_1, \ldots, x_n$ *and if* c *leaves the values of those variables unchanged, then we have:*

$$c \cdot b \equiv b \cdot c \tag{24}$$

2. *If* $c \equiv c'$ *and* $c \blacktriangleleft \beta$*, then* $c' \blacktriangleleft \beta$*.*

Observe that property 1 holds because the syntax of programs does not allow any form of aliasing. As to property 2, it holds because the definition of $c \blacktriangleleft \beta$ only depends on the semantic interpretation $\mathcal{P}_i[\![c]\!]$.

Let us now illustrate the use of aGKAT on a small example.

Example 4.1 (Double tossing)**.** *Consider the program c below:*

$$\begin{aligned} c &= (x \xleftarrow{\$} Coin) \cdot (c_1 +_{[x=1]} (y \leftarrow 0)) \\ c_1 &= (x \xleftarrow{\$} Coin) \cdot ((y \leftarrow 1) +_{[x=1]} (y \leftarrow 0)) \end{aligned}$$

Because of the semantics of the distribution Coin we can assume:

$$(x \xleftarrow{\$} Coin) \cdot [x = 1] \blacktriangleleft 1/2, \quad (x \xleftarrow{\$} Coin) \cdot [x \neq 1] \blacktriangleleft 1/2$$

We want to prove that after the execution of c, the probability that y equals 0 is inferior to $3/4$, and the probability that y equals 1 is inferior to $1/4$, that is to say that:

$$c \cdot [y = 0] \blacktriangleleft 3/4 \text{ and } c \cdot [y = 1] \blacktriangleleft 1/4$$

Now, by using Axiom (5), $c \cdot [y = 0]$ can be rewritten as follows:

$$\begin{aligned} c \cdot [y = 0] &= (x \xleftarrow{\$} Coin) \cdot (c_1' +_{[x=1]} (y \leftarrow 0)[y = 0]) \\ c_1' &= (x \xleftarrow{\$} Coin) \cdot ((y \leftarrow 1)[y = 0] +_{[x=1]} (y \leftarrow 0)[y = 0]) \end{aligned}$$

We then enumerate the various possible branches of executions of $c \cdot [y = 0]$:

$$c_2 = (x \xleftarrow{\$} Coin) \cdot [x = 1] \cdot (x \xleftarrow{\$} Coin) \cdot [x = 1] \cdot (y \leftarrow 1) \cdot [y = 0] \tag{25}$$

$$c_3 = (x \xleftarrow{\$} Coin) \cdot [x = 1] \cdot (x \xleftarrow{\$} Coin) \cdot [x \neq 1] \cdot (y \leftarrow 0) \cdot [y = 0] \tag{26}$$

$$c_4 = (x \xleftarrow{\$} Coin) \cdot [x \neq 1] \cdot (y \leftarrow 0) \cdot [y = 0] \tag{27}$$

First, from the model we know that $(y \leftarrow 1) \cdot [y = 0] \equiv 0$, so $c_2 \equiv 0$, so $c_2 \blacktriangleleft 0$.

Then, as $(x \xleftarrow{\$} Coin) \cdot [x \neq 1] \blacktriangleleft 1/2$, by Axioms (21) and (18) we have $c_4 \blacktriangleleft 1/2$.

Then, as $(x \xleftarrow{\$} Coin) \cdot [x = 1] \blacktriangleleft 1/2$, by Axioms (18) and (21) we have $c_3 \blacktriangleleft 1/4$.

By applying Prop. 4.3 to c_2 and c_3 we get:

$$(x \xleftarrow{\$} Coin) \cdot [x = 1] \cdot c_1' \quad \blacktriangleleft \quad 1/4 \tag{28}$$

By applying again Prop. 4.3, this time to (28) and $c_4 \blacktriangleleft 1/2$ we finally obtain:

$$c \cdot [y = 0] \triangleleft 3/4 \; (= 1/4 + 1/2) \tag{29}$$

The proof that $c \cdot [y = 1] \blacktriangleleft 1/4$ holds is similar.

5 Encoding aHL in aGKAT

We want to relate deduction in aHL and reasoning in aGKAT. We follow for that the approach of [13] which shows how to encode propositional Hoare logic in KAT. In this paper a propositional Hoare logic judgement $\vdash c : \phi \Rightarrow \phi'$ is encoded in KAT as $b \cdot c \cdot \overline{b'} = 0$ if the predicates ϕ and ϕ' are equivalent resp. to the boolean tests b and b'.

In the case of aHL, we will encode an aHL judgement $\vdash_\beta c : \phi \Rightarrow \phi'$ by the aGKAT statement $b \cdot c \cdot \overline{b'} \triangleleft \beta$, if the predicates ϕ and ϕ' are equivalent resp. to the boolean tests b and b'.

In this approach, showing that an aHL rule is sound in aGKAT will consist in proving that the conjunction of the aGKAT equations encoding the premises of the aHL rule implies the equation encoding the conclusion of the rule.

Observe that similarly as for Hoare logic, some rules of aHL, namely axiom rules (*Assn*) and (*Rand*), do not depend on aHL judgements as premises but rather on an interpretation of actions and predicates, and possibly a semantic condition (for (*Rand*)). Thus we do not expect to derive their encoding as an equation valid in the theory of aGKAT. Instead, when we will deal with examples we will consider a particular interpretation and thus reason on equalities of expressions in the model.

We now list on Fig. 4 the interpretations of the rules of aHL (Figure 2) in aGKAT, by using the encoding of aHL judgments as aGKAT equations. Recall that the interpretation is valid only when the rule is instantiated with predicates $(\phi, \phi' \ldots)$ in the premises which are equivalent to some GKAT boolean tests. For more readability we will then use the same notation $(\phi, \phi' \ldots)$ for the GKAT boolean

- Skip:

$$\phi 1 \neg\ \phi \lhd 0 \tag{30}$$

- Assn:

$$\phi[t/x](x \leftarrow t) \neg\ \phi \lhd 0 \tag{31}$$

- Rand:

$$\forall\, m \cdot m \models \phi \Rightarrow \mathcal{P}_i[\![x \xleftarrow{\$} d]\!](m)[\neg\ \psi] \leq \beta \Rightarrow \phi(x \xleftarrow{\$} d) \neg\ \psi \lhd \beta \tag{32}$$

- Seq:

$$(\phi c \neg\ \phi' \lhd \beta) \wedge (\phi' c' \neg\ \phi'' \lhd \beta') \Rightarrow \phi c c' \neg\ \phi'' \lhd \beta'' \tag{33}$$

- Cond:

$$(\phi b c \neg\ \psi \lhd \beta) \wedge (\phi \neg\ b c' \neg\ \psi \lhd \beta) \Rightarrow \phi(c +_b c') \neg\ \psi \lhd \beta \tag{34}$$

- Weak:

$$(\phi \leq \phi') \wedge (\phi c \neg\ \psi \lhd \beta) \wedge (\psi' \leq \psi) \wedge (\beta \leq \beta') \Rightarrow \phi' c \neg\ \psi' \lhd \beta' \tag{35}$$

- And:

$$(\phi c \neg\ \psi \lhd \beta) \wedge (\phi c \neg\ \psi' \lhd \beta') \Rightarrow \phi c \neg\ (\psi \psi') \lhd \beta + \beta' \tag{36}$$

- Or:

$$(\phi c \neg\ \psi \lhd \beta) \wedge (\phi' c \neg\ \psi \lhd \beta) \Rightarrow (\phi + \phi') c \neg\ \psi \lhd \beta \tag{37}$$

- False:

$$\phi c \neg\ \bot \lhd 1 \tag{38}$$

- While:

$$(b_v : \mathbb{N} \models \phi) \wedge (b_v \leq 0 \rightarrow \neg\ b) \wedge (\phi c \neg\ \phi \lhd \beta) \wedge (\forall_{\eta > 0} \cdot \phi b[b_v = \eta] c \neg\ [b_v < \eta] \lhd 0) \Rightarrow \phi[b_v \leq k] c^{(b)} \neg\ (\phi \neg\ b) \tag{39}$$

Figure 4: Interpretation of aHL rules in aGKAT

tests as for the predicates. The aHL rule (31) uses a predicate $\phi[t/x]$ obtained by substitution of a term t in the formula ϕ. By abuse of notation we will also write as $\phi[t/x]$ the test e' obtained by evaluating the formula $\phi[t/x]$ in the model. Therefore, as already stressed above, the aGKAT interpretation (31) of this rule depends on the interpretation in the model. Similarly the rule (32) for handling sampling and corresponding to aHL rule (*Rand*) also depends on a semantic condition from the model.

Theorem 5.1. *All the rules of the system aHL, union bound logic, except* (Assn) *and* (Rand)*, i.e. (30)-(39) have an aGKAT interpretation (given in Fig. 4) valid in any aGKAT.*

Proof. We give here the proof for the rule (**Seq**). The cases of the other rules are given in Appendix F.

By assumption we have: $\phi c \neg\, \phi' \lhd \beta$ and $\phi' c' \neg\, \phi'' \lhd \beta'$. Then we have:

$$\begin{aligned} \phi c c' \neg\, \phi'' &= \phi c 1 c' \neg\, \phi'' \\ &= \phi c(\phi' +_{\phi'} \neg\, \phi') c' \neg\, \phi'' \text{ by Prop. 2.1} \\ &= \phi c(\phi' c' \neg\, \phi'' +_{\phi'} \neg\, \phi' c' \neg\, \phi'') \text{ by (5)} \end{aligned}$$

We have $\phi c(\phi' c' \neg\, \phi'') \lhd \beta'$, by axiom (18) because $\phi c \lhd 1$ (axiom 21) and $\phi' c' \neg\, \phi'' \lhd \beta'$.

Additionally, we have $\phi c(\neg\, \phi' c' \neg\, \phi'') = (\phi c \neg\, \phi') c' \neg\, \phi'' \lhd \beta$, by axiom (18) because $\phi c \neg\, \phi' \lhd \beta$ and $c' \neg\, \phi'' \lhd 1$ (axiom 21).

Hence, by axiom (20), $\phi c(\phi' c' \neg\, \phi'' +_{\phi'} \neg\, \phi' c' \neg\, \phi'') \lhd \beta + \beta'$, so $\phi c c' \neg\, \phi'' \lhd \beta + \beta'$, by using axiom (16). □

6 Example

We now consider the example of the *Report-noisy-max* algorithm, which has been analysed in [4] with the logic aHL. Our analysis here using aGKAT will be similar to the previous one in aHL, but the equational approach of aGKAT will simplify some steps. We only sketch the main steps, but the full proof can be found in Appendix G.

We consider a finite set $\mathcal{R}$ and a quality score function *qscore*, which takes as input a pair of an element r of $\mathcal{R}$ and a database d, and returns a real number. The goal of the algorithm is to find an element r^* of $\mathcal{R}$ which approximately minimizes the function *qscore* on d. The algorithm is randomized and only computes an approximate minimization because it is designed to satisfy a differential privacy property (see [6]).

The algorithm proceeds by computing for each element r of $\mathcal{R}$ the quality score $qscore(r, d)$ and adding to it a Laplacian noise (according to the Laplace mechanism for differential privacy [6]) and returning the element r^* with the highest noisy value.

Here we do not deal with the privacy property of this program, but instead our objective is to study its *accuracy*, that is to say to bound the difference between the value of $qscore(r^*, d)$ and the real minimum of $qscore(\cdot, d)$ on $\mathcal{R}$. The algorithm is written in GKAT as program c:

$$\begin{aligned} c &= (\mathit{flag} \leftarrow 1);\ (\mathit{best} \leftarrow 0);\ (\mathcal{R}_0 \leftarrow \mathcal{R});\ (\mathcal{R}' \leftarrow \varnothing);\ c'^{[\mathcal{R} \neq \varnothing]};\ \mathit{return}(r^*) \\ \text{where } c' &= (r \leftarrow \mathit{pick}(\mathcal{R}));\ (\mathit{noisy}[r] \xleftarrow{\$} \mathcal{L}_{\epsilon/2}(\mathit{qscore}(r, d)));\\ &\quad (c_1 +_b 1);\\ &\quad (\mathcal{R} \leftarrow \mathcal{R} \backslash \{r\});\ (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}) \\ c_1 &= (\mathit{flag} \leftarrow 0);\ (r^* \leftarrow r);\ (\mathit{best} \leftarrow \mathit{noisy}[r])) \\ b &= (\mathit{noisy}[r] > \mathit{best}) + (\mathit{flag} == 1) \end{aligned}$$

The variable *flag* has boolean values ($\{0, 1\}$), $\mathcal{R}$, $\mathcal{R}_0$ and $\mathcal{R}'$ are sets, r, r^* range over elements of $\mathcal{R}$, $\mathit{noisy}[r]$ and *best* range over reals. The notation $\mathit{noisy}[r]$ is an array-like notation for representing n variables, where n is the size of the set $\mathcal{R}$.

This program uses the following kinds of actions and tests:

- actions for operations on sets: picking an (arbitrary) element r from a set ($r \leftarrow \mathit{pick}(\mathcal{R})$), removing ($\mathcal{R} \leftarrow \mathcal{R} \backslash \{r\}$)) and adding an element ($\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}$),
- sampling from a Laplacian distribution centered in a with parameter p : ($x \xleftarrow{\$} \mathcal{L}_p(a)$),
- comparison tests: inequalities for reals, equality for boolean value, comparison to empty set for sets $[\mathcal{R} \neq \varnothing]$.

We recall the following accuracy property of the Laplace distribution [4]:

Lemma 6.1. *Assume $\beta \in [0,1]$ and ν be a sample from $\mathcal{L}_p(a)$, then:*
$Pr_{\mathcal{L}_p(a)}[\mid \nu - a \mid > \frac{1}{p}\log(\frac{1}{\beta})] < \beta$.

Therefore the Laplacian distribution satisfies $(x \xleftarrow{\$} \mathcal{L}_p(a))[\mid x - a \mid > \frac{1}{p}\log(\frac{1}{\beta})] \blacktriangleleft \beta$. This gives us for the sampling in program c':

$$(noisy[r] \xleftarrow{\$} \mathcal{L}_{\epsilon/2}(qscore(r,d)))[\mid noisy[r] - qscore(r,d) \mid > \frac{2}{\epsilon}\log(\frac{\mid \mathcal{R}_0 \mid}{\beta})] \triangleleft \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{40}$$

Now we want to establish a property for the whole program c'. Denote as b_1 the test $[\mid noisy[r] - qscore(r,d) \mid \leq \frac{2}{\epsilon}\log(\frac{|\mathcal{R}_0|}{\beta})]$. Observe that b_1 only depends on the values of $noisy[r]$ and $qscore(r,d)$. Moreover $noisy[r]$ and $qscore(r,d)$ are not changed by the last 3 actions of c'. Therefore by applying Prop.4.4.24 we get c'; $\overline{b_1} \equiv c''$, where c'' is obtained by inserting in c' the test $\overline{b_1}$ just after $(noisy[r] \xleftarrow{\$} \mathcal{L}_{\epsilon/2}(qscore(r,d)))$. So we know that:

$$(noisy[r] \xleftarrow{\$} \mathcal{L}_{\epsilon/2}(qscore(r,d)));\ \overline{b_1} \quad \triangleleft \quad \frac{\beta}{\mid \mathcal{R}_0 \mid} \text{ (this is (40))} \tag{41}$$

$$c_0 \quad \triangleleft \quad 1 \text{ for any } c_0, \text{ by axiom (21)} \tag{42}$$

So by applying axiom (18) to (41) and (42) we obtain that $c'' \triangleleft \frac{\beta}{|\mathcal{R}_0|}$. This is a step where aGKAT has provided us a concise and simple reasoning. Therefore as c'; $\overline{b_1} \equiv c''$ we get by Prop. 4.4.2:

$$c' \cdot \overline{b_1} \triangleleft \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{43}$$

We want to prove an invariant for the body c' of the *while* loop in c. For that consider the test b_2 corresponding to the following predicate:

$$\phi_2 = \forall\, r \in \mathcal{R}',\ \mid noisy[r] - qscore(r,d) \mid \leq \frac{2}{\epsilon}\log(\frac{\mid \mathcal{R}_0 \mid}{\beta})$$

Observe that ϕ_2 is equivalent to a finite conjunction ranging over the elements r of $\mathcal{R}'$ of primitive tests, therefore b_2 is a valid test. We have:

$$b_2 \cdot b_1 \cdot (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}) \cdot \overline{b_2} \quad = \quad 0 \tag{44}$$

Let us denote as c_2 the expression c' deprived of the last action, that is to say: $c' = c_2 \cdot (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\})$. The reasoning we did on c' before can be repeated for c_2, and so just as (43) we have: $c_2 \cdot \overline{b_1} \triangleleft \frac{\beta}{|\mathcal{R}_0|}$. Therefore by axioms (18) and (21) we get: $b_2 \cdot c_2 \cdot \overline{b_1} \triangleleft \frac{\beta}{|\mathcal{R}_0|}$ (here again aGKAT helps us with conciseness). Moreover as c_2 does not modify $\mathcal{R}'$, by using Prop.4.4.24 we get: $b_2 \cdot c_2 \cdot \overline{b_2} \triangleleft 0$. Thus by using the aGKAT encoding (Theorem 5.1) of the aHL rule (And) we obtain from the two previous statements

$$b_2 \cdot c_2 \cdot \overline{b_1 \cdot b_2} \triangleleft \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{45}$$

$$\text{Equation (44) gives us:} \quad (b_1 \cdot b_2) \cdot (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}) \cdot \overline{b_2} \quad \triangleleft \quad 0 \tag{46}$$

By using the aGKAT encoding of the aHL rule (Seq) we get from (45) and (46): $b_2 \cdot c' \cdot \overline{b_2} \triangleleft \frac{\beta}{|\mathcal{R}_0|}$. By using the aGKAT encoding of the aHL rule (While) we get from this last statement as the loop runs for $\mid \mathcal{R}_0 \mid$ iterations: $b_2 \cdot c'^{[\mathcal{R} \neq \varnothing]} \cdot \overline{b_2} \triangleleft \beta$.

Finally as $(\mathcal{R}' \leftarrow \varnothing) \cdot \overline{b_2} = 0$ we deduce from this statement using (Seq) that: $c \cdot \overline{b_2} \triangleleft \beta$. So we have proven in aGKAT the encoding of: $\vdash_\beta c : T \Rightarrow \forall\, r \in \mathcal{R}',\ \mid noisy[r] - qscore(r,d) \mid \leq \frac{2}{\epsilon}\log(\frac{|\mathcal{R}_0|}{\beta})$.

In Appendix G we show to combine this result with another invariant to obtain finally an accuracy bound for the algorithm.

7 Related work

A probabilistic extension of GKAT was introduced in [18] for reasoning on imperative programs with probabilistic branching. The main difference to our approach consists on the way the probability is introduced. The authors introduce the probability syntactically, by extending the set of actions with a biased coin flip $\oplus_r$, where r is a probability; we rather do it semantically, by taking samplings as basic programs. They need naturally an additional set of axioms for the introduced operator.

Another extension of KAT was given in [8] which aimed at capturing computations with some notion of uncertainty, namely fuzzy programs. While in the present paper we have deterministic (boolean) tests to reason on deterministic properties on states, in the approach mentioned the Boolean algebra of KAT was replaced by a lattice, with the goal of being able to reason on fuzzy (non Boolean) properties [10].

Concerning GKAT again, [9] introduced a variant of GKAT to reason on properties of pairs of probabilistic programs. Instead of being an approach to deal with unary executions of programs, this framework is rather a relational structure in the style of [3], able to reason, for example, on properties of non-interference between variables of pairs of distinct programs or two executions of the same program.

Some other related works are in the area of Hoare logics. In the present paper we refered to union bound logic [4]. A more general notion of Graded Hoare logic (GHL) was actually introduced in [7], as a parameterisable framework for augmenting Hoare logic with a preordered monoidal analysis. A logic of this framework is defined by a monoidal structure of grades; some examples of such logics are: the union bound logic aHL [4] that we considered here; logics for analysis of computation time; or the logic for reasoning about program counter security [15].

A Hoare style logic for reasoning about differential privacy was introduced in [5], named *Approximate probabilistic relational Hoare logic*. It allows to reason on two probabilistic programs (or two executions of the same program). The term 'approximate' refers here to the parameters associated to reasoning on judgments, which are related to the distance between the probabilistic distributions generated by the probabilistic programs.

Both frameworks could benefit from an algebraic approach, by the introduction of Kleene algebra with tests kind of structures which could encode the deductive apparatus from the logical systems, simplifying it into a a quasi-equational style of reasoning. For the first case, one would want a Kleene algebra with tests parametric on a monoidal structure in order to capture the different examples addressed with Graded HL. Such a structure would be a generalisation of the one presented in this paper, capturing a wider range of situations. The second case calls for a structure taking into account the parametric reasoning on judgments themselves. One would need to embed the parameters into the structure itself, resorting, for example, to a relation between algebraic terms and the elements from the structure which gives corps to the parameters.

8 Discussion and future work

In our approach, we restricted ourselves to a specific pod-monoid with specific interval of values and set of operators, which were enough to capture aHL and handle the initial intended goals. However we would like to push this approach further. By using a generic and external structure to the main algebraic model of programs, we could follow a parametric approach, and obtain more freedom on the structure chosen to capture a wider range of examples: the analysis of computation time model, by taking the natural numbers and the arithmetic sum as the monoidal composition; the program counter security model, by taking a set of binary values and the string concatenation as the composition; and the union bound logic itself. Those are a few concrete models considered for a generic version of Hoare Logic, analysed in [7].

We want to stress however that it is not trivial to capture in the same generic setting both the union bound logic and the logic for analysis of computation time. The system aGKAT does not allow to do that. In particular axiom (21) requires that the neutral element of the first monoid is also maximal for the order; this is not the case in the monoid $(\mathbb{N}, +)$ (or even $(\mathbb{N}^\infty, +)$) used for the Hoare logic for analysis of computation time.

One of the advantages of the syntactical restriction of GKAT is to facilitate the inclusion of probabilistic models, by neglecting nondeterminism. While usually avoided, and always difficult, one possible direction for future work could be to consider a language with both nondeterminism and probabilities, capturing more application scenarios.

References

[1] D. Kozen A. Angus. Kleene algebra with tests and program schematology. Technical report, Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA, July 2001. Technical Report TR2001-1844.

[2] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. Netkat: semantic foundations for networks. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 113–126. ACM, 2014. doi:10.1145/2535838.2535862.

[3] Timos Antonopoulos, Eric Koskinen, Ton Chanh Le, Ramana Nagasamudram, David A. Naumann, and Minh Ngo. An algebra of alignment for relational verification. *CoRR*, abs/2202.04278, 2022. URL: https://arxiv.org/abs/2202.04278, arXiv:2202.04278.

[4] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. A program logic for union bounds. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 107:1–107:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.107.

[5] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Trans. Program. Lang. Syst.*, 35(3):9:1–9:49, 2013. doi:10.1145/2492061.

[6] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014. doi:10.1561/0400000042.

[7] Marco Gaboardi, Shin-ya Katsumata, Dominic Orchard, and Tetsuya Sato. Graded hoare logic and its categorical semantics. In Nobuko Yoshida, editor, *Programming Languages and Systems - 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12648 of *Lecture Notes in Computer Science*, pages 234–263. Springer, 2021. doi:10.1007/978-3-030-72019-3_9.

[8] L. Gomes, A. Madeira, and L. S. Barbosa. Generalising KAT to verify weighted computations. *Scient. Annals of Comp. Sc.*, 29(2):141–184, 2019. doi:10.7561/SACS.2019.2.141.

[9] Leandro Gomes, Patrick Baillot, and Marco Gaboardi. BiGKAT: an algebraic framework for relational verification of probabilistic programs. working paper or preprint, March 2023. URL: https://hal.science/hal-04017128.

[10] Leandro Gomes, Alexandre Madeira, and Luís Soares Barbosa. A semantics and a logic for fuzzy arden syntax. *Soft Comput.*, 25(9):6789–6805, 2021. doi:10.1007/s00500-021-05593-9.

[11] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. URL: https://mitpress.mit.edu/books/introduction-computational-learning-theory.

[12] D. Kozen. Kleene algebra with tests. *ACM Trans. on Prog. Lang. and Systems*, 19(3):427–443, 1997. doi:10.1145/256167.256195.

[13] D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. on Comp. Logic*, 1(212):1–14, 2000. URL: http://dl.acm.org/citation.cfm?id=343378, doi:10.1109/LICS.1999.782610.

[14] Dexter Kozen and Maria-Christina Patron. Certification of compiler optimizations using kleene algebra with tests. In John W. Lloyd, Verónica Dahl, Ulrich Furbach, Manfred Kerber, Kung-Kiu Lau, Catuscia Palamidessi, Luís Moniz Pereira, Yehoshua Sagiv, and Peter J. Stuckey, editors, *Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, volume 1861 of *Lecture Notes in Computer Science*, pages 568–582. Springer, 2000. doi:10.1007/3-540-44957-4_38.

[15] David Molnar, Matt Piotrowski, David Schultz, and David A. Wagner. The program counter security model: Automatic detection and removal of control-flow side channel attacks. In Dongho Won and Seungjoo Kim, editors, *Information Security and Cryptology - ICISC 2005, 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers*, volume 3935 of *Lecture Notes in Computer Science*, pages 156–168. Springer, 2005. doi:10.1007/11734727_14.

[16] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. doi:10.1017/cbo9780511814075.

[17] Damien Pous. Kleene algebra with tests and coq tools for while programs. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 180–196. Springer, 2013. doi:10.1007/978-3-642-39634-2_15.

[18] Wojciech Rozowski, Tobias Kappé, Dexter Kozen, Todd Schmid, and Alexandra Silva. Probabilistic guarded KAT modulo bisimilarity: Completeness and complexity. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 136:1–136:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.136.

[19] 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. *Proc. ACM Program. Lang.*, 4(POPL):61:1–61:28, 2020. doi:10.1145/3371129.

[20] Jana Wagemaker, Nate Foster, Tobias Kappé, Dexter Kozen, Jurriaan Rot, and Alexandra Silva. Concurrent netkat - modeling and analyzing stateful, concurrent networks. In Ilya Sergey, editor, *Programming Languages and Systems - 31st European Symposium on Programming, ESOP 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13240 of *Lecture Notes in Computer Science*, pages 575–602. Springer, 2022. doi:10.1007/978-3-030-99336-8_21.

[21] Cheng Zhang, Arthur Azevedo de Amorim, and Marco Gaboardi. On incorrectness logic and kleene algebra with top and tests. *Proc. ACM Program. Lang.*, 6(POPL):1–30, 2022. doi:10.1145/3498690.

Appendix

A Derivable GKAT facts

$$
\begin{array}{rcll}
c_1 +_{b_1} (c_2 +_{b_2} c_3) & = & (c_1 +_{b_1} c_2) +_{b_1+b_2} c_3 & (47) \\
c_1 +_b c_2 & = & c_1 +_b \neg\, b \cdot c_2 & (48) \\
b_1 \cdot (c_1 +_{b_2} c_2) & = & b_1 c_1 +_{b_2} b_1 c_2 & (49) \\
c +_b 0 & = & bc & (50) \\
c_1 +_0 c_2 & = & c_2 & (51) \\
b \cdot (c_1 +_b c_2) & = & bc_1 & (52)
\end{array}
\qquad
\begin{array}{rcll}
c^{(b)} & = & c^{(b)} \cdot \neg\, b & (53) \\
c^{(b)} & = & (bc)^{(b)} & (54) \\
c^{(0)} & = & 1 & (55) \\
c^{(1)} & = & 0 & (56) \\
b_1^{(b_2)} & = & \neg\, b_2 & (57) \\
c^{(b_2)} & = & c^{(b_1 b_2)} \cdot c^{(b_2)} & (58)
\end{array}
$$

Figure 5: Derivable GKAT facts

B Proof of Proposition 2.1

$$
\begin{array}{ll}
 & b \cdot b_1 + \neg\, b \cdot b_2 \\
= & \quad \{\ (1)\} \\
 & (b \cdot b_1 + \neg\, b \cdot b_2) +_b (b \cdot b_1 + \neg\, b \cdot b_2) \\
= & \quad \{\ (4) \text{ and } (48)\} \\
 & b \cdot (b \cdot b_1 + \neg\, b \cdot b_2) +_b \neg\, b \cdot (b \cdot b_1 + \neg\, b \cdot b_2) \\
= & \quad \{\ 49\} \\
 & (b \cdot b \cdot b_1 + b \cdot \neg\, b \cdot b_2) +_b (\neg\, b \cdot b \cdot b_1 + \neg\, b \cdot \neg\, b \cdot b_2) \\
= & \quad \{\ \text{B.A. and } (7)\} \\
 & (b \cdot b_1 + 0) +_b (0 + \neg\, b \cdot b_2) \\
= & \quad \{\ \text{B.A.}\} \\
 & (b \cdot b_1) +_b (\neg\, b \cdot b_2) \\
= & \quad \{\ (4) \text{ and } (48)\} \\
 & b_1 +_b b_2
\end{array}
$$

C Proof of Proposition 2.2

Because the tests form a boolean algebra we have:

$$b_1 = b_1 + b_1 b_2 \quad \text{(absorption law)}$$

Now we have:

$$
\begin{array}{rcl}
b_1 +_{b_1} b_2 & = & b_1 b_1 + \overline{b_1} b_2 \text{ by Prop. 2.1} \\
 & = & b_1 + \overline{b_1} b_2 \text{ by idempotency} \\
 & = & (b_1 + b_1 b_2) + \overline{b_1} b_2 \text{ by the equation above} \\
 & = & b_1 + (b_1 + \overline{b_1}) b_2 \\
 & = & b_1 + b_2
\end{array}
$$

D Proof of Proposition 4.1

We prove that axioms (16)-(22) are valid in the probabilistic model. Consider a set of states S. Axioms (16) and (17) are trivial.

- Axiom (18):

Assumptions: $c_1 \triangleleft \beta_1 \Leftrightarrow \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') \leq \beta_1$
and $c_2 \triangleleft \beta_2 \Leftrightarrow \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_2]\!](s)(s') \leq \beta_2$.

Goal: $c_1 \cdot c_2 \triangleleft \beta_1 \cdot \beta_2 \Leftrightarrow \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1 \cdot c_2]\!](s)(s') \leq \beta_1 \cdot \beta_2$.

$$
\begin{aligned}
& \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1 \cdot c_2]\!](s)(s') \\
= & \quad \{ \text{ Definition 2.1} \} \\
& \sum_{s'\in S} \sum_{s''\in S} \mathcal{P}_i[\![c_1]\!](s)(s'') \cdot \mathcal{P}_i[\![c_2]\!](s'')(s') \\
= & \quad \{ \text{ commutativity of } + \} \\
& \sum_{s'\in S} (\mathcal{P}_i[\![c_1]\!](s)(s') \cdot \sum_{s''\in S} \mathcal{P}_i[\![c_2]\!](s'')(s')) \\
\leq & \quad \{ \text{ assumptions} \} \\
& \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') \cdot \beta_2 \\
\leq & \quad \{ \text{ assumptions} \} \\
& \beta_1 \cdot \beta_2
\end{aligned}
$$

We then conclude, by Definition 4.2, $c_1 \cdot c_2 \triangleleft \beta_1 \cdot \beta_2$.

- Axiom (19):
  Assumptions: $c_1 \triangleleft \beta \Leftrightarrow \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') \leq \beta$
  and $c_2 \triangleleft \beta \Leftrightarrow \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_2]\!](s)(s') \leq \beta$.

  Goal: $c_1 +_b c_2 \triangleleft \beta \Leftrightarrow \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1 +_b c_2]\!](s)(s') \leq \beta$

$$
\sum_{s'\in S} \mathcal{P}_i[\![c_1 +_b c_2]\!](s)(s') = \begin{cases} \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') & \text{if } s \in sat^{\dagger}(b) \\ \sum_{s'\in S} \mathcal{P}_i[\![c_2]\!](s)(s') & \text{if } s \in sat^{\dagger}(\neg\, b) \end{cases}
$$

  By assumptions, $\forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c_1]\!](s)(s') \leq \beta$ and $\sum_{s'\in S} \mathcal{P}_i[\![c_2]\!](s)(s') \leq \beta$.
  Hence we conclude $\sum_{s'\in State} \mathcal{P}_i[\![c_1 +_b c_2]\!](s)(s') \leq \beta$, i.e. by Definition 4.2 $c_1 +_b c_2 \triangleleft \beta$.

- Axiom (20):
  Assumptions: $cc_1 \triangleleft \beta_1 \Leftrightarrow \forall_{s\in State}, \sum_{s'\in S} \mathcal{P}_i[\![cc_1]\!](s)(s') \leq \beta_1$
  and $cc_2 \triangleleft \beta_2 \Leftrightarrow \forall_{s\in State}, \sum_{s'\in S} \mathcal{P}_i[\![cc_2]\!](s)(s') \leq \beta_2$.

  Goal: $c(c_1 +_b c_2) \triangleleft \beta_1 + \beta_2 \Leftrightarrow \forall_{s\in State}, \sum_{s'\in S} \mathcal{P}_i[\![c(c_1 +_b c_2)]\!](s)(s') \leq \beta_1 + \beta_2$

$$
\begin{aligned}
& \forall_{s\in S}, \sum_{s'\in S} \mathcal{P}_i[\![c(c_1 +_b c_2)]\!](s)(s') \\
= & \quad \{ \text{ Definition 2.1} \} \\
& \sum_{s'\in S} \sum_{s''\in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \mathcal{P}_i[\![c_1 +_b c_2]\!](s'')(s')
\end{aligned}
$$

$$
\begin{aligned}
&= && \{ \text{ commutativity of } + \} \\
&& \sum_{s'' \in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \sum_{s' \in S} \mathcal{P}_i[\![c_1 +_b c_2]\!](s'')(s') \\
&= && \{ \text{ (Definition 2.1)} \} \\
&& \sum_{s'' \in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \Big(\sum_{s' \in sat^\dagger(b)} \mathcal{P}_i[\![c_1]\!](s'')(s') + \sum_{s' \in sat^\dagger(\neg\, b)} \mathcal{P}_i[\![c_2]\!](s'')(s') \Big) \\
&= && \{ \text{ distributivity of } \cdot \text{ over } + \} \\
&& \sum_{s'' \in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \sum_{s' \in sat^\dagger(b)} \mathcal{P}_i[\![c_1]\!](s'')(s') \\
&+ && \sum_{s'' \in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \sum_{s' \in sat^\dagger(\neg\, b)} \mathcal{P}_i[\![c_2]\!](s'')(s') \\
&= && \{ \text{ commutativity of } + \} \\
&& \sum_{s' \in sat^\dagger(b)} \sum_{s'' \in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \mathcal{P}_i[\![c_1]\!](s'')(s') \\
&+ && \sum_{s' \in sat^\dagger(\neg\, b)} \sum_{s'' \in S} \mathcal{P}_i[\![c]\!](s)(s'') \cdot \mathcal{P}_i[\![c_2]\!](s'')(s') \\
&= && \{ \text{ Definition 2.1} \} \\
&& \sum_{s' \in sat^\dagger(b)} \mathcal{P}_i[\![cc_1]\!](s)(s') + \sum_{s' \in sat^\dagger(\neg\, b)} \mathcal{P}_i[\![cc_2]\!](s)(s') \\
&\leq && \{ \text{ assumptions} \} \\
&& \beta_1 + \beta_2
\end{aligned}
$$

Hence, by Definition 4.2, $c(c_1 +_b c_2) \leq \beta_1 + \beta_2$.

The validity of Axiom (21) comes directly from Definition 4.2.

- Axiom (22):
  $\forall_{s \in S}, \sum_{s' \in S} \mathcal{P}_i[\![0]\!](s)(s') = 0 \leq 0$. Hence, by Definition 4.2, $0 \triangleleft 0$.

E Counter-example for Remark 4.1 (Invalid statement in the probabilistic model)

Let us take as set of states $States = \{s_1, s_2\}$. We use a matrix notation for Markov kernels: the matrix $\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$ represents the Markov kernel f such that, for all $i, j \in \{1, 2\}$, $f(s_i)(s_j) = a_{i,j}$. So in particular the sum over each line is inferior or equal to 1. We denote abusively: $f = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$.

Let us define:

$$
c = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}
$$

$$
c_1 = \begin{pmatrix} 1/2 & 1/2 \\ 0 & 0 \end{pmatrix}
$$

$$
c_2 = \begin{pmatrix} 0 & 0 \\ 1/2 & 1/2 \end{pmatrix}
$$

Then we have:

$$
c \cdot c_1 = c \cdot c_2 = \begin{pmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{pmatrix}
$$

So we have: $c \cdot c_1 \triangleleft 1/2$ and $c \cdot c_2 \triangleleft 1/2$, because the sum over each line is equal to $1/2$.

Let us take for b the test on states s defined by $b = [s = s_1]$. Then we have:

$$c_1 +_b c_2 = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

So:

$$c \cdot (c_1 +_b c_2) = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

So $c \cdot (c_1 +_b c_2) \triangleleft 1 (= \frac{1}{2} + \frac{1}{2})$ holds, Axiom (20) is satisfied. And the minimum β which satisfies $c \cdot (c_1 +_b c_2) \triangleleft \beta$ is 1, thus the candidate statement of Remark 4.1 is not valid.

F Proof of Theorem 5.1

- **Skip**:

  Goal: $\phi 1 \neg\, \phi \triangleleft 0$

$$\begin{aligned} & \phi 1 \neg\, \phi \\ = & \quad \{ \text{(9)} \} \\ & \phi \neg\, \phi \\ = & \quad \{ \text{B.A.} \} \\ & 0 \end{aligned}$$

  By axiom (22), $0 \triangleleft 0$.

- **Seq**:

  Assumptions:

$$\phi c \neg\, \phi' \triangleleft \beta \tag{59}$$

  and

$$\phi' c' \neg\, \phi'' \triangleleft \beta' \tag{60}$$

  Goal: $\phi c c' \neg\, \phi'' \triangleleft \beta + \beta'$

$$\begin{aligned} & \phi c c' \neg\, \phi'' \\ = & \quad \{ \text{B.A.} \} \\ & \phi c 1 c' \neg\, \phi'' \\ = & \quad \{ \text{Proposition 2.1 and B.A.} \} \\ & \phi c (\phi' +_{\phi'} \neg\, \phi') c' \neg\, \phi'' \\ = & \quad \{ \text{(5)} \} \\ & \phi c (\phi' c' \neg\, \phi'' +_{\phi'} \neg\, \phi' c' \neg\, \phi'') \end{aligned}$$

  We have $\phi c (\phi' c' \neg\, \phi'') \triangleleft \beta'$, by axiom (18) because $\phi c \triangleleft 1$ (axiom 21) and (60).

  Additionally, we have $\phi c (\neg\, \phi' c' \neg\, \phi'') = (\phi c \neg\, \phi') c' \neg\, \phi'' \triangleleft \beta$, by axiom (18) because (59) and $c' \neg\, \phi'' \triangleleft 1$ (axiom 21).

  Hence, by axiom (20), $\phi c (\phi' c' \neg\, \phi'' +_{\phi'} \neg\, \phi' c' \neg\, \phi'') \triangleleft \beta + \beta'$, so $\phi c c' \neg\, \phi'' \triangleleft \beta + \beta'$.

- **Cond**:

  Assumptions:

$$\phi b c \neg\, \psi \triangleleft \beta \tag{61}$$

and

$$\phi\neg\, bc'\neg\, \psi \lhd \beta \tag{62}$$

Goal: $\phi(c +_b c')\neg\, \psi \lhd \beta$

By axiom (5) and fact (49), $\phi(c +_b c')\neg\, \psi = \phi c\neg\, \psi +_b \phi c'\neg\, \psi$ and by (61), (62) and axiom (19), we have $\phi c\neg\, \psi +_b \phi c'\neg\, \psi \lhd \beta$.

- **Weak**:

  Assumptions:

$$\phi c\neg\, \psi \lhd \beta \tag{63}$$

$$\phi'\phi = \phi' \tag{64}$$

$$\psi\psi' = \psi \tag{65}$$

  and

$$\beta \leq \beta' \tag{66}$$

  Goal: $\phi' c\neg\, \psi' \lhd \beta'$

  By Boolean algebra, we derive

$$\psi\psi' = \psi \Leftrightarrow \neg\, \psi'\neg\, \psi = \neg\, \psi' \tag{67}$$

  Hence, we reason,

$$\begin{aligned} & \phi' c\neg\, \psi' \\ = & \quad \{\ (64) \text{ and } (68)\} \\ & \phi'\phi c\neg\, \psi'\neg\, \psi \\ = & \quad \{\ (\text{B.A.})\} \\ & \phi'(\phi c\neg\, \psi)\neg\, \psi' \end{aligned}$$

  By axiom (21), $\phi' \lhd 1$, $\neg\, \psi' \lhd 1$. So, by (63) and axiom (18), we have $\phi'(\phi c\neg\, \psi)\neg\, \psi' \lhd 1 \cdot \beta \cdot 1 = \beta$. By (66) and axiom (17), we conclude $\phi'(\phi c\neg\, \psi)\neg\, \psi' \lhd \beta'$.

- **And**:

  Assumptions:

$$\phi c\neg\, \psi \lhd \beta \tag{68}$$

  and

$$\phi' c\neg\, \psi' \lhd \beta' \tag{69}$$

  Goal: $\phi c\neg\, (\psi\psi') \lhd \beta + \beta'$

  By Boolean algebra and Proposition 2.1, we reason $\phi c\neg\, (\psi\psi') = \phi c(\neg\, \psi + \neg\, \psi') = \phi c(\neg\, \psi +_{\neg\, \psi} \neg\, \psi')$ and by premises (68) and (69), and by axiom (20) we deduce

  $\phi c(\neg\, \psi +_{\neg\, \psi} \neg\, \psi') \lhd \beta + \beta'$

  and by axiom (16) we have $\phi c\neg\, (\psi\psi') \lhd \beta + \beta'$.

- **Or**:
  Assumptions:
  $$\phi c\neg\ \psi \lhd \beta \tag{70}$$
  and
  $$\phi' c\neg\ \psi \lhd \beta \tag{71}$$
  Goal: $(\phi + \phi')c\neg\ \psi \lhd \beta$

$$\begin{array}{ll} & (\phi + \phi')c\neg\ \psi \\ = & \quad \{\ (4)\} \\ & (1 +_\phi \phi')c\neg\ \psi \\ = & \quad \{\ (5)\} \\ & c\neg\ \psi +_\phi \phi' c\neg\ \psi \\ = & \quad \{\ (4)\} \\ & \phi c\neg\ \psi +_\phi \phi' c\neg\ \psi \end{array}$$

  By premises (70), (71) and axiom (19), $\phi c\neg\ \psi +_\phi \phi' c\neg\ \psi \lhd \beta$.

- **False**:
  $\phi c\neg\ 0 = \phi c1 = \phi c \lhd 1$ by axioms (21), (18) and (16).

- **While**:
  Assumptions:
  $$\phi c\neg\ \phi \lhd \beta \tag{72}$$
  $$\phi b[b_v = \eta]c\neg\ [b_v < \eta] \lhd 0 \tag{73}$$
  $$b_v : \mathbb{N} \models \phi \wedge (b_v \leq 0) \Rightarrow \neg\ e \tag{74}$$
  Goal: $phi[b_v \leq k]c^{(b)}\neg\ (\phi\neg\ b) = \phi[b_v \leq k]c^{(b)}(\neg\ phi + b) = \phi[b_v \leq k]c^{(b)}\neg\ \phi \lhd k\beta$
  We proof by induction on k.
  - $k = 0$:

$$\begin{array}{ll} & \phi[b_v \leq k]c^{(b)}\neg\ (\phi\neg\ b) \\ = & \quad \{\ \text{B.A. and (53)}\} \\ & \phi[b_v \leq k]c^{(b)}\neg\ \phi \\ = & \quad \{\ \text{(11) and (2)}\} \\ & \phi[b_v \leq k](1 +_{\neg\ b}\ cc^{(b)})\neg\ \phi \\ = & \quad \{\ \text{(2) and (4)}\} \\ & \phi[b_v \leq k](1 +_{\neg\ b}\ ecc^{(b)})\neg\ \phi \\ = & \quad \{\ (5)\} \\ & \phi[b_v \leq k](\neg\ \phi +_{\neg\ b}\ ecc^{(b)}\neg\ \phi) \end{array}$$

    We can conclude, by axiom (22), $\phi[b_v \leq k]\neg\ \phi = 0 \lhd 0$, and, by premise (74) and axiom (18), $\phi[b_v \leq k]bcc^{(b)} \lhd 0$.

– k:

$$\begin{aligned}
& \phi[b_v \le k]c^{(b)}\neg\,(\phi\neg\, b) \\
= \quad & \{\text{ B.A. and (53)}\} \\
& \phi[b_v \le k]c^{(b)}\neg\,\phi \\
= \quad & \{\text{ (11)}\} \\
& \phi[b_v \le k](cc^{(b)}+1)\neg\,\phi \\
= \quad & \{\text{ (2)}\} \\
& \phi[b_v \le k](1 +_{\neg\, b}\, cc^{(b)})\neg\,\phi \\
= \quad & \{\text{ definition of } {}^{(b)}\} \\
& \phi[b_v \le k](1 +_{\neg\, b}\, c(1 +_{\neg\, b}\, c(1 +_{\neg\, b} \ldots c(1+_{\neg\, b cc^{(b)}}))))\neg\,\phi
\end{aligned}$$

Repeating k times, by (72) and (Seq) we have

$$\phi[b_v \le k](1 +_{\neg\, b}\, c(1 +_{\neg\, b}\, c(1 +_{\neg\, b} \ldots c(1+_{\neg\, b cc^{(b)}}))))\neg\,\phi \triangleleft k\beta$$

Hence we assume $H(k) : \phi[b_v = k]c^{(b)}\neg\,\phi \triangleleft k\beta$.

– $k+1$:

$$\begin{aligned}
& \phi[b_v = k+1]c^{(b)}\neg\,\phi \\
= \quad & \{\text{ (11), (2) and (4)}\} \\
& \phi[b_v = k+1](1 +_{\neg\, b}\, ecc^{(b)})\neg\,\phi \\
= \quad & \{\text{ (5)}\} \\
& \phi[b_v = k+1](\neg\,\phi +_{\neg\, b}\, ecc^{(b)}\neg\,\phi)
\end{aligned}$$

We observe that $\phi[b_v = k+1]\neg\,\phi = 0 \triangleleft 0$ by B.A. and axiom (22).

G Example: Report-noisy-max algorithm

We will now consider the example of the *Report-noisy-max* algorithm, which has been analysed in [4] with the logic aHL (see also [6] for more background on this algorithm). Our analysis here using aGKAT will be similar to the previous one in aHL, but the equational approach of aGKAT will simplify some steps.

We consider a finite set $\mathcal{R}$ and a quality score function *qscore*, which takes as input a pair of an element r of $\mathcal{R}$ and a database d, and returns a real number. The goal of the Report-noisy-max algorithm is to find an element r^* of $\mathcal{R}$ which approximately minimizes the function *qscore* on d. The algorithm is randomized and only computes an approximate minimization because it is designed to satisfy a differential privacy property (see [6]).

Report-noisy-max proceeds by computing for each element r of $\mathcal{R}$ the quality score $qscore(r, d)$ and adding to it a Laplacian noise (according to the Laplace mechanism for differential privacy [6]) and returning the element r^* with the highest noisy value.

Here we do not deal with the differential privacy property of this program, but instead our objective is to study its *accuracy*, that is to say to bound the difference between the value of $qscore(r^*, d)$ and the real minimum of $qscore(\cdot, d)$ on $\mathcal{R}$.

The algorithm Report-noisy-max can be written as a GKAT program c as follows (we use intermediary notations c', c_1 and b for readability):

$$c \quad = \quad (\mathit{flag} \leftarrow 1);\ (\mathit{best} \leftarrow 0);\ (\mathcal{R}_0 \leftarrow \mathcal{R});\ (\mathcal{R}' \leftarrow \varnothing);\ c'^{[\mathcal{R} \neq \varnothing]};\ \mathit{return}(r^*)$$

where

$$
\begin{array}{rcl}
c' & = & (r \leftarrow pick(\mathcal{R}));\ (noisy[r] \stackrel{\$}{\leftarrow} \mathcal{L}_{\epsilon/2}(qscore(r,d)));\\
 & & (c_1 +_b 1);\\
 & & (\mathcal{R} \leftarrow \mathcal{R}\backslash\{r\});\ (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\})\\
c_1 & = & (flag \leftarrow 0);\ (r^* \leftarrow r);\ (best \leftarrow noisy[r]))\\
b & = & (noisy[r] > best) + (flag == 1)
\end{array}
$$

The variable *flag* has boolean values ($\{0,1\}$), $\mathcal{R}$, $\mathcal{R}_0$ and $\mathcal{R}'$ are sets, r, r^* range over elements of $\mathcal{R}$, $noisy[r]$ and *best* range over reals. The notation $noisy[r]$ is an array-like notation for representing n variables, where n is the size of the set $\mathcal{R}$.

This program uses the following kinds of actions and tests:

- actions representing basic operations on sets: picking an (arbitrary) element r from a set ($r \leftarrow pick(\mathcal{R})$), removing an element ($\mathcal{R} \leftarrow \mathcal{R}\backslash\{r\}$)) and adding an element ($\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}$),
- sampling from a Laplacian distribution centered in a with parameter p : ($x \stackrel{\$}{\leftarrow} \mathcal{L}_p(a)$),
- comparison tests: inequalities for reals, equality for boolean value, comparison to empty set for sets $[\mathcal{R} \neq \varnothing]$.

We recall the following accuracy property of the Laplace distribution [4]:

Lemma G.1. *Assume β belongs to $[0,1]$ and let ν be a sample from the Laplace distribution $\mathcal{L}_p(a)$, then we have:*

$$Pr_{\mathcal{L}_p(a)}[\mid \nu - a \mid > \frac{1}{p}\log(\frac{1}{\beta})] < \beta \tag{75}$$

Therefore the Laplacian distribution satisfies the following property:

$$(x \stackrel{\$}{\leftarrow} \mathcal{L}_p(a))[\mid x - a \mid > \frac{1}{p}\log(\frac{1}{\beta})] \blacktriangleleft \beta \tag{76}$$

This corresponds to the following instance of the (Rand) axiom in aHL (see Fig. 2):

$$\frac{\forall\, m, \mathcal{P}_i[\![x \stackrel{\$}{\leftarrow} \mathcal{L}_p(a)]\!](m)[\mid x - a \mid > \frac{1}{p}\log(\frac{1}{\beta})] \leq \beta}{\vdash_\beta \mathbf{do}\ .(x \stackrel{\$}{\leftarrow} \mathcal{L}_p(a)) : T \Rightarrow \mid x - a \mid \leq \frac{1}{p}\log(\frac{1}{\beta})}$$

Now, (76) gives us for the sampling in program c':

$$(noisy[r] \stackrel{\$}{\leftarrow} \mathcal{L}_{\epsilon/2}(qscore(r,d)))[\mid noisy[r] - qscore(r,d) \mid > \frac{2}{\epsilon}\log(\frac{\mid \mathcal{R}_0 \mid}{\beta})] \triangleleft \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{77}$$

We want to establish a property for the whole program c'.

Denote as b_1 the test $[\mid noisy[r] - qscore(r,d) \mid \leq \frac{2}{\epsilon}\log(\frac{|\mathcal{R}_0|}{\beta})]$. Observe that b_1 (and thus also $\overline{b_1}$) only depends on the values of $noisy[r]$ and $qscore(r,d)$. Moreover $noisy[r]$ and $qscore(r,d)$ are not changed by the last 3 actions of c'. Therefore by applying Prop.4.4.24 we get c'; $\overline{b_1} \equiv c''$, where c'' is the expression obtained by replacing in c' $(noisy[r] \stackrel{\$}{\leftarrow} \mathcal{L}_{\epsilon/2}(qscore(r,d)))$ by $(noisy[r] \stackrel{\$}{\leftarrow} \mathcal{L}_{\epsilon/2}(qscore(r,d)))$; $\overline{b_1}$.

So we know that:

$$(noisy[r] \stackrel{\$}{\leftarrow} \mathcal{L}_{\epsilon/2}(qscore(r,d)));\ \overline{b_1} \quad \triangleleft \quad \frac{\beta}{\mid \mathcal{R}_0 \mid} \text{ (this is (77))} \tag{78}$$

$$c_0 \quad \triangleleft \quad 1 \text{ for any } c_0, \text{ by axiom (21)} \tag{79}$$

So by applying axiom (18) to (78) and (79) we obtain that $c'' \triangleleft \frac{\beta}{|\mathcal{R}_0|}$. Therefore as c'; $\overline{b_1} \equiv c''$ we get by Prop. 4.4.2 that:

$$c' \cdot \overline{b_1} \triangleleft \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{80}$$

We want to prove an invariant for the body c' of the *while* loop in c. For that consider the test b_2 corresponding to the following predicate:

$$\phi_2 = \forall\, r \in \mathcal{R}',\ \mid noisy[r] - qscore(r,d) \mid \leq \frac{2}{\epsilon}\log(\frac{\mid \mathcal{R}_0 \mid}{\beta})$$

Observe that ϕ_2 is equivalent to a finite conjunction ranging over the elements r of $\mathcal{R}'$ of primitive tests, therefore b_2 is a valid test. We have:

$$b_2 \cdot b_1 \cdot (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}) \cdot \overline{b_2} \quad = \quad 0 \tag{81}$$

Let us denote as c_2 the expression c' deprived of the last action, that is to say: $c' = c_2 \cdot (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\})$. The reasoning we did on c' before can be done for c_2, and so just as (80) we have:

$$c_2 \cdot \overline{b_1} \lhd \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{82}$$

Therefore by axioms (18) and (21) we get:

$$b_2 \cdot c_2 \cdot \overline{b_1} \lhd \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{83}$$

Moreover as c_2 does not modify $\mathcal{R}'$, by using Prop.4.4.24 we get:

$$b_2 \cdot c_2 \cdot \overline{b_2} \lhd 0 \tag{84}$$

Thus by using the aGKAT encoding (Theorem 5.1) of the aHL rule (And) we obtain from (83) and (84):

$$b_2 \cdot c_2 \cdot \overline{b_1 \cdot b_2} \lhd \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{85}$$

Equation (81) gives us:

$$(b_1 \cdot b_2) \cdot (\mathcal{R}' \leftarrow \mathcal{R}' \cup \{r\}) \cdot \overline{b_2} \quad \lhd \quad 0 \tag{86}$$

By using the aGKAT encoding of the aHL rule (Seq) we get from (85) and (86):

$$b_2 \cdot c' \cdot \overline{b_2} \lhd \frac{\beta}{\mid \mathcal{R}_0 \mid} \tag{87}$$

By using the aGKAT encoding of the aHL rule (While) we get from (87), as the loop runs for $\mid \mathcal{R}_0 \mid$ iterations:

$$b_2 \cdot c'^{[\mathcal{R} \neq \varnothing]} \cdot \overline{b_2} \lhd \beta \tag{88}$$

Finally as $(\mathcal{R}' \leftarrow \varnothing) \cdot \overline{b_2} = 0$ we deduce from (88) using (Seq) that:

$$c \cdot \overline{b_2} \lhd \beta \tag{89}$$

So we have proven in aGKAT the encoding of:

$$\vdash_\beta c : T \Rightarrow \forall\, r \in \mathcal{R}',\ \mid noisy[r] - qscore(r, d) \mid \leq \frac{2}{\epsilon} \log(\frac{\mid \mathcal{R}_0 \mid}{\beta}) \tag{90}$$

In order to obtain a property relating $qscore(r^*, d)$ to the values of $qscore(r, d)$ in order to have an accuracy result, we need to relate $noisy[r^*]$ to the $noisy[r]$. For that we will consider the following predicate:

$$\phi_3 = \forall\, r \in \mathcal{R}',\ (noisy[r^*] \geq noisy[r]) \wedge (best = noisy[r^*]) \tag{91}$$

Denote as b_3 the corresponding test. We have:

$$b_3 \cdot c' \cdot \overline{b_3} \quad \lhd \quad 0 \tag{92}$$

$$b_3 \cdot c'^{[\mathcal{R} \neq \varnothing]} \cdot \overline{b_3} \quad \lhd \quad 0 \text{ by (While) rule} \tag{93}$$

$$c \cdot \overline{b_3} \quad \lhd \quad 0 \text{ because } (\mathcal{R}' \leftarrow \varnothing) \cdot \overline{b_3} = 0 \tag{94}$$

Then from (89) and (94) we get with rule (And):

$$c \cdot \overline{b_2 \cdot b_3} \quad \lhd \quad \beta \tag{95}$$

By arithmetical reasoning we have that $\phi_2 \wedge \phi_3$ implies that for any r in $\mathcal{R}'$ we have:

$$qscore(r^*, d) \quad > \quad qscore(r, d) - \frac{2}{\epsilon} \log(\frac{|\ \mathcal{R}_0\ |}{\beta}) \tag{96}$$

Therefore if b_4 denotes the test corresponding to

$$\phi_4 = \forall\, r \in \mathcal{R}',\ qscore(r^*, d) > qscore(r, d) - \frac{2}{\epsilon} \log(\frac{|\ \mathcal{R}_0\ |}{\beta}) \tag{97}$$

we have:

$$b_2 \cdot b_3 = (b_2 \cdot b_3) \cdot b_4 \tag{98}$$

Therefore from (95) and (98) we get by rule (Weak):

$$c \cdot \overline{b_4} \quad \triangleleft \quad \beta \tag{99}$$

So we have proven in aGKAT the encoding of:

$$\vdash_\beta c : T \Rightarrow \forall\, r \in \mathcal{R}',\ qscore(r^*, d) > qscore(r, d) - \frac{2}{\epsilon} \log(\frac{|\ \mathcal{R}_0\ |}{\beta}) \tag{100}$$

This shows an accuracy property for c: with failure probability β, the result r^* of c gives a quality score which is not far below the quality score of any other element r of $\mathcal{R}'$ (that is to say $\mathcal{R}_0$).