# BiGKAT: an algebraic framework for relational verification of probabilistic programs

Leandro Gomes, Patrick Baillot, Marco Gaboardi

# BiGKAT: an algebraic framework for relational verification of probabilistic programs

Leandro Gomes[1], Patrick Baillot[1], and Marco Gaboardi[2]

[1]Univ. Lille, CNRS, Inria, Centrale Lille, UMR9189 CRIStAL, F-59000 Lille, France
[2]Boston University, USA

**Abstract**

This work is devoted to formal reasoning on relational properties of probabilistic imperative programs. Relational properties are properties which relate the execution of two programs (possibly the same one) on two initial memories. We aim at extending the algebraic approach of Kleene Algebras with Tests (KAT) to relational properties of probabilistic programs. For that we consider the approach of Guarded Kleene Algebras with Tests (KAT), which can be used for representing probabilistic programs, and define a relational version of it, called Bi-guarded Kleene Algebras with Tests (BiGKAT). We show that the setting of BiGKAT is expressive enough to interprete probabilistic Relational Hoare Logic (pRHL), a program logic that has been introduced in the literature for the verification of relational properties of probabilistic programs.

**keywords**: Kleene algebra with tests, relational reasoning, probabilistic programs, Hoare logic

## 1 Introduction

Formal verification of program properties has triggered a variety of methods, among which the algebraic approach of Kleene Algebras with Tests (KAT) stands out as an elegant, simple and automatizable framework [11, 9]. It is closely related to modeling with finite automata and has stimulated the development of techniques from coalgebra for reasoning about program behavior, for instance based on bisimulation checking [8]. Among the properties one might want to check on programs, some important ones are expressed by relating the execution of two programs on two initial states, or of the same program on two initial states. They are called *relational properties* or *2-properties*. One can think for instance of simulation properties, refinements, extensional equivalence. Another example is that of non-interference: assume the variables are divided into public ones and private ones, a program satisfies non-interference if the final

value of public variables after an execution only depends on the initial value of public variables (and not on private ones).

Actually in a large number of situations the software systems one wants to verify are not deterministic but admit a probabilistic behaviour. Think for instance of randomized algorithms, cryptography, security, network programming or differential privacy where one uses random noise to ensure the protection of private informations. Here also many crucial properties are relational ones. For instance in cryptography one can express the fact that a randomized encryption scheme is safe by a probabilistic non-interference property: a public variable is assigned a ciphered value, obtained from a private variable, and we want to ensure that one cannot distinguish between two ciphered values computed from the same private initial state. Similarly in differential privacy (see e.g. [4]), in order to protect private data one might want to verify that two executions of a given program on two data-bases that differ only by one individual give indistinguishable result.

In order to express and prove relational properties on imperative programs some specific methods have been introduced. First in the deterministic case let us mention Relational Hoare Logic [7], that extends the classic Floyd-Hoare logic approach to reason on pairs of programs. This approach has been upgraded to the setting of probabilistic relational Hoare Logic (pRHL) by Barthe and coauthors [5]. It has then been extensively applied to the verification of cryptographic schemes, in particular through the development of the Certicrypt [6] and Easycrypt [2] tools.

However it would still be useful to benefit from additional techniques which could help for the automation and the understandability of such reasoning methods. In particular one difficulty with (probabilistic) relational Hoare Logic is to find a suitable alignment of the two programs in order to be able, in a second step, to find the intermediate properties needed for the proof (see [1]). Algebraic methods coming from Kleene algebras with tests are promising in these respects. In particular they facilitate reasoning on simple program transformations.

Our goal in this paper is thus to introduce a KAT approach to reason on relational properties of probabilistic programs. An important step has already been made in the non-probabilistic setting with the introduction of BiKAT [1]. This setting allows to apply the KAT approach to reasoning on pairs of programs. Unfortunately standard KAT techniques cannot be applied directly to probabilistic programs. On this question however a recent progress is the introduction of Guarded Kleene Algebras with tests (GKAT) [13]. In this setting non-deterministic union and iteration are replaced by guarded union and iteration. One initial motivation of the authors with GKAT was to design a more efficient version of KAT where the complexity of the equational theory is reduced, but they also showed that GKAT admits a probabilistic model and can be used to interpret probabilistic programs.

Our strategy is thus to adapt the relational extension BiKAT of KAT to the setting of GKAT, in order to be able to apply this relational approach to pairs of probabilistic programs.

The goal will be to apply such framework to probabilistic programs which

could be expressed in the imperative programming language defined in Table 1.

| | |
|---|---|
| $t ::= \mathrm{k} \mid x \mid \mathrm{f}(t_1, \ldots, t_n)$ | functional terms |
| $p ::= \mathrm{p}(t_1, \ldots, t_n) \mid \neg\, p \mid p_1 \wedge p_2 \mid p_1 \vee p_2$ | predicate terms |
| $c ::= \mathbf{skip} \mid x \leftarrow t \mid x \overset{\$}{\leftarrow} d \mid c \cdot c \mid \mathbf{if}\ p\ \mathbf{then}\ c\ \mathbf{else}\ c$ | compound programs |
| $\mathbf{while}\ p\ \mathbf{do}\ c$ | |

Table 1: Syntax of $\mathcal{PI}$

where:

- $x \in X$ are *variables*;
- $\mathrm{f} \in F$ are *function symbols*. $(F_n)_{n\in\mathbb{N}_0} \subseteq F$ denotes sets of function symbols with arity $n$. Symbols $k \in F_0$ are called constants. Function symbols are interpreted in $F$ as $f : \mathbb{Z}^n \to \mathbb{Z}$ (e.g. $+$, $\surd$);
- $\mathrm{p} \in \mathrm{P}$ are *predicate symbols*. $(\mathrm{P}_n)_{n\in\mathbb{N}_0} \subseteq \mathrm{P}$ denotes sets of predicate symbols with arity $n$. Predicate symbols are interpreted in P as $\mathrm{p} : \mathbb{Z}^n \to \{0, 1\}$ (e.g. $=$, $\geq$);
- $d$ are sub-distributions on $\mathbb{Z}$.

Additionally, notation $T(X)$ stands for the set of terms with variables in $X$, and $T_F(X)$ (respectively, $T_P(X)$) represents its restriction to functional (respectively, predicate) terms.

In order to demonstrate the expressivity of our framework we want to show how probabilistic relational Hoare Logic reasoning can be interpreted in it, in a similar way as (standard) Hoare logic can be interpreted in KAT [10]. This will raise some specific difficulties, in particular for proving the validity of the rule dealing with the *while* construct. Finally we will illustrate the benefits of our setting on some examples.

**Outline of the paper**. In Sect. 2 we introduce GKAT and its probabilistic model, then define the variant we consider, including an additional theory for assignments and probabilistic sampling. Then in Sect. 3 we introduce the relational extension BiGKAT of GKAT, define the interpretation of pRHL judgments in BiGKAT and prove our main theorem, the soundness of this interpretation. Sect. 4 is then devoted to the study of several examples.

# 2 Guarded Kleene algebra with tests

This section recalls the language and the semantics of *guarded Kleene algebra with tests (GKAT)*, an abstraction of imperative programs, with conditionals $(c_1 +_e c_2)$ and loops $(c^{(e)})$ are guarded by Boolean predicates $e$. As explained in the introduction, it presents 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 (over PSPACE of KAT), and better foundation for probabilistic applications (not necessarily to deal with nondeterminism and probabilities in the same language). Although the first one was the main motivation to introduce the structure [13], we are more interested in the second advantage for this paper.

## 2.1 Syntax

The language of GKAT, that we will present below, encodes the probabilistic programming language $\mathcal{PI}$ (1). Consider a set of actions $\Sigma$ and predicates $P$, where $\Sigma$ and $P$ are nonempty and disjoint. Elements of $\Sigma$ encode either assignments $x \leftarrow t$ or samplings $x \overset{\$}{\leftarrow} d$, and elements of $P$, denoted as $\rho$, encode predicate terms p ∈ P. The grammar of an arbitrary Boolean expression and GKAT expression are constructed, respectively, as follows:

$$e, e_1, e_2 \in \text{BExp} ::= 0 \mid 1 \mid \rho \mid \neg\ e \mid e_1 \cdot e_2 \mid e_1 + e_2$$

$$c, c_1, c_2 \in \text{Exp} ::= a \mid e \mid c_1 \cdot c_2 \mid c_1 +_e c_2 \mid c^{(e)}$$

where, for any $e, e_1, e_2 \in$ BExp, operators $\cdot$, $+$ and $\neg$ denote conjunction, disjunction and negation, respectively, and, for any $c, c_1, c_2 \in$ Exp, operator $\cdot$ denotes sequential composition. The Boolean expression 1, by being also an element of Exp, encodes command **skip**, and the conditional and iteration imperative programming constructs can be abbreviated as GKAT terms as follows:

$$c_1 +_e c_2 \overset{\text{def}}{\equiv} \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2$$
$$c^{(e)} \overset{\text{def}}{\equiv} \textbf{while } e \textbf{ do } c$$

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 $c_1, c_2 \in$ Exp.

## 2.2 Semantics

We present now the semantic interpretation of GKAT that we will be using to interpret GKAT expressions, the *Probabilistic model* [13]. Note that more interpretations of GKAT are presented in the literature, namely a relational model and a language model [13]. We revise first some basic concepts needed for the semantics we present next. Given the set of relative integers $\mathbb{Z}$, $\mathcal{D}(\mathbb{Z})$ is the set of sub-distributions over $\mathbb{Z}$, i.e. the set of functions $f : \mathbb{Z} \to [0, 1]$ summing up to at most 1, i.e. $\sum_{z \in \mathbb{Z}} f(z) \leq 1$. In particular, the *Dirac* distribution $\delta_z \in \mathcal{D}(\mathbb{Z})$ is the map $w \to [w = z] = \begin{cases} 1, \text{ if w=z} \\ 0, \text{ otherwise} \end{cases}$

Typical models of imperative programming languages interpret programs as Markov kernels. This semantic model interprets programs as sub-Markov kernels, i.e. Markov kernels over probability sub-distributions.

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

- $\mathit{State}$ is a set of states,
- $\mathit{eval}(a) \subseteq \mathit{State} \to \mathcal{D}(\mathit{State})$ is a sub-Markov kernel, for each action $a \in \Sigma$,
- $\mathit{sat}(\rho) \subseteq \mathit{State}$ is a set of states, for each predicate $\rho \in P$.

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

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

where $\mathrm{sat}^\dagger : \mathrm{BExp} \to 2^{\mathrm{State}}$ is the lifting of $\mathrm{sat} : P \to 2^{\mathrm{State}}$ to an arbitrary Boolean expression over $P$.
The interpretation of actions $a \in \Sigma$ as sub-Markov Kernels is given as:

- $\mathit{eval}(x \leftarrow t)(\sigma) := \delta_{\sigma[x \leftarrow t]}$
- $\mathit{eval}(x \overset{\$}{\leftarrow} d)(\sigma) := \sum_t d(f(t)) \cdot \delta_{\sigma[x \leftarrow t]}$

Additionally, to fully interpret programs written in $\mathcal{PI}$ we need the measure monad $M(X)$ whose constructor is defined as

$$M(X) \overset{\text{def}}{=} (X \to [0,1]) \to [0,1]$$

and operators are defined as

$$\texttt{unit} : X \to M(X) \overset{\text{def}}{=} \lambda\, x.\, \lambda f. f\ x$$
$$\texttt{bind} : M(X) \to (X \to M(Y)) \to M(Y) \overset{\text{def}}{=} \lambda\, d.\, \lambda\, M.\, \lambda f. d(\lambda\, x. M\ x\ f)$$

$$
\begin{array}{rcll}
c +_e c & = & c & (1)\\
c_1 +_e c_2 & = & c_2 +_{\neg\, e} c_1 & (2)\\
(e +_b f) +_c g & = & e +_{bc} (f +_c g) & (3)\\
c_1 +_e c_2 & = & e c_1 +_e c_2 & (4)\\
c_1 c_3 +_e c_2 c_3 & = & (c_1 +_e c_2) \cdot c_3 & (5)\\
(c_1 \cdot c_2) \cdot c_3 & = & c_1 \cdot (c_2 \cdot c_3) & (6)\\
0 \cdot c & = & 0 & (7)\\
c \cdot 0 & = & 0 & (8)\\
1 \cdot c & = & c & (9)\\
c \cdot 1 & = & c & (10)\\
c^{(e)} & = & c \cdot c^{(e)} +_e 1 & (11)\\
(c +_{e_2} 1)^{(e_1)} & = & (e_2 \cdot c)^{(e_1)} & (12)\\
\dfrac{c_3 = c_1 \cdot c_3 +_e c_2}{c_3 = c_1^{(e)} \cdot c_2} & \text{if} & E(c_1) = 0 & (13)
\end{array}
$$

Table 2: Axiomatisation of Guarded Kleene algebra with tests

## 2.3 Axioms

The list of axioms presented in Table 2 is the one obtained from [13].

Note in particular for the fixpoint axiom (13). Intuitively, it says that if expression $c_3$ chooses (using guard $e$) between executing $c_1$ and loop again, and execute $c_2$, then $c_3$ is a $e$-guarded loop followed by $c_2$. However, the rule is not sound in general (see [13]) for more details. in order to overcome such limitation, the side condition $E(c_1) = 0$ is introduced, assuring that command $c_1$ is productive, i.e. that it performs some action. To this end, the function $E$ is inductively defined as follows:

$$
\begin{array}{l}
E(e) := e\\
E(c) := 0\\
E(c_1 +_e c_2) := e \cdot E(c_1) + \neg\, e \cdot E(c_2)\\
E(c_1 \cdot c_2) := E(c_1) \cdot E(c_2)\\
E(c^{(e)}) := \neg\, e
\end{array}
$$

We can see $E(c)$ as the weakest test that guarantees that command $c$ terminates, but nevertheless it does not perform any action.

Moreover, note particularly the following observation: the encoding $c_1;\ (e;\ c_2 + \neg\, e;\ c_3) = c_1;\ e;\ c_2 + c_1;\ \neg\, e;\ c_3$ is not an if then else; it is rather a nondeterministic choice between executing $c_1$, then testing $e$ and executing $c_2$, and

executing $c_1$, then testing $\neg\ e$ and executing $c_3$. That is why left distributivity does not hold in GKAT for any $c \in \text{Exp}$; only holds for $e \in \text{BExp}$, i.e. if $e$ is a test.

In Table 3 we list additional derivable equations in GKAT, also obtained from [13].

$$
\begin{aligned}
c_1 +_{e_1} (c_2 +_{e_2} c_3) &= (c_1 +_{e_1} c_2) +_{e_1+e_2} c_3 && (14)\\
c_1 +_e c_2 &= c_1 +_e \neg\ e \cdot c_2 && (15)\\
e_1 \cdot (c_1 +_{e_2} c_2) &= e_1 \cdot c_1 +_{e_2} e_1 \cdot c_2 && (16)\\
c +_e 0 &= e \cdot c && (17)\\
c_1 +_0 c_2 &= c_2 && (18)\\
e \cdot (c_1 +_e c_2) &= ec_1 && (19)\\
c^{(e)} &= c^{(e)} \cdot \neg\ e && (20)\\
c^{(e)} &= (e \cdot c)^{(e)} && (21)\\
c^{(0)} &= 1 && (22)\\
c^{(1)} &= 0 && (23)\\
e_1^{(e_2)} &= \neg\ e_2 && (24)\\
c^{(e_2)} &= c^{(e_1\, e_2)} \cdot c^{(e_2)} && (25)
\end{aligned}
$$

Table 3: Derivable GKAT facts

## 2.4 Equational theory for effects

This section presents an equational theory of GKAT which includes additional axioms to deal with the effects of assignments and samplings in the course of execution of a program in language $\mathcal{PI}$. In any GKAT, in general two actions $a_1, a_2 \in \Sigma$ are not commutable, however they can commute for the particular case in which they don't share variables. Those facts will be useful to deal with examples later in the paper. Hence, the equational theory of GKAT that we will resort on ads to the base theory the following additional axioms:

$$
x_1 \leftarrow t_1 \cdot x_2 \leftarrow t_2 = \begin{cases} x_2 \leftarrow t_2[t_1/x_1] \cdot x_1 \leftarrow t_1 \ \text{ if } x_1 \neq x_2 \text{ and } x_2 \notin FV(t_1) \\ x_1 \leftarrow t_2[t_1/x_1] \ \text{ if } x_1 = x_2 \end{cases} \quad (26)
$$

$$
x_1 \overset{\$}{\leftarrow} d_1 \cdot x_2 \overset{\$}{\leftarrow} d_2 = \begin{cases} x_2 \overset{\$}{\leftarrow} d_2 \cdot x_1 \overset{\$}{\leftarrow} d_1 \ \text{ if } x_1 \neq x_2 \\ x_1 \overset{\$}{\leftarrow} d_2 \ \text{ if } x_1 = x_2 \end{cases} \quad (27)
$$

$$
x_1 \leftarrow t \cdot x_2 \overset{\$}{\leftarrow} d = \begin{cases} x_2 \overset{\$}{\leftarrow} d \cdot x_1 \leftarrow t \ \text{ if } x_1 \neq x_2 \\ x_1 \overset{\$}{\leftarrow} d \ \text{ if } x_1 = x_2 \end{cases} \quad (28)
$$

$$
x_1 \overset{\$}{\leftarrow} d \cdot x_2 \leftarrow t = x_1 \leftarrow t \quad \text{if } x_1 = x_2 \quad (29)
$$

**Proposition 2.2.** *The axioms (26)-(29) are valid in the Probabilistic model of Definition 2.1.*

Proof. Proof in appendix. □

We already mentioned that GKAT does not allow to construct an arbitrary program by using freely the nondeterministic choice operator $+$, allowing only guarded choice $+_e$, for any $e \in \mathrm{BExp}$. However, the $+$ operator is included in the grammar of BExp, representing the Boolean disjunction. Nevertheless, the grammar also allows to write expressions as $e +_e e$, for any $e \in \mathrm{BExp}$. We add the additional axiom

$$e +_e \neg\, e = e \cdot e + \neg\, e \neg\, e \tag{30}$$

to the theory of GTAK which expresses the guarded sum $+_e$, for any $e \in \mathrm{BExp}$, in terms of the disjunction $+$ on tests. By Boolean reasoning, we can easily observe that $e \cdot e + \neg\, e \neg\, e = 1$. Such property will be useful later in the paper.

# 3 Bi-guarded Kleene algebra with tests

To handle relational reasoning on probabilistic programs, we introduce in this section *Bi-guarded Kleene algebra with tests*, an algebraic structure inspired by Bi Kleene algebra with tests [1], defined over a GKAT.

**Definition 3.1.** A *Bi guarded Kleene algebra with tests (BiGKAT)* over a GKAT
$(\mathrm{A}, \mathrm{B}, +_e, \cdot, ^{(e)}, \neg, +, 1, 0)$ is a GKAT

$$(\ddot{\mathrm{A}}, \ddot{\mathrm{B}}, \oplus_E, \fatsemi, ^{(E)}, \ddot{\neg}, \ddot{\oplus}, \ddot{1}, \ddot{0})$$

such that $E \in \ddot{\mathrm{B}}, \ddot{\mathrm{B}} \subseteq \ddot{\mathrm{A}}$, the operator $\oplus$ is applied only to elements of $B$, and $\langle\ |\ : \mathrm{A} \to \ddot{\mathrm{A}},\ |\ \rangle : \mathrm{A} \to \ddot{\mathrm{A}}$ are homomorphisms satisfying

$$\forall\, c_1, c_2 \in \mathrm{A},\ \langle c_1| \fatsemi |c_2\rangle = |c_2\rangle \fatsemi \langle c_1|; \tag{31}$$

We call $\ddot{\mathrm{A}}$ the underlying GKAT, and elements of $\ddot{\mathrm{B}}$ are called bi-tests.

We define notation $\langle \_|\_\rangle$ as $\langle c|c'\rangle \stackrel{\text{def}}{=} \langle c| \fatsemi |c'\rangle$, with the following consequences: $\langle c|1\rangle = \langle c|$ and $\langle 1|c\rangle = |c\rangle$ since $|1\rangle = \ddot{1}$ is the identity of $\fatsemi$. Another property that arrives naturally from the definition of $\langle \_|\_\rangle$ is $\langle 0|c\rangle = 0 = \langle c|0\rangle$, for any $c \in A$.

The fact that $\langle\ |$ is an homomophism means that the following properties hold for any $e_1,\ e_2,\ e \in B,\ c_1,\ c_2,\ c \in A$:

- $\langle e_1 + e_2| = \langle e_1| \oplus \langle e_2|$,
- $\langle c_1 \cdot c_2| = \langle c_1| \fatsemi \langle c_2|$;
- $\langle c_1 +_e c_2| = \langle c_1| \oplus_E \langle c_2|$;

- $\langle c^{(e)}| = \langle c|^E$.

where $E$ stands for $\langle e| \in \ddot{\mathrm{B}}$. Similarly for $|\,\rangle$. The operators have the same precedence as in ordered-GKAT. For readability we use interchangeably the same notation for operators in GKAT and BiGKAT, i.e. operators $\cdot$, $\neg$ and $+_e$, for any $e \in B$, and for constants 1, 0, in GKAT stand also for $\fatsemi$, $\overline{\neg}$, $\oplus_{\langle e|}$ ( $\oplus_{|e\rangle}$), $\ddot{1}$ and $\ddot{0}$ , respectively.

## 3.1 Encoding pRHL in BiGKAT

In this section we want to prove that *probabilistic relational Hoare logic (pRHL)* [5] can be soundly encoded in BiGKAT. For that let us briefly recall *probabilistic relational Hoare logic (pRHL)*. It can be understood as an extension of Benton's Relational Hoare Logic [7] to probabilistic programs. In Relational Hoare Logic a judgement has the form:

$$\vdash c \sim c' : \phi \Rightarrow \psi$$

where $c$, $c'$ are deterministic programs and $\phi$, $\psi$ (resp. pre- and postcondition) are relations on states. It means that for any memories $m_1$, $m_2$ such that $m_1 \phi m_2$, if the evaluation of $c$ on $m_1$ and $c'$ on $m_2$ terminate with memories $m_1'$ and $m_2'$, then $m_1' \psi m_2'$ holds.

Now, in the probabilistic case the evaluation of a program on a memory gives a subdistribution. In order to extend Relational Hoare Logic, the system pRHL thus lifts relations over memories to relations over distributions. For that the lifting to subdistributions of a unary predicate $P$ and of a binary relation $\phi$ are defined as follows:

$$\begin{aligned} \texttt{range}\ P\ d &\stackrel{\text{def}}{=} \forall f.(\forall\, a.\ P\ a = 0 \Rightarrow f\ a = 0) \Rightarrow d\ f = 0) \\ d_1 \sim_\psi d_2 &\stackrel{\text{def}}{=} \exists\, d.\ \pi_1(d) = d_1\ \wedge \pi_2(d) = d_2 \wedge \texttt{range}\ \psi\ d \end{aligned}$$

where the projections $\pi_1(d)$ and $\pi_2(d)$ are defined by:

$$\pi_1(d) \stackrel{\text{def}}{=} \texttt{bind}\ d\ (\lambda(x,y).\texttt{unit}\ x)\ ,\ \pi_2(d) \stackrel{\text{def}}{=} \texttt{bind}\ d\ (\lambda(x,y).\texttt{unit}\ y).$$

Now that these definitions have been set we can describe the judgements in pRHL.

**Definition 3.2.** Given two probabilistic programs $c$, $c'$ and $\phi$, $\psi$ relations on states, the pRHL judgement $\vdash c \sim c' : \phi \Rightarrow \psi$ stands for the following property:

$$\forall\, m_1, m_2,\ m_1\ \phi\ m_2 \Rightarrow [\![c]\!] m_1 \sim_\psi [\![c']\!] m_2.$$

We say in this case that programs $c$ and $c'$ are equivalent with respect to precondition $\phi$ and postcondition $\psi$.

Following this interpretation, we encode such judgment in BiGKAT as the equation

$$\varphi \cdot \langle c|c'\rangle = \varphi \cdot \langle c|c'\rangle \cdot \psi \tag{32}$$

- *R-Assign rule*:

$$\frac{}{x \leftarrow v \sim x' \leftarrow v' : \varphi[v/x, v'/x'] \Rightarrow \varphi}$$

- *R-Assign left rule*:

$$\frac{}{x \leftarrow v \sim \mathit{skip} : \varphi[v/x] \Rightarrow \varphi}$$

- *R-Rand assign rule*:

$$\frac{h \lhd (d, d')}{x \overset{\$}{\leftarrow} d \sim x' \overset{\$}{\leftarrow} d' : \varphi \Rightarrow \psi}$$

- *R-Seq rule*:

$$\frac{c_1 \sim c_1' : \phi \Rightarrow \psi \quad c_2 \sim c_2' : \psi \Rightarrow \xi}{c_1 \cdot c_2 \sim c_1' \cdot c_2' : \phi \Rightarrow \xi}$$

- *R-Cond rule*:

$$\frac{\phi \Rightarrow e \ddot{=} e' \;\; c_1 \sim c_1' : \phi \wedge \langle e| \wedge |e'\rangle \Rightarrow \psi \;\; c_2 \sim c_2' : \phi \wedge \langle \neg\, e| \wedge |\neg\, e'\rangle \Rightarrow \psi}{\textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \;\sim \textbf{if } e' \textbf{ then } c_1' \textbf{ else } c_2' \;: \phi \Rightarrow \psi}$$

- *R-Cond left rule*:

$$\frac{c_1 \sim c_1' : \phi \wedge \langle e| \Rightarrow \psi \;\; c_2 \sim c_1' : \phi \wedge \langle \neg\, e| \Rightarrow \psi}{\textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \;\sim c_1' : \phi \Rightarrow \psi}$$

- *R-Cond right rule*:

$$\frac{c_1 \sim c_1' : \phi \wedge |e\rangle \Rightarrow \psi \;\; c_1 \sim c_2' : \phi \wedge |\neg\, e'\rangle \Rightarrow \psi}{c_1 \;\sim \textbf{if } e' \textbf{ then } c_1' \textbf{ else } c_2' \;: \phi \Rightarrow \psi}$$

- *Weak R-Whl rule*:

$$\frac{\phi \Rightarrow e \ddot{=} e' \;\; c \sim c' : \phi \wedge \langle e| \wedge |e'\rangle \Rightarrow \phi \;\; E(c) = E(c') = 0}{\phi \textbf{ while } e \textbf{ do } c \sim \textbf{while } e' \textbf{ do } c' : \phi \wedge \langle \neg\, e| \wedge |\neg\, e'\rangle \Rightarrow \phi}$$

- *R-Sub rule*:

$$\frac{\phi' \Rightarrow \phi \;\; c \sim c' : \phi \Rightarrow \psi \;\; \psi \Rightarrow \psi'}{c \sim c' : \phi' \Rightarrow \psi'}$$

- *R-Case rule*:

$$\frac{c \sim c' : \phi \wedge \phi' \Rightarrow \psi \;\; c \sim c' : \phi \wedge \neg\phi' \Rightarrow \psi}{c \sim c' : \phi \Rightarrow \psi}$$

Figure 1: Probabilistic Relational Hoare Logic rules (pRHL)

where $\varphi, \psi \in \ddot{\mathrm{B}}$, and $c, c' \in \ddot{\mathrm{A}}$. Let us make a few comments to compare this encoding to other ones in the literature:

- Note that we do not use the encoding as $\varphi \cdot \langle c|c'\rangle \leq \varphi \cdot \langle c|c'\rangle \cdot \psi$ because in GKAT and BiGKAT there is no natural notion of order $\leq$ as in KAT [11, 9] ;
- We do not use either the encoding as $\varphi \cdot \langle c|c'\rangle \cdot \neg\, \psi = 0$ as in BiKAT [1]. In KAT, $\varphi \cdot c = \varphi \cdot c \cdot \psi$ is equivalent to $\varphi \cdot c \cdot \neg\, \psi = 0$, but this cannot be proved in the same way in GKAT and we suspect the equivalence does not hold. We only have the implication: $(\varphi \cdot c = \varphi \cdot c \cdot \psi) \Rightarrow (\varphi \cdot c \cdot \neg\, \psi = 0)$, and we choose as encoding the stronger property.

It is necessary also to establish a set of Hoare logic rules that make sense for reasoning on probabilistic programs. We consider as such rules the ones of pHRL [5], listed in Figure 1. We use use different notation for pre and post conditions ($\varphi$, $\psi$) and for guards $\langle e|$, $|e\rangle$. Note in particular the side condition $\phi \Rightarrow e \doteq e'$ in rules *R-Cond* and *R-Whl*, where the right-hand side $e \doteq e'$ is equivalent to $\langle e|e'\rangle + \langle \neg\, e|\neg\, e'\rangle$ so the following holds

$$\phi\langle e|\neg\, e'\rangle = 0 \quad \phi\langle \neg\, e|e'\rangle = 0 \tag{33}$$

These equalities assure that the predicates $e$ and $e'$ are evaluated to the same value on both left and right programs. In particular, for the *R-Cond* rule it means that the same branch is executed for right-hand side and left-hand side programs. One difference from rules in [5] is the additional condition in *R-Whl* rule: in our case, we impose that two commands $c$, $c'$ are guaranteed to perform some action, which we will use to prove the soundness of *R-Whl*. Note also for the *R-Assign*, *R-Assign left* and *R-Rand* which are axioms: the first derives a valid Hoare triple with the substitution of variables $x, x'$ by expressions $v$, $v'$, respectively; the second just derives an assignment on the left-hand side, while the right-hand side is a **skip** instruction; the third derives a valid triple with samplings over distributions $d, d'$. The coupling function $h : \mathrm{supp}\{d\} \to \mathrm{supp}\{d'\}$ is essential to relate the two samplings over distributions $d, d'$, and must satisfy the following conditions:

- $h$ is bijective;
- for every $i \in \mathrm{supp}\{d\}$, $h(i) \in \mathrm{supp}\{d'\}$;
- $P_{x\sim d}[x = i] = P_{x\sim d'}[x = h(i)]$

If such a function exists, i.e. there exists a coupling between distributions $d$, $d'$, we write $h \lhd (d, d')$. For more details on coupling see reference [5].

Observe that the *R-Whl rule* of Figure 1 is actually weaker than the classical one from the literature, as it has a premise requiring that $E(c) = E(c') = 0$. We added this condition because we need it for the proof of soundness of the encoding in BiGKAT (Theorem 3.9). More precisely we use this condition in the

proof of the intermediary Lemma 3.7. This *R-Whl rule* is actually expressive enough for many examples.

Now, to show that the rules of Figure 1 are sound in BiGKAT, we interprete them as follows:

- *R-Assign rule*:

$$\varphi[v/x, v'/x']\langle x \leftarrow v | x' \leftarrow v' \rangle = \varphi[v/x, v'/x']\langle x \leftarrow v | x' \leftarrow v' \rangle \varphi \tag{34}$$

- *R-Assign left rule*:

$$\varphi[v/x]\langle x \leftarrow v | \mathbf{skip} \rangle = \varphi[v/x]\langle x \leftarrow v | \mathbf{skip} \rangle \varphi \tag{35}$$

- *R-Rand assign rule*:

$$\varphi\langle x \overset{\$}{\leftarrow} d | x' \overset{\$}{\leftarrow} d' \rangle = \varphi\langle x \overset{\$}{\leftarrow} d | x' \overset{\$}{\leftarrow} d' \rangle \psi \tag{36}$$

- *R-Seq rule*:

$$\begin{array}{l} \phi\langle c_1 | c_1' \rangle = \phi\langle c_1 | c_1' \rangle \psi \quad \wedge \quad \psi\langle c_2 | c_2' \rangle = \psi\langle c_2 | c_2' \rangle \xi \\ \Rightarrow \quad \phi\langle c_1 \cdot c_2 | c_1' \cdot c_2' \rangle = \phi\langle c_1 \cdot c_2 | c_1' \cdot c_2' \rangle \xi \end{array} \tag{37}$$

- *R-Cond rule*:

$$\begin{array}{l} \phi \leqq e \doteq e' \ \wedge \ \phi \cdot \langle e | \cdot | e' \rangle \cdot \langle c_1 | c_1' \rangle = \phi \cdot \langle e | \cdot | e' \rangle \cdot \langle c_2 | c_2' \rangle \cdot \psi \quad \wedge \\ \phi \cdot \ \langle \neg \ e | \fatsemi | \neg \ e' \rangle \cdot \langle e | e' \rangle = \phi \cdot \ \langle \neg \ e | \cdot \ | \neg \ e' \rangle \cdot \langle c_1 | c_1' \rangle \cdot \psi \\ \Rightarrow \ \phi \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2' \rangle = \phi \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2' \rangle \cdot \psi \end{array} \tag{38}$$

- *R-Cond-left rule*:

$$\begin{array}{l} \phi \cdot \langle e | \cdot \langle c_1 | c_1' \rangle = \phi \cdot \langle e | \cdot \langle c_1 | c_1' \rangle \cdot \psi \quad \wedge \quad \phi \cdot \ \langle \neg \ e | \cdot \langle c_2 | c_1' \rangle = \phi \cdot \ \langle \neg \ e | \cdot \ \langle c_2 | c_1' \rangle \cdot \psi \\ \Rightarrow \ \phi \cdot \langle c_1 +_e c_2 | c_1 \rangle = \phi \cdot \langle c_1 +_e c_2 | c_1' \rangle \cdot \psi \end{array} \tag{39}$$

- *R-Cond-right rule*:

$$\begin{array}{l} \phi \cdot | e \rangle \cdot \langle c_1 | c_1' \rangle = \phi \cdot | e \rangle \cdot \langle c_1 | c_1' \rangle \cdot \psi \quad \wedge \quad \phi \cdot \ | \neg \ e \rangle \cdot \langle c_1 | c_2' \rangle = \phi \cdot \ | \neg \ e \rangle \cdot \ \langle c_1 | c_2' \rangle \cdot \psi \\ \Rightarrow \ \phi \cdot \langle c_1 | c_1 +_e c_2 \rangle = \phi \cdot \langle c_1 | c_1 +_e c_2 \rangle \cdot \psi \end{array} \tag{40}$$

- *R-Whl rule*: we can apply it only if $E(c) = E(c') = 0$,

$$\begin{array}{l} \phi \ \leq \ e \doteq e' \ \wedge \ \phi \cdot \langle e | \cdot | e' \rangle \langle c | c' \rangle = \phi \cdot \langle e | \cdot | e' \rangle \langle c | c' \rangle \cdot \phi \ \wedge \ E(c) = E(c') = 0 \\ \Rightarrow \phi \cdot \langle c^{(e)} | e'^{(e')} \rangle = \phi \cdot \langle c^{(e)} | c'^{(e')} \rangle \cdot \phi \end{array} \tag{41}$$

- *R-Sub rule*:

$$\phi' \leq \phi \quad \wedge \quad \phi\langle c | c' \rangle = \phi\langle c | c' \rangle \psi \quad \wedge \quad \psi \leq \psi' \quad \Rightarrow \quad \phi'\langle c | c' \rangle = \phi'\langle c | c' \rangle \psi' \tag{42}$$

- *R-Case rule*:

$$\begin{aligned}&\phi \cdot \phi' \cdot \langle c|c'\rangle = \phi \cdot \phi' \cdot \langle c|c'\rangle\psi \quad \wedge \phi \cdot \neg\, \phi' \cdot \langle c|c'\rangle = \phi \cdot \neg\, \phi' \cdot \langle c|c'\rangle\psi \\ &\Rightarrow \phi \cdot \langle c|c'\rangle = \phi \cdot \langle c|c'\rangle\psi \end{aligned} \tag{43}$$

To prove some of the rules, namely *R-Cond* and *R-Whl*, we need to establish some auxiliary results.

**Lemma 3.3.** *In any BiGKAT the following two equalities hold:*

$$\langle e| \cdot \langle c_1|c_1'\rangle = \langle e| \cdot \langle c_1 +_e c_2|c_1'\rangle \tag{44}$$

*and*

$$\langle \neg\, e| \cdot \langle c_2|c_2'\rangle = \langle e| \cdot \langle c_1 +_e c_2|c_2'\rangle \tag{45}$$

Proof. Proof in appendix. □

**Lemma 3.4.** *For any BiGKAT,*

$$\phi \cdot \langle e +_e \neg\, e|e' +'_e \neg\, e'\rangle = \phi \cdot (\langle e|e'\rangle +_{e'} \langle \neg\, e|\neg\, e'\rangle) \tag{46}$$

Proof. Proof in appendix. □

Now we state the invariance result, adapted from the standard result on KAT and the equivalent for GKAT, which was proved in [13]. It will be useful for the While rule.

**Lemma 3.5** (Invariance)**.** *Let $c, c' \in A$ and $\phi, e, e' \in \ddot{B}$. If*

$$\phi e\langle c|c'\rangle = \phi e\langle c|c'\rangle\phi$$

*then*

$$\phi c^{(e)} = (\phi c)^{(e)}\phi$$

Proof. Since a BiGKAT is a GKAT (Definition 3.1), it holds by the invariance lemma (Lemma 3.11) of GKAT [13]. □

Now we establish a GKAT property that will be used in the proofs ahead.

**Proposition 3.6.** *For any $e, c$ in GKAT, $ecc^{(e)} = ec^{(e)}$.*

Proof. Proof in appendix. □

The following result is useful for reasoning about two while loops. Based on an analogous property defined for BiKAT [1], we state the corresponding one for BiGKAT:

**Lemma 3.7** (Expansion)**.** *The following property holds in any BiGKAT. Assume $E(c) = E(c') = 0$, then we have:*

$$\langle c^{(e)} | c'^{(e')} \rangle = \langle c|c' \rangle^{\langle e|e' \rangle} (\langle c|\neg\ e' \rangle^{\langle e|} +_{\langle e|} \langle \neg\ e|c' \rangle^{|e' \rangle}) \tag{47}$$

Proof.

$$\begin{aligned}
& \langle c^{(e)} | c'^{(e')} \rangle \\
= & \quad \{ \text{ (11)} \} \\
& \langle cc^{(e)} +_{\langle e|} 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ } \langle \text{ } | \text{ is homomorphism} \} \\
& (\langle cc^{(e)}| +_{\langle e|} \langle 1|)(|c'^{(e')}\rangle) \\
= & \quad \{ \text{ (5)} \} \\
& \langle cc^{(e)} | c'^{(e')} \rangle +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (31)} \} \\
& (|c'^{(e')}\rangle \cdot \langle cc^{(e)}|) +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (11)} \} \\
& (|c'c'^{(e')}\rangle +_{|e' \rangle} |1\rangle)\langle cc^{(e)}| +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (5)} \} \\
& (\langle cc^{(e)} | c'c'^{(e')} \rangle +_{|e' \rangle} \langle cc^{(e)} | 1 \rangle) +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (3)} \} \\
& \langle cc^{(e)} | c'c'^{(e')} \rangle +_{\langle e|e' \rangle} (\langle cc^{(e)} | 1 \rangle +_{\langle e|} \langle 1 | c'^{(e')} \rangle) \\
= & \quad \{ \text{ homomorphism} \} \\
& \langle c|c' \rangle \langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e' \rangle} (\langle cc^{(e)} | 1 \rangle +_{\langle e|} \langle 1 | c'^{(e')} \rangle) \\
= & \quad \{ \text{ (4) and (2)} \} \\
& \langle c|c' \rangle \langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e' \rangle} (\langle ecc^{(e)} | 1 \rangle +_{\langle e|} \langle \neg\ e | c'^{(e')} \rangle) \\
= & \quad \{ \text{ Lemma 3.6 and (4)} \} \\
& \langle c|c' \rangle \langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e' \rangle} (\langle c^{(e)} | 1 \rangle +_{\langle e|} \langle \neg\ e | c'^{(e')} \rangle) \\
= & \quad \{ \text{ (2) and (4)} \} \\
& \langle c|c' \rangle \langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e' \rangle} (\langle \neg\ e| + |\neg\ e' \rangle)(\langle c^{(e)} | 1 \rangle +_{\langle e|} \langle \neg\ e | c'^{(e')} \rangle) \\
= & \quad \{ \text{ fact u5'} \}
\end{aligned}$$

$$
\begin{aligned}
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} ((\langle\neg\ e| + |\neg\ e'\rangle)\langle c^{(e)}|1\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ (4)\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} (\langle e|(\langle\neg\ e| + |\neg\ e'\rangle)\langle c^{(e)}|1\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ \text{B.A.}\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} ((\langle e\cdot\neg\ e| + \langle e|\neg\ e'\rangle)\langle c^{(e)}|1\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ \text{B.A.}\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} ((0 + \langle e|\neg\ e'\rangle)\langle c^{(e)}|1\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ (4)\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} (|\neg\ e'\rangle\langle c^{(e)}|1\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ \text{homomorphism}\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} (\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ (2),\ (4),\ \text{(fact u5')}\ \text{and B.A.}\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} ((\langle\neg\ e| + |\neg\ e'\rangle)\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} (\langle\neg\ e| + |\neg\ e'\rangle)\langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ \text{(fact u5')}\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} (\langle\neg\ e| + |\neg\ e'\rangle)(\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle) \\
= & \quad \{\ (2)\ \text{and}\ (4)\} \\
& \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} (\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle)
\end{aligned}
$$

By the fixpoint axiom (13), considering $g = \langle c^{(e)}|c'^{(e')}\rangle$, $e = \langle c|c'\rangle$, $b = \langle e|e'\rangle$ and $f = \langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle$, we conclude

$$
\begin{aligned}
& \langle c^{(e)}|c'^{(e')}\rangle = \langle c|c'\rangle\langle c^{(e)}|c'^{(e')}\rangle +_{\langle e|e'\rangle} (\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle) \\
\Rightarrow\ & \langle c^{(e)}|c'^{(e')}\rangle = \langle c|c'\rangle^{(\langle e|e'\rangle)}(\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle)
\end{aligned}
$$

which proves (47).

□

The intuitive meaning of this equation is that executing two while loops in parallel ($c^{(e)}$ and $c'^{(e')}$) is equal to loop $c$ and $c'$ guarded by $\langle e|e'\rangle$, assuring that if one of them stops i.e. either $e$ or $e'$ is false, the other loop continues to execute (until its guard is also false). Note that our proof of Lemma 3.7 differs from the proof of the analog lemma in BiKAT [1].

**Lemma 3.8.** *In any BiGKAT, if* $\phi \leq e \doteq e'$ *then we have:*

$$
\phi(\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle) = \langle\neg\ e|\neg\ e'\rangle\phi \tag{48}
$$

Proof. Proof in appendix. □

Now we present the main result on the soundness of pRHL rules in BiGKAT.

**Theorem 3.9** (Soundness of pRHL in BiGKAT)**.** *The rules (37)-(43) are sound in any BiGKAT.*

Proof.

We present here the proofs for *R-Cond* and *R-Whl* and include the remaining ones in the Appendix.

*R-Cond rule*:

$$
\begin{aligned}
& \phi \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle \\
= \quad & \{ \text{ B.A. and (48)} \} \\
& \phi \cdot (\langle e | e' \rangle +_{e'} \langle \neg\, e | \neg\, e' \rangle) \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle \\
= \quad & \{ \text{ (U5,U5')} \} \\
& \phi \cdot \langle e | e' \rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle +_{e'} \phi \cdot \langle \neg\, e | \neg\, e' \rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle \\
= \quad & \{ \text{ (44), (45)} \} \\
& \phi \cdot \langle e | e' \rangle \cdot \langle c_1 | c_1' \rangle +_{e'} \phi \cdot \langle \neg\, e | \neg\, e' \rangle \cdot \langle c_2 | c_2' \rangle \\
= \quad & \{ \text{ premises} \} \\
& \phi \cdot \langle e | e' \rangle \cdot \langle c_1 | c_1' \rangle \cdot \psi +_{e'} \phi \cdot \langle \neg\, e | \neg\, e' \rangle \cdot \langle c_2 | c_2' \rangle \cdot \psi \\
= \quad & \{ \text{ (44), (45)} \} \\
& \phi \cdot \langle e | e' \rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle \cdot \psi +_{e'} \phi \cdot \langle \neg\, e | \neg\, e' \rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle \cdot \psi \\
= \quad & \{ \text{ (U5,U5')} \} \\
& \phi \cdot (\langle e | e' \rangle +_{e'} \langle \neg\, e | \neg\, e' \rangle) \cdot (\langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle) \cdot \psi \\
= \quad & \{ \text{ (48)} \} \\
& \phi \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2 \rangle \cdot \psi
\end{aligned}
$$

*R-Whl rule*:

$$
\begin{aligned}
& \phi \langle c^{(e)} | c'^{e'} \rangle \\
= \quad & \{ \text{ Lemma 3.7 (47)} \} \\
& \phi \langle c | c' \rangle^{\langle e | e' \rangle} (\langle c^{(\langle e |)} | \neg\, e' \rangle +_{\langle e |} \langle \neg\, e | c'^{(| e' \rangle)} \rangle) \\
= \quad & \{ \text{ premise and Lemma 3.5} \} \\
& \phi \langle c | c' \rangle^{\langle e | e' \rangle} \phi (\langle c^{(\langle e |)} | \neg\, e' \rangle +_{\langle e |} \langle \neg\, e | c'^{(| e' \rangle)} \rangle) \\
= \quad & \{ \text{ Lemma 3.8 (48)} \} \\
& \phi \langle c | c' \rangle^{\langle e | e' \rangle} \langle \neg\, e | \neg\, e' \rangle \phi \\
= \quad & \{ \text{ B.A.} \}
\end{aligned}
$$

$$
\begin{aligned}
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}\langle\neg\ e|\neg\ e'\rangle\phi\phi \\
= \quad & \{\ \text{Lemma 3.8 (48) reverse direction}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}\phi(\langle c^{(\langle e|)}|\neg\ e'\rangle +_{\langle e|}\langle\neg\ e|c'^{(|e'\rangle)}\rangle)\phi \\
= \quad & \{\ \text{Lemma 3.5 (reverse direction)}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}(\langle c^{(\langle e|)}|\neg\ e'\rangle +_{\langle e|}\langle\neg\ e|c'^{(|e'\rangle)}\rangle)\phi \\
= \quad & \{\ \text{Lemma 3.7 (47) reverse direction}\} \\
& \phi\langle c^{(e)}|c'^{(e')}\rangle\phi \\
= \quad & \{\ \text{(fact w4)}\} \\
& \phi\langle c^{(e)}\neg\ e|c'^{(e')}\neg\ e'\rangle\phi \\
= \quad & \{\ \text{Def. 3.1}\} \\
& \phi\langle c^{(e)}|c'^{(e')}\rangle\langle\neg\ e|\neg\ e'\rangle\phi
\end{aligned}
$$

□

# 4 Examples

In this section we use the framework presented before to reason about invariance features of probabilistic programs. We take two executions of one program containing random assignments, which produces a probabilistic distribution of states. That means that two executions may lead to different outputs, due to the random nature of the assignments. In the following examples we prove the invariance of certain variables of probabilistic programs in the output relatively to the input, by relational reasoning on two executions of those programs.

**Example 4.1.** *Consider the following program:*

```
var x : mybool;
var y : mybool;
var b : mybool;
if (x = tt) {
     b <-$ dmybool;
     if (b = tt) {
          y <- y xor tt;
     }
}
else {
     b <- ff;
}
     y <- y xor b;
```

*Abbreviate the above program as $c$, and one copy of it as $c'$. We prove the invariance of variables $y, y'$, relational predicate $[y = y']$, over executions of $c, c'$, which corresponds to the following pRHL judgment:*

$$\vdash c \sim c' : [y = y'] \Rightarrow [y = y'] \tag{49}$$

*We translate this judgment to the BiGKAT equation:*

$$[y = y']\langle c|c'\rangle = [y = y']\langle c|c'\rangle[y = y'] \tag{50}$$

*To prove this equation, we encode the above program as the BiGKAT term*

$$\big(b \overset{\$}{\leftarrow} dbool;\ ((y \leftarrow y\ xor\ tt) +_{[b=tt]} 1) +_{[x=tt]} (b \leftarrow ff)\big) \cdot (y \leftarrow y\ xor\ b) \tag{51}$$

*In order to simplify the writing we denote $d_1 = b \overset{\$}{\leftarrow} dbool;\ (y \leftarrow y\ xor\ tt)$, $d_2 = b \leftarrow ff$ and $c_2 = (y \leftarrow y\ xor\ b)$.*

*Thus we prove*

$$\begin{aligned}
& [y = y']\langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x'=tt]} d_2') \cdot c_2' \rangle \\
= \quad & \{\ e +_e \neg\, e = 1\} \\
& [y = y'](\,[x = x'] + \neg\,[x = x'])\langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x'=tt]} d_2') \cdot c_2' \rangle \\
= \quad & \{\ (16)\} \\
& [y = y'][x = x']\langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x=tt]} d_2') \cdot c_2' \rangle \\
= \quad & \{\ (5)\} \\
& [y = y'][x = x']\langle (d_1 \cdot c_2) +_{[x=tt]} (d_2;\ c_2) | (d_1' \cdot c_2') +_{[x=tt]} (d_2' \cdot c_2') \rangle
\end{aligned}$$

*which we subdivide into 4 subgoals, depending on the evaluation of $[x = tt]$ and $[x' = tt]$:*

1. $[x = tt][x' = tt]$
2. $[x \neq tt][x' = tt]$
3. $[x = tt][x \neq tt]$
4. $[x \neq tt][x' \neq tt]$

- *subgoal (1): To prove this subgoal, we introduce a coupling in order to apply the* R-Rand *rule, to assure the invariance of variable $b$ in the sampling $b \overset{\$}{\leftarrow} dmybool$. For this example, we chose as coupling the function $h$, defined such that $b = h(b)$. Hence we use rule (*R-Rand*) in BiGKAT to obtain*

$$
\begin{aligned}
&[y = y']\langle b \xleftarrow{\$} dmybool | b' \xleftarrow{\$} dmybool' \rangle \\
=&[y = y']\langle b \xleftarrow{\$} dmybool | b' \xleftarrow{\$} dmybool' \rangle [y = y'][b = b']
\end{aligned}
$$

*rule* R-Assign *to obtain*

$$
[y = y']\langle y \leftarrow y \; xor \; tt | y' \leftarrow y' \; xor \; tt \rangle = [y = y']\langle y' \leftarrow y' \; xor \; tt | y' \leftarrow y' \; xor \; tt \rangle [y = y']
$$

*which, by 'adding'* $[b = tt][b' = tt]$ *on both sides yields,*

$$
\begin{aligned}
&[y = y'][b = tt][b' = tt]\langle y \leftarrow y \; xor \; tt | y' \leftarrow y' \; xor \; tt \rangle \\
=&[y = y'][b = tt][b' = tt]\langle y \leftarrow y \; xor \; tt | y' \leftarrow y' \; xor \; tt \rangle [y = y']
\end{aligned}
$$

*to form the premise of the conditional rule (40).*
*By rule (38) and the equations above we obtain*

$$
\begin{aligned}
&[y = y'][b = tt][b' = tt]\langle (y \leftarrow y \; xor \; tt) +_{[b=tt]} 1 | (y' \leftarrow y' \; xor \; tt) +_{[b'=tt]} 1 \rangle \\
=&[y = y'][b = tt][b' = tt]\langle (y \leftarrow y \; xor \; tt) +_{[b=tt]} 1 | (y' \leftarrow y' \; xor \; tt) +_{[b'=tt]} 1 \rangle [y = y']
\end{aligned}
$$

*and finally for* $\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle$ *we reason with* R-Rand *to obtain*

$$
\begin{aligned}
&[y = y'][b = b']\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle \\
=&[y = y'][b = b']\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y']
\end{aligned}
$$

*and the main proof of subgoal (1) proceeds by proving the invariance of* $[y = y']$ *as follows:*

$$
\begin{aligned}
& [y = y']\langle d_1 \cdot c_2 | d_1' \cdot c_2' \rangle \\
= & \quad \{ \; \textit{abbreviation and homomorfism} \} \\
& [y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool' \rangle \langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1 \rangle \\
& \langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle \\
= & \quad \{ \; \textit{R-Rand} \} \\
& [y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool' \rangle [y = y'][b = b'] \\
& bihomy \leftarrow y \; xor \; tt +_{[b=tt]} 1 y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1 bihomy \leftarrow y \; xor \; by' \leftarrow y' \; xor \; b'
\end{aligned}
$$

$=$ $\{$ *(38)* $\}$

$[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool' \rangle [y = y'][b = b']$
$\langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1 \rangle [y = y'][b = b']$
$\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle$

$=$ $\{$ *R-Assign* $\}$

$[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool' \rangle [y = y'][b = b']$
$\langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1 \rangle [y = y'][b = b']$
$\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y']$

$=$ $\{$ *(37)* $\}$

$[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool' \rangle \langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1 \rangle$
$\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y']$

$=$ $\{$ *homomorfism and abbreviation* $\}$

$[y = y']\langle d_1 \cdot c_2 | d_1' \cdot c_2' \rangle [y = y']$

- *subgoal (2):*

  $[y = y'][x \neq tt][x' = tt]\langle (d_2 \cdot c_2) | (d_1' \cdot c_2') \rangle = [y = y'][x \neq tt][x' = tt]\langle (d_2 \cdot c_2) | (d_1' \cdot c_2') \rangle [y = y']$

  *On one side, pogram* $(d_2 \cdot c_2)$ *yields* $y := y$ `xor` $ff$*, while on the other side, program* $(d_1' \cdot c_2')$ *yields*

  $d_1';\; c_2'$

  $=$ $\{$ *defn* $\}$

  $b' \xleftarrow{\$} dbool;\; ((y' \leftarrow y' \; xor \; tt) +_{[b'=tt]} 1) \cdot y' \leftarrow y' \; xor \; b'$

  $=$ $\{$ *(5)* $\}$

  $b' \xleftarrow{\$} dbool \cdot \big((y' \leftarrow y' \; xor \; tt \cdot y' \leftarrow y' \; xor \; b') +_{[b'=tt]} (y' \leftarrow y' \; xor \; b')\big)$

  $=$ $\{$ *(4) and (2)* $\}$

  $b' \xleftarrow{\$} dbool \cdot \big([b' = tt] \cdot (y' \leftarrow y' \; xor \; tt \cdot y' \leftarrow y' \; xor \; b')$
  $+_{[b'=tt]} [b' = ff](y' \leftarrow y' \; xor \; b')\big)$

  $=$ $\{$ *instantiation of* $b'$ $\}$

  $b' \xleftarrow{\$} dbool \cdot \big([b' = tt] \cdot (y' \leftarrow y' \; xor \; tt \cdot y' \leftarrow y' \; xor \; tt)$
  $+_{[b'=tt]} [b' = ff](y' \leftarrow y' \; xor \; ff)\big)$

  $=$ $\{$ *B.A.* $\}$

$$b' \xleftarrow{\$} dbool \cdot \big([b' = tt] \cdot (y' \leftarrow y' \; xor \; tt) +_{[b'=tt]} [b' = ff](y' \leftarrow y' \; xor \; ff)\big)$$

$=$ $\{$ *(5) and* $e +_e \neg\, e = 1\}$

$$b' \xleftarrow{\$} dbool \cdot y' \leftarrow y' \; xor \; ff$$

*Since variable* $b'$ *does not interfere in the assignment* $y' \leftarrow y' \; xor \; ff$*, we derive the post condition* $[y = y']$*.*

- *subgoal (3): symmetrical to the previous one relatively to variables* $x, x'$*.*
- *subgoal (4):*

$$[y = y'][x = tt][x \neq tt]\langle d_2 \cdot c_2 | d_2' \cdot c_2' \rangle$$

$=$ $\{$ *Abbreviations*$\}$

$$[y = y'][x = tt][x \neq tt]\langle b \leftarrow ff \cdot y \leftarrow y \; xor \; b | b' \leftarrow ff \cdot y' \leftarrow y' \; xor \; b' \rangle$$

$=$ $\{$ *homomorfism*$\}$

$$[y = y'][x = tt][x \neq tt]\langle b \leftarrow ff | b' \leftarrow ff \rangle \langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle$$

$=$ $\{$ *R-Assign*$\}$

$$[y = y'][x = tt][x \neq tt]\langle b \leftarrow ff | b' \leftarrow ff \rangle [y = y'][b = b']$$
$$\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y']$$

$=$ $\{$ *(37)*$\}$

$$[y = y'][x = tt][x \neq tt]\langle b \leftarrow ff | b' \leftarrow ff \rangle \langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y']$$

$=$ $\{$ *homomorfism*$\}$

$$[y = y'][x = tt][x \neq tt]\langle b \leftarrow ff \cdot y \leftarrow y \; xor \; b | b' \leftarrow ff \cdot y' \leftarrow y' \; xor \; b' \rangle [y = y']$$

# 5 Related work

The GKAT system was introduced in [13], which also introduced its probabilistic model with sub-Markov kernels. It was investigated further in [12], which in particular provides a semantics for which the equational theory is complete.

Relational Hoare logic was introduced in [7]. Probabilistic relational Hoare logic (pRHL) is due to Barthe and coauthors in [5], where it was motivated by the certification of cryptographic proofs.

The relational extension BiKAT of KAT was introduced in [1]. It is shown in this paper that the rules of relational Hoare logic [7] can be interpreted in BiKAT.

# 6 Conclusion and perspectives

In this work we have introduced a variant of KAT allowing to reason on relational properties of probabilistic programs. We have illustrated its expressivity by

proving how probabilistic relational Hoare logic [5] can be interpreted in it. In future work we would like to explore if the soundness theorem (soundness of pRHL in BigKAT) can be extended to the logic with the general form of *while* rule, without side condition ($E(c) = E(c') = 0$). We would also be interested in exploring the application of GKAT to unary (non-relational) properties of probabilistic programs, and for that to investigate the relationships with the probabilistic Hoare logic aHL of [3].

**Acknowledgements** The first and second authors have been supported by the french Program "Investissements d'avenir" (I-ULNE SITE / ANR-16-IDEX-0004 ULNE) managed by the National Research Agency.

# References

[1] 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`.

[2] Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. Easycrypt: A tutorial. In Alessandro Aldini, Javier López, and Fabio Martinelli, editors, *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 146–166. Springer, 2013. `doi:10.1007/978-3-319-10082-1\_6`.

[3] 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`.

[4] Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin C. Pierce. Programming language techniques for differential privacy. *ACM SIGLOG News*, 3(1):34–53, 2016. `doi:10.1145/2893582.2893591`.

[5] Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In Zhong Shao and Benjamin C. Pierce, editors, *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, pages 90–101. ACM, 2009. `doi:10.1145/1480881.1480894`.

[6] Santiago Zanella Béguelin, Gilles Barthe, Benjamin Grégoire, and Federico Olmedo. Formally certifying the security of digital signature schemes. In

*30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, pages 237–250. IEEE Computer Society, 2009. doi:10.1109/SP.2009.17.

[7] Nick Benton. Simple relational correctness proofs for static analyses and program transformations. In Neil D. Jones and Xavier Leroy, editors, *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004*, pages 14–25. ACM, 2004. doi:10.1145/964001.964003.

[8] Filippo Bonchi and Damien Pous. Checking NFA equivalence with bisimulations up to congruence. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 457–468. ACM, 2013. doi:10.1145/2429069.2429124.

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

[10] 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.

[11] Dexter Kozen. Kleene algebra with tests and commutativity conditions. In Tiziana Margaria and Bernhard Steffen, editors, *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings*, volume 1055 of *Lecture Notes in Computer Science*, pages 14–33. Springer, 1996. doi:10.1007/3-540-61042-1_35.

[12] Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 142:1–142:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.142.

[13] 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.

# Appendix

Proof of Proposition 2.2.

To prove this proposition we use the interpretation of assignment and samplings in the probabilistic model (Definition 2.1). We give the proof of axiom (2.1), the remaining ones are proved analogously. Given a probabilistic model $i$,

$$
\begin{array}{ll}
 & \mathcal{P}_i[\![x_1 \leftarrow t_1 \cdot x_2 \leftarrow t_2]\!](\sigma)(\sigma') \\
= & \quad \{ \text{ Definition 2.1} \} \\
 & \sum_{\sigma''} \mathcal{P}_i[\![x_1 \leftarrow t_1]\!](\sigma)(\sigma'') \times \mathcal{P}_i[\![x_2 \leftarrow t_2]\!](\sigma'')(\sigma') \\
= & \quad \{ \text{ Definition 2.1} \} \\
 & \sum_{\sigma''} \mathit{eval}(x_1 \leftarrow t_1)(\sigma'') \times \mathit{eval}(x_2 \leftarrow t_2)(\sigma') \\
= & \quad \{ \text{ Definition of } \mathit{eval} \} \\
 & \sum_{\sigma''} \delta_{\sigma''[x_1 \leftarrow t_1]} \times \delta_{\sigma'[x_2 \leftarrow t_2]} \\
= & \quad \{ \text{ commutativity of } \times \} \\
 & \sum_{\sigma''} \delta_{\sigma''[x_2 \leftarrow t_2]} \times \delta_{\sigma'[x_1 \leftarrow t_1]} \\
= & \quad \{ \text{ Definition of } \mathit{eval} \} \\
 & \sum_{\sigma''} \mathit{eval}(x_2 \leftarrow t_2)(\sigma'') \times \mathit{eval}(x_1 \leftarrow t_1)(\sigma') \\
= & \quad \{ \text{ Definition 2.1} \} \\
 & \sum_{\sigma''} \mathcal{P}_i[\![x_2 \leftarrow t_2]\!](\sigma)(\sigma'') \times \mathcal{P}_i[\![x_1 \leftarrow t_1]\!](\sigma'')(\sigma') \\
= & \quad \{ \text{ Definition 2.1} \} \\
 & \mathcal{P}_i[\![x_2 \leftarrow t_2 \cdot x_1 \leftarrow t_1]\!](\sigma)(\sigma')
\end{array}
$$

Proof of Lemma 3.3.

To prove the first equality, reason

$$
\begin{array}{ll}
 & \langle e| \cdot \langle c_1|c_1' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
 & \langle e \cdot c_1|c_1' \rangle \\
= & \quad \{ \text{ (U8)} \} \\
 & \langle e \cdot (c_1 +_e c_2)|c_1' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
 & \langle e| \cdot \langle c_1 +_e c_2|c_1' \rangle
\end{array}
$$

For the second equality, we reason analogously

$$
\begin{aligned}
& \langle \neg\ e| \cdot \langle c_2 | c_2' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
& \langle \neg\ e \cdot c_2 | c_2' \rangle \\
= & \quad \{ \text{ (U8)} \} \\
& \langle \neg\ e \cdot (c_2 +_{\neg\ e}\ c_1) | c_2' \rangle \\
= & \quad \{ \text{ (U2)} \} \\
& \langle \neg\ e \cdot (c_1 +_e\ c_2) | c_2' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
& \langle \neg\ e| \cdot \langle c_1 +_e\ c_2 | c_2' \rangle
\end{aligned}
$$

Proof of Lemma 3.4.
To prove the equality, first note that

$$
\begin{aligned}
& \langle e +_e \neg\ e | e' +_{e'} \neg\ e' \rangle \\
= & \langle e \cdot e + \neg\ e \cdot \neg\ e | e' \cdot e' + \neg\ e' \cdot \neg\ e' \rangle \\
= & \langle 1 | 1 \rangle \\
= & 1
\end{aligned}
$$

by axiom (30) and Boolean algebra.
Using this observation, we reason for (48)

$$
\begin{aligned}
& \phi \cdot \langle e +_e \neg\ e | e' +'_e \neg\ e' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
& \phi \cdot \langle e +_e \neg\ e| \cdot |e' +'_e \neg\ e' \rangle \\
= & \quad \{ \text{ (U5')} \} \\
& \phi \cdot (\langle e +_e \neg\ e| \cdot |e' \rangle +_{e'} \langle e +_e \neg\ e | \neg\ e' \rangle \\
= & \quad \{ \text{ (U5)} \} \\
& \phi \cdot ((\langle e| \cdot |e' \rangle +_e \langle \neg\ e| \cdot |e' \rangle) + e'(\langle e| \cdot |\neg\ e' \rangle +_e \langle \neg\ e| \cdot |\neg\ e' \rangle)) \\
= & \quad \{ \text{ (U5')} \} \\
& (\phi \cdot \langle e| \cdot |e' \rangle +_e \phi \cdot \langle \neg\ e| \cdot |e' \rangle) + e'(\phi \cdot \langle e| \cdot |\neg\ e' \rangle +_e \phi \cdot \langle \neg\ e| \cdot |\neg\ e' \rangle) \\
= & \quad \{ \text{ (side condition)} \} \\
& \phi \langle e | e' \rangle + e' \phi \langle \neg\ e | \neg\ e' \rangle \\
= & \quad \{ \text{ (U5')} \} \\
& \phi(\langle e | e' \rangle + e' \langle \neg\ e | \neg\ e' \rangle)
\end{aligned}
$$

Proof of Proposition 3.6.

$$ec^{(e)}$$

$$
\begin{array}{ll}
= & \{\ (11)\} \\
& e(cc^{(e)} +_e 1) \\
= & \{\ \text{fact u5' and (10)}\} \\
& ecc^{(e)} +_e e \\
= & \{\ (2) \text{ and } (4)\} \\
& ecc^{(e)} +_e \neg\ e \cdot e \\
= & \{\ \text{B.A.}\} \\
& ecc^{(e)} +_e 0 \\
= & \{\ \text{fact u6 and B.A.}\} \\
& ecc^{(e)}
\end{array}
$$

Proof of Lemma 3.8.

$$
\begin{array}{ll}
& \phi(\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle \neg\ e|c'^{(e')}\rangle) \\
= & \{\ \text{homomorfism and } (4)\} \\
& \phi(\langle e|\neg\ e'\rangle\langle c^{(e)}| +_{\langle e|} \langle \neg\ e|c'^{(e')}\rangle) \\
= & \{\ \text{fact u5'}\} \\
& \phi\langle e|\neg\ e'\rangle\langle c^{(e)}| +_{\langle e|} \phi\langle \neg\ e|c'^{(e')}\rangle \\
= & \{\ (33) \text{ and } (7)\} \\
& 0 +_{\langle e|} \phi\langle \neg\ e|c'^{(e')}\rangle \\
= & \{\ (2) \text{ and fact u6}\} \\
& \langle \neg\ e|\phi\langle \neg\ e|c'^{(e')}\rangle \\
= & \{\ \text{B.A.}\} \\
& \phi\langle \neg\ e|c'^{(e')}\rangle \\
= & \{\ (11)\} \\
& \phi\langle \neg\ e|c'c'^{(e')} +_{e'} 1\rangle \\
= & \{\ (2), (4) \text{ and } (10)\} \\
& \phi\langle \neg\ e|e'c'c'^{(e')} +_{e'} (\neg\ e')\rangle \\
= & \{\ \text{fact u5' and homomorfism}\} \\
& \phi(\langle \neg\ e|e'c'c'^{(e')}\rangle +_{|e'\rangle} \langle \neg\ e|\neg\ e'\rangle)
\end{array}
$$

$$
\begin{aligned}
&= \quad \{ \text{ homomorfism and (fact u5')} \} \\
&\quad \phi\langle \neg\ e|e'\rangle|c'c'^{(e')}\rangle +_{|e'\rangle}\ \phi\langle \neg\ e|\neg\ e'\rangle \\
&= \quad \{ \text{ (33)} \} \\
&\quad 0 +_{|e'\rangle}\ \phi\langle \neg\ e|\neg\ e'\rangle \\
&= \quad \{ \text{ (2) and fact u6} \} \\
&\quad |\neg\ e'\rangle\phi\langle \neg\ e|\neg\ e'\rangle \\
&= \quad \{ \text{ B.A.} \} \\
&\quad \langle \neg\ e|\neg\ e'\rangle\phi
\end{aligned}
$$

Proof of Theorem 3.9.
*R-Seq rule*:

$$
\begin{aligned}
&\quad \phi \cdot \langle c_1 \cdot c_2|c_1' \cdot c_2'\rangle \\
&= \quad \{ \text{ homomorfism} \} \\
&\quad \phi \cdot \langle c_1|c_1'\rangle \cdot \langle c_2|c_2'\rangle \\
&= \quad \{ \text{ premises} \} \\
&\quad \phi \cdot \langle c_1|c_1'\rangle \cdot \psi \cdot \langle c_2|c_2'\rangle\xi \\
&= \quad \{ \text{ premise} \} \\
&\quad \phi \cdot \langle c_1|c_1'\rangle \cdot \langle c_2|c_2'\rangle \cdot \xi \\
&= \quad \{ \text{ homomorfism} \} \\
&\quad \phi \cdot \langle c_1 \cdot c_2|c_1' \cdot c_2'\rangle \cdot \xi
\end{aligned}
$$

*R-Cond right rule*:

$$
\begin{aligned}
&\quad \phi\langle c_1|c_1' +_e c2'\rangle \\
&= \quad \{ \text{ (30)} \} \\
&\quad \phi(|e\rangle +_{|e\rangle}\ |\neg\ e\rangle)\langle c_1|c_1' +_e c2'\rangle \\
&= \quad \{ \text{ (5)} \} \\
&\quad \phi|e\rangle\langle c_1|c_1' +_e c2'\rangle +_{|e\rangle}\ \phi|\neg\ e\rangle\langle c_1|c_1' +_e c2'\rangle \\
&= \quad \{ \text{ (31) and (19)} \} \\
&\quad \phi\langle c_1|ec_1'\rangle +_{|e\rangle}\ \phi\langle c_1|\neg\ ec_2'\rangle \\
&= \quad \{ \text{ premises} \} \\
&\quad \phi\langle c_1|ec_1'\rangle\psi +_{|e\rangle}\ \phi\langle c_1|\neg\ ec_2'\rangle\psi \\
&= \quad \{ \text{ (31), (19), (5) and (30) reverse steps} \}
\end{aligned}
$$

$$
\begin{aligned}
& \phi(|e\rangle +_{|e\rangle} |\neg\, e\rangle)\langle c_1|c_1' +_e c2'\rangle\psi \\
= & \quad \{ \text{ B.A.}\} \\
& \phi\langle c_1|c_1' +_e c2'\rangle\psi
\end{aligned}
$$

*R-Cond left rule*: symetrical to *R-Cond right rule*.
*R-Sub rule*:

$$
\begin{aligned}
& \phi' \cdot \langle c|c'\rangle \\
= & \quad \{ \text{ ax (30)}\} \\
& \phi' \cdot \phi \cdot \langle c|c'\rangle \\
= & \quad \{ \text{ premise}\} \\
& \phi' \cdot \phi \cdot \langle c|c'\rangle \cdot \psi \\
= & \quad \{ \text{ ax (30)}\} \\
& \phi' \cdot \phi \cdot \langle c|c'\rangle \cdot \psi \cdot \psi' \\
= & \quad \{ \text{ premise}\} \\
& \phi' \cdot \phi \cdot \langle c|c'\rangle \cdot \psi' \\
= & \quad \{ \text{ ax (30)}\} \\
& \phi' \cdot \langle c|c'\rangle \cdot \psi'
\end{aligned}
$$

*R-Case rule*:

$$
\begin{aligned}
& \phi \cdot \langle c|c'\rangle \\
= & \quad \{ \text{ B.A., ax (30)}\} \\
& \phi \cdot (\phi'_{\phi'}, \phi') \cdot \langle c|c'\rangle \\
= & \quad \{ \text{ (U5), (U5') of GKAT}\} \\
& \phi \cdot \phi' \cdot \langle c|c'\rangle +_{\phi'} \phi \cdot \neg\, \phi' \cdot \langle c|c'\rangle \\
= & \quad \{ \text{ premises}\} \\
& \phi \cdot \phi' \cdot \langle c|c'\rangle \cdot \psi +_{\phi'} \phi \cdot \neg\, \phi' \cdot \langle c|c'\rangle \cdot \psi \\
= & \quad \{ \text{ (U5') of GKAT}\} \\
& \phi \cdot (\phi' \cdot \langle c|c'\rangle \cdot \psi +_{\phi'} \neg\, \phi' \cdot \langle c|c'\rangle \cdot \psi) \\
= & \quad \{ \text{ (U5) of GKAT}\} \\
& \phi \cdot ((\phi' +_{\phi'} \neg\, \phi') \cdot \langle c|c'\rangle \cdot \psi) \\
= & \quad \{ \text{ (B.A.), ax (30)}\} \\
& \phi \cdot \langle c|c'\rangle \cdot \psi
\end{aligned}
$$