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

[1]Univ. Lille, CNRS, Inria, Centrale Lille, UMR9189 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

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 (GKAT), 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 interpret 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 Algebra with Tests (KAT) stands out as an elegant, simple and automatizable framework [12, 10]. 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 [9]. It has also been implemented in a library for the Coq proof-assistant [13]. 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, or extensional equivalence. Another example is 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, network programming or differential privacy. In those scenarios many crucial properties are also relational ones. For instance in cryptography one can express the fact that a randomized encryption scheme is safe as 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. [5]), in order to protect private data one might want to verify that two executions of a given program on two databases 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 [8], 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 [6]. It

Property nature	Unary	Relational deterministic	Relational probabilistic
Hoare logic	HL	RHL	pRHL
Algebra	KAT	BiKAT	**BiGKAT**
Examples of properties	Partial correctness	Validation of program transformations	Probabilistic non-interference

Figure 1: Program logics and algebras

has then been extensively applied to the verification of cryptographic schemes, in particular through the development of the Certicrypt [7] and Easycrypt [2] tools.

However one would still benefit from additional techniques 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 algebra with tests are promising in these respects. In particular they facilitate the reasoning on simple program transformations.

Our goal 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], allowing to apply the KAT approach to reasoning on pairs of programs. Unfortunately standard KAT techniques cannot be applied directly to probabilistic programs, since there is no known probabilistic interpretation for KAT. To handle this question, recent progress was made by the introduction of Guarded Kleene Algebra with tests (GKAT) [15], in which non-deterministic union and iteration are replaced by guarded union and iteration. The main motivation for the introduction of GKAT was to design a more efficient version of KAT where the complexity of the decision procedure is reduced, but it was also shown that GKAT admits a probabilistic model that can be used to interpret probabilistic programs.

Our strategy is thus to adapt the relational extension BiKAT to the setting of GKAT, in order to apply this relational approach to pairs of probabilistic programs. We want to apply such framework to probabilistic programs as those from 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
$c ::= \mathbf{skip} \mid x \leftarrow t \mid x \overset{\$}{\leftarrow} d \mid c \cdot c \mid \mathbf{if}\ p\ \mathbf{then}\ c\ \mathbf{else}\ c \mid \mathbf{while}\ p\ \mathbf{do}\ c$	compound programs

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. $+$, $\sqrt{}$);
- $\mathrm{p} \in \mathrm{P}$ 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 $\mathbb{Z}$, i.e. maps $d : \mathbb{Z} \to [0, 1]$ such that $\sum_{z\in\mathbb{Z}} d(z) \leq 1$;
- The command $x \leftarrow t$ assigns the value of t to x, and $x \overset{\$}{\leftarrow} d$ samples from distribution d and assigns the result to x.

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 [11] and Relational Hoare Logic in BiKAT [1], filling in this way the middle-right square in Fig.1. This will raise some specific difficulties, in particular for proving the validity of the rule dealing with the *while* construct, and the addition of a new axiom for GKAT. Finally we will illustrate the benefits of our framework on some examples, including a random walk in dimension 1.

Outline of the paper. In Sect. 2 we recall 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 two examples. The proofs omitted in the paper can be found in the Appendix.

2 Guarded Kleene algebra with tests

This section recalls the language and the semantics of *Guarded Kleene Algebra with Tests (GKAT)* [15], an abstraction of imperative programs where conditionals $(c_1 +_e c_2)$ and loops $(c^{(e)})$ are guarded by Boolean predicates e. As explained before, 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 [15], we are more interested in the second advantage for the purpose of this paper.

2.1 Syntax

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

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

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

where $a \in \Sigma$, for any $e, e_1, e_2 \in \mathrm{BExp}$, operators $\cdot$, $+$ and $\neg$ denote conjunction, disjunction and negation, respectively, and, for any $c, c_1, c_2 \in \mathrm{Exp}$, 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, respectively as $c_1 +_e c_2 \overset{\text{def}}{\equiv}$ **if** e **then** c_1 **else** c_2 and $c^{(e)} \overset{\text{def}}{\equiv}$ **while** e **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 $c_1, c_2 \in \mathrm{Exp}$.

2.2 Semantics

We now present the semantic interpretation of GKAT that we will be using, the *Probabilistic model* [15]. Note that more interpretations of GKAT are presented in [15], namely a relational model and a language model. We first revise some basic concepts needed for the semantics. Given a countable set S, $\mathcal{D}(S)$ is the set of *probability sub-distributions* over S, i.e. the set of functions $f : S \to [0, 1]$ summing 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

$$w \to [w = s] = \begin{cases} 1, \text{ if w=s} \\ 0, \text{ otherwise} \end{cases}$$

In general terms, 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 probability sub-distributions.

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

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

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
$\mathit{eval}(x \leftarrow t)(\sigma) := \delta_{\sigma[x\leftarrow t]}$ and $\mathit{eval}(x \overset{\$}{\leftarrow} d)(\sigma) := \sum_{t\in\mathbb{Z}} d(t) \cdot \delta_{\sigma[x\leftarrow t]}$.

2.3 Axioms

The theory of GKAT introduced in [15] is given by the axioms from Fig. 2. Note in particular for the

$$
\begin{array}{rcll}
c +_e c & = & c & (1)\\
c_1 +_e c_2 & = & c_2 +_{\neg\, e} c_1 & (2)\\
(e +_b f) +_c g & = & e +_{bc} (f +_c g) & (3)\\
c_1 +_e c_2 & = & ec_1 +_e c_2 & (4)\\
c_1 c_3 +_e c_2 c_3 & = & (c_1 +_e c_2) \cdot c_3 & (5)\\
(c_1 \cdot c_2) \cdot c_3 & = & c_1 \cdot (c_2 \cdot c_3) & (6)\\
0 \cdot c & = & 0 & (7)
\end{array}
\qquad
\begin{array}{rcll}
c \cdot 0 & = & 0 & (8)\\
1 \cdot c & = & c & (9)\\
c \cdot 1 & = & c & (10)\\
c^{(e)} & = & c \cdot c^{(e)} +_e 1 & (11)\\
(c +_{e_2} 1)^{(e_1)} & = & (e_2 \cdot c)^{(e_1)} & (12)\\
\dfrac{c_3 = c_1 \cdot c_3 +_e c_2}{c_3 = c_1^{(e)} \cdot c_2} & \text{if} & E(c_1) = 0 & (13)
\end{array}
$$

Figure 2: Axiomatisation of Guarded Kleene algebra with tests

fixpoint axiom (13). Intuitively, it says that if expression c_3 chooses (using guard e) between executing c_1 and looping again, and executing c_2, then c_3 is a e-guarded loop followed by c_2. However, the rule is not sound in general (see [15] 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: $E(e) := e$, $E(a) := 0$, $E(c_1 +_e c_2) := e \cdot E(c_1) + \neg\, e \cdot E(c_2)$, $E(c_1 \cdot c_2) := E(c_1) \cdot E(c_2)$, $E(c^{(e)}) := \neg\, e$. 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
$c_1;\ (e;\ c_2 + \neg\ e;\ c_3) = c_1;\ e;\ c_2 + c_1;\ \neg\ e;\ c_3$ is not an **if-then-else** statement; 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 $c \in \mathrm{Exp}$; it only holds for the particular case of $e \in \mathrm{BExp}$, i.e. if e is a test.

In the Appendix we list additional derivable equations in GKAT, also given in [15].

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 $e_1 +_e e_2$, for any $e \in \mathrm{BExp}$. We thus add the following new axiom

$$e_1 +_e e_2 = e \cdot e_1 + \neg\, e \cdot e_2 \tag{14}$$

to the theory of GKAT which expresses the guarded sum $+_e$, for any $e \in \mathrm{BExp}$, in terms of the disjunction $+$ on tests. This axiom is valid in the probabilistic model. By Boolean reasoning, we can observe that $e \cdot e + \neg\, e \cdot \neg\, e = 1$. Such property will be useful later to prove the soundness of R-Case rule (23).

Additionally, we propose in the Appendix an equational theory for 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.

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], which we define over a GKAT.

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

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

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

$$\forall\, c_1, c_2 \in A,\ \langle c_1|\fatsemi|c_2\rangle = |c_2\rangle \fatsemi \langle c_1| \tag{15}$$

We call $\ddot{A}$ *the underlying GKAT, and elements of* $\ddot{B}$ *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, for any $e_1,\ e_2,\ e \in B,\ c_1,\ c_2,\ c \in A$, the properties $\langle e_1 + e_2| = \langle e_1| \oplus \langle e_2|$, $\langle c_1 \cdot c_2| = \langle c_1| \fatsemi \langle c_2|$, $\langle c_1 +_e c_2| = \langle c_1| \oplus_E \langle c_2|$ and $\langle c^{(e)}| = \langle c|^E$ hold, where E stands for $\langle e| \in \ddot{B}$. Similarly for $|\ \rangle$. The operators have the same precedence as in 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 constants 1 and 0 in GKAT stand for $\fatsemi$, $\ddot{\neg}$, $\oplus_{\langle e|}$ ($\oplus_{|e\rangle}$), $\ddot{1}$ and $\ddot{0}$, respectively. Often we go even further and omit the operator $\cdot$ and we write $\langle c_1|\langle c_2|\ (|c_1\rangle|c_2\rangle)$ for $\langle c_1| \cdot \langle c_2|\ (|c_1\rangle \cdot |c_2\rangle)$.

3.1 Encoding pRHL in BiGKAT

In this section we want to prove that *probabilistic relational Hoare logic (pRHL)* [6] can be soundly encoded in BiGKAT. For that goal let us briefly recall *probabilistic relational Hoare logic (pRHL)*, which can be understood as an extension of Benton's Relational Hoare Logic [8] 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 ϕ, ψ (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.

In the probabilistic scenario, however, the evaluation of a program on a memory gives a subdistribution (Definition 2.1). The system pRHL thus lifts relations over memories to relations over distributions, which we restrict, in our setting, to subdistributions. To define that, following [6], we use a monadic semantics. The measure monad $M(X)$ is defined as $M(X) \stackrel{\text{def}}{=} (X \to [0,1]) \to [0,1]$ and its operators are:

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

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

Intuitively, the value of a subdistribution d of $M(X)$ on an element s of X is given by $d(\delta_s)$. The liftings to subdistributions of a unary predicate P and of a binary relation ϕ 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 as $\pi_1(d) \stackrel{\text{def}}{=} \texttt{bind}\ d\ (\lambda(x,y).\texttt{unit}\ x)$ and $\pi_2(d) \stackrel{\text{def}}{=} \texttt{bind}\ d\ (\lambda(x,y).\texttt{unit}\ y)$.

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

Definition 3.2. *Given two probabilistic programs* c*,* c' *and* ϕ*,* ψ *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* ϕ *and postcondition* ψ*.*

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

$$\varphi \cdot \langle c|c'\rangle = \varphi \cdot \langle c|c'\rangle \cdot \psi$$

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

- Note that we do not use the encoding $\varphi \cdot \langle c|c'\rangle \leq \varphi \cdot \langle c|c'\rangle \cdot \psi$ since in GKAT and BiGKAT there is no natural notion of order $\leq$ as in KAT [12, 10] ;
- We do not use either the encoding $\varphi \cdot \langle c|c'\rangle \cdot \neg\, \psi = 0$. 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.

We now display on Figure 3 the rules of pRHL defined in [6], with a restriction on the rule for *While* (*Weak R-Whl rule*) which we will explain below. The support $\mathtt{supp}\{d\}$ of a distribution d is defined by: $(s \in \mathtt{supp}\{d\})$ iff $d(\delta_s) \neq 0$.

- *R-Assign rule*:

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

- *R-Rand assign rule*:

$$\frac{h \lhd (d, d') \wedge \forall\, v \in \mathtt{supp}([\![d]\!]).\psi[v/x, h(v)/x']}{x \overset{\$}{\leftarrow} d \sim x' \overset{\$}{\leftarrow} d' : \varphi \Rightarrow \psi}$$

- *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*:

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

- *R-Sub rule*:

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

- *R-Case rule*:

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

- *Weak R-Whl rule*:

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

Figure 3: Probabilistic Relational Hoare Logic rules (pRHL)

There are also one-sided versions of some of these rules, which are just particular cases, and so we list them in the Appendix. We use different notation for pre and post conditions (φ, ψ) and for guards ($\langle e|$, $|e'\rangle$). Note in particular the side condition $\phi \Rightarrow e \doteq e'$ in rules *R-Cond* and *Weak 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

$$\phi\langle e|\neg\, e'\rangle = 0 \quad \phi\langle \neg\, e|e'\rangle = 0 \tag{16}$$

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 [6] is the additional premise condition $(E(c) = E(c') = 0)$ in the *Weak R-Whl* rule: ours is weaker, since we impose that two commands c, c' are guaranteed to perform some action, property that we will use to prove the soundness of the rule.

More precisely we use this condition in the proof of the intermediary Lemma 3.4. This rule is actually expressive enough for many examples. Note also for the *R-Assign*, *R-Assign left* and *R-Rand* rules, which are axioms: the first one derives a valid Hoare triple with the substitution of variables x, x' by expressions v, v', respectively; the second One derives an assignment on the left-hand side, while the right-hand side is a **skip** instruction; the third one 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 $v \in \mathrm{supp}\{d\}$, $h(v) \in \mathrm{supp}\{d'\}$;
- $P_{x\sim d}[x = v] = P_{x\sim d'}[x = h(v)]$

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 [6].

Now, to show that the rules of Figure 3 are sound in BiGKAT, we interprete them as follows, by using the encoding of pRHL judgements as BiGKAT equations defined previously:

- *R-Assign rule*:

$$\varphi[v/x, v'/x']\langle x \leftarrow v | x' \leftarrow v'\rangle = \varphi[v/x, v'/x']\langle x \leftarrow v | x' \leftarrow v'\rangle\varphi \tag{17}$$

- *R-Rand assign rule*:

$$h \lhd (d, d') \wedge \forall\, v \in \mathtt{supp}([\![d]\!]).\psi[v/x, h(v)/x'] \Rightarrow \varphi\langle x \xleftarrow{\$} d | x' \xleftarrow{\$} d'\rangle = \varphi\langle x \xleftarrow{\$} d | x' \xleftarrow{\$} d'\rangle\psi \tag{18}$$

- *R-Seq rule*:
-

$$\phi\langle c_1|c_1'\rangle = \phi\langle c_1|c_1'\rangle\psi \;\;\wedge\;\; \psi\langle c_2|c_2'\rangle = \psi\langle c_2|c_2'\rangle\xi \Rightarrow \;\; \phi\langle c_1 \cdot c_2|c_1' \cdot c_2'\rangle = \phi\langle c_1 \cdot c_2|c_1' \cdot c_2'\rangle\xi \tag{19}$$

- *R-Cond rule*:
-

$$\begin{aligned}&\phi \overline{\leq} e \ddot{=} e' \;\wedge\; \phi \cdot \langle e|e'\rangle \cdot \langle c_1|c_1'\rangle = \phi \cdot \langle e|e'\rangle \cdot \langle c_2|c_2'\rangle \cdot \psi \;\;\wedge\\ &\phi \cdot \;\langle \neg\, e|\neg\, e'\rangle \cdot \langle e|e'\rangle = \phi \cdot \;\langle \neg\, e|\neg\, e'\rangle \cdot \langle c_1|c_1'\rangle \cdot \psi\\ &\Rightarrow\; \phi \cdot \langle c_1 +_e c_2|c_1' +_{e'} c_2'\rangle = \phi \cdot \langle c_1 +_e c_2|c_1' +_{e'} c_2'\rangle \cdot \psi\end{aligned} \tag{20}$$

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

$$\phi \;\leq\; e \ddot{=} e' \;\wedge\; \phi \cdot \langle e|e'\rangle\langle c|c'\rangle = \phi \cdot \langle e|e'\rangle\langle c|c'\rangle \cdot \phi \Rightarrow \phi \cdot \langle c^{(e)}|e'^{(e')}\rangle = \phi \cdot \langle c^{(e)}|c'^{(e')}\rangle \cdot \phi \tag{21}$$

- *R-Sub rule*:

$$\phi' \leq \phi \;\;\wedge\;\; \phi\langle c|c'\rangle = \phi\langle c|c'\rangle\psi \;\;\wedge\;\; \psi \leq \psi' \;\;\Rightarrow\;\; \phi'\langle c|c'\rangle = \phi'\langle c|c'\rangle\psi' \tag{22}$$

- *R-Case rule*:

$$\phi \cdot \phi' \cdot \langle c|c'\rangle = \phi \cdot \phi' \cdot \langle c|c'\rangle\psi \;\; \wedge \phi \cdot \neg\, \phi' \cdot \langle c|c'\rangle = \phi \cdot \neg\, \phi' \cdot \langle c|c'\rangle\psi \Rightarrow \phi \cdot \langle c|c'\rangle = \phi \cdot \langle c|c'\rangle\psi \tag{23}$$

Note that the encoding of the one-sided rules are listed in the Appendix. Our goal is now to prove that these rules are valid in any BiGKAT. To prove some of these rules, namely *R-Cond* and *Weak R-Whl*, we need to establish some auxiliary results.

Lemma 3.1. *In any BiGKAT the following two equalities hold:*

$$\langle e| \cdot \langle c_1|c_1'\rangle = \langle e| \cdot \langle c_1 +_e c_2|c_1'\rangle \tag{24}$$

$$\langle \neg\, e| \cdot \langle c_2|c_2'\rangle = \langle e| \cdot \langle c_1 +_e c_2|c_2'\rangle \tag{25}$$

Lemma 3.2. *For any BiGKAT,*

$$\phi \cdot \langle e +_e \neg\, e|e' +_{e'} \neg\, e'\rangle = \phi \cdot (\langle e|e'\rangle +_{e'} \langle \neg\, e|\neg\, e'\rangle) \tag{26}$$

Now we state the invariance result, adapted from the standard result on KAT and the equivalent one for GKAT, which was proved in [15]. It will be useful for the *R-Whl* rule.

Lemma 3.3 (Invariance)**.** *Let $c, c' \in A$ and $\phi, e \in \ddot{B}$. If $\phi e\langle c|c'\rangle = \phi e\langle c|c'\rangle\phi$ then $\phi\langle c|c'\rangle^{(e)} = (\phi\langle c|c'\rangle)^{(e)}\phi$.*

Proof. Since a BiGKAT is a GKAT (Definition 3.1), it holds by the invariance lemma (Lemma 3.11) of GKAT [15]. □

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

Proposition 3.1. *For any e, c in GKAT, $ecc^{(e)} = ec^{(e)}$.*

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

Lemma 3.4 (Expansion)**.** *The following property holds in any BiGKAT. Assume $E(c) = E(c') = 0$, then we have:*

$$\langle c^{(e)}|c'^{(e')}\rangle = \langle c|c'\rangle^{\langle e|e'\rangle}(\langle c|\neg\ e'\rangle^{\langle e|} +_{\langle e|} \langle\neg\ e|c'\rangle^{|e'\rangle}) \tag{27}$$

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 3.4 differs from the proof of the analogous lemma in BiKAT [1] and uses the the fixpoint axiom (13) (see Appendix). Now we establish the following lemma that it is useful to prove the soundness of the *Weak R-Whl* rule (21).

Lemma 3.5. *In any BiGKAT, if $\phi \leq e \ddot{=} e'$ then we have:*

$$\phi(\langle c^{(e)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(e')}\rangle) = \langle\neg\ e|\neg\ e'\rangle\phi \tag{28}$$

Finally we obtain the main result on the soundness of pRHL rules in BiGKAT.

Theorem 3.1 (Soundness of pRHL in BiGKAT)**.** *The rules of probabilistic Relational Hoare Logic, that is to say (17)-(23) and (45)-(47), are sound in any BiGKAT.*

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 probabilistic distributions 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. We explain the examples an give the main ideas of the proofs, leaving the complete details for appendix.

Example 4.1. *Consider the following program:*

```
1   var x : mybool; var y : mybool; var b : mybool; // x private variable, y public
    variable
2   if (x = tt) {
3       b <-$ dmybool;
4       if (b = tt) {
5           y <- y xor tt;
6       }
7   }
8   else {
9       b <- ff;
10  };
11      y <- y xor b;
12
```

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 $\vdash c \sim c' : [y = y'] \Rightarrow [y = y']$, which is translated into the BiGKAT equation $[y = y']\langle c|c'\rangle = [y = y']\langle c|c'\rangle[y = y']$.

Program c is encoded as the BiGKAT term

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

In order to simplify the writing we denote $d_1 = b \xleftarrow{\$} dbool$; $(y \leftarrow y \ xor \ tt)$, $d_2 = b \leftarrow ff$ and $c_2 = (y \leftarrow y \ xor \ b)$. We then use some equational reasoning to obtain

$$\begin{aligned}
& [y = y']\langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x'=tt]} d_2') \cdot c_2' \rangle \\
= \quad & \{ \ \textit{appendix (48)} \} \\
& [y = y'][x = x']\langle (d_1 \cdot c_2) +_{[x=tt]} (d_2 \cdot c_2) | (d_1' \cdot c_2') +_{[x'=tt]} (d_2' \cdot c_2') \rangle
\end{aligned}$$

which we subdivide into four subgoals, since as guarded sums, they depend on the evaluation of $[x = tt]$ and $[x' = tt]$: (1)$[x = tt][x' = tt]$, (2)$[x \neq tt][x' = tt]$, (3)$[x = tt][x' \neq tt]$ and (4)$[x \neq tt][x' \neq tt]$.

We present here the proof of subgoal (2) to illustrate the use of equational reasoning on BiGKAT, leaving the proofs of the other cases in appendix.

***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 $y := y \ \ \texttt{xor} \ ff$, while on the other side, program $(d_1' \cdot c_2')$ yields

$$\begin{aligned}
& d_1'; \ c_2' \\
= \quad & \{ \ \textit{defn} \} \\
& b' \xleftarrow{\$} dbool; \ ((y' \leftarrow y' \ xor \ tt) +_{[b'=tt]} 1) \cdot y' \leftarrow y' \ xor \ b' \\
= \quad & \{ \ \textit{(5)} \} \\
& b' \xleftarrow{\$} dbool \cdot \big((y' \leftarrow y' \ xor \ tt \cdot y' \leftarrow y' \ xor \ b') +_{[b'=tt]} (y' \leftarrow y' \ xor \ b')\big) \\
= \quad & \{ \ \textit{(4) and (2)} \} \\
& b' \xleftarrow{\$} dbool \cdot \big([b' = tt] \cdot (y' \leftarrow y' \ xor \ tt \cdot y' \leftarrow y' \ xor \ b') \\
& +_{[b'=tt]} [b' = ff](y' \leftarrow y' \ xor \ b')\big) \\
= \quad & \{ \ \textit{instantiation of } b' \} \\
& b' \xleftarrow{\$} dbool \cdot \big([b' = tt] \cdot (y' \leftarrow y' \ xor \ tt \cdot y' \leftarrow y' \ xor \ tt) \\
& +_{[b'=tt]} [b' = ff](y' \leftarrow y' \ xor \ ff)\big) \\
= \quad & \{ \ \textit{B.A.} \} \\
& b' \xleftarrow{\$} dbool \cdot \big([b' = tt] \cdot (y' \leftarrow y' \ xor \ tt) +_{[b'=tt]} [b' = ff](y' \leftarrow y' \ xor \ ff)\big) \\
= \quad & \{ \ \textit{(5) and } e +_e \neg e = 1 \} \\
& b' \xleftarrow{\$} dbool \cdot y' \leftarrow y' \ xor \ ff
\end{aligned}$$

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

Example 4.2. *Consider the following program, corresponding to the classic example of Random walk in dimension 1, a path which describes a succession of random steps (see [3]). That means that, starting in an initial position, at each step we toss a fair coin. If heads, we move one step to the right, otherwise we move one step to the left. The variable H records the history of coin flips.*

```
pos<- start; H<- []; i<-0;
while (i<k) do{
    b<-${0,1};
    H<-b:: H;
    if b then pos++ else pos--;
    i<-i+1;
}
return pos;
```

The goal of this example is to prove that, by taking two executions of the program above, the corresponding paths converge as the number of steps increases. Let us denote by

$$c := pos \leftarrow start;\ H < -[\,];\ i \leftarrow 0;\ (b \overset{\$}{\leftarrow} \{0,1\};\ H \leftarrow b :: H;\ ((pos++)\ +_b\ (pos--));\ i \leftarrow i+1)^{(i<k)}$$

the encoding of the program in BiGKAT, and by c' one copy.

We follow the approach of [3], Sect. 3.1. Consider two processes that start at locations start and $start'$ such that $start' - start = 2n \geq 0$. We define $\Sigma(H)$ as the number of 1 in H minus the number of 0 (so the net change of position of a process with history H). Then $P(H)$ is the predicate which holds when H contains a prefix H_0 such that $\Sigma(H_0) = n$.

Formally, we want to prove the pRHL judgment $\vdash\ c \sim c' : [start + 2n = start'] \Rightarrow [P(H) \rightarrow pos = pos']$, which corresponds in BiGKAT to the equation $[start + 2n = start']\langle c|c'\rangle = [start + 2n = start']\langle c|c'\rangle[P(H) \rightarrow (pos = pos')]$.

In order to simplify the writing, we also abbreviate the loop body

$$d = b \overset{\$}{\leftarrow} 0,1;\ H \leftarrow b :: H;\ ((pos++)\ +_b\ (pos--));\ i \leftarrow i+1$$

A way to relate the while loops of c and c' is through an invariant. The idea to obtain an invariant follows the reasoning: before the two points meet their trajectories are mirrored and after they meet they coincide forever. Thus, the invariant we present is

$$\varphi \hat{=} ((pos \neq pos') \rightarrow (pos = i + \Sigma(H) \wedge pos' = i' - \Sigma(H')) \wedge (P(H) \rightarrow (pos = pos')) \wedge (i = i') \wedge (k = k')$$

To relate the loop bodies we perform a case distinction analysis on pos, pos': if they are different, their moves are mirrored, otherwise they move together. In that sense, we need one coupling function h for each case: if $pos = pos'$, $h \overset{def}{=} id$, if $pos \neq pos'$, $h \overset{def}{=} \neg$. We also know that as long as we are "inside" the loop body, the predicate $[i < k \wedge i' < k']$ always holds. Hence we reason using the R-case rule (23) to do the case analysis as follows:

$$\begin{aligned}
&[\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle d|d'\rangle = [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle d|d'\rangle\varphi \\
\wedge&[\varphi \wedge (i < k \wedge i' < k') \wedge (pos \neq pos')]\langle d|d'\rangle = [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle d|d'\rangle\varphi \\
\Rightarrow&[\varphi \wedge (i < k \wedge i' < k')]\langle d|d'\rangle = [\varphi \wedge (i < k \wedge i' < k')]\langle d|d'\rangle\varphi
\end{aligned}$$

The proof of this BiGKAT judgment relies on the proof of the two distinct cases, i.e. $pos = pos'$ and $pos \neq pos'$, and as a final step on the application of the Weak Whl rule (21). The details of the proof are in the appendix.

5 Related work

The GKAT system was introduced in [15], which also introduced its probabilistic model with sub-Markov kernels. It was investigated further in [14], which in particular provides a semantics for which the equational theory is complete.

Relational Hoare logic was introduced in [8]. Probabilistic relational Hoare logic (pRHL) is due to Barthe and coauthors in [6], 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 [8] 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, based on GKAT. This has in particular led us to introduce an additional axiom (14) to the theory of GKAT. We have illustrated the expressivity of our system, BiGKAT, by proving how probabilistic relational Hoare logic [6] (up to a restriction on the *while* rule) can be soundly interpreted in it. In future work we would like to explore if this soundness theorem 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 [4].

Acknowledgements The first and second authors were partially supported for this work 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, Thomas Espitau, Benjamin Grégoire, Justin Hsu, Léo Stefanesco, and Pierre-Yves Strub. Relational reasoning via probabilistic coupling. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 387–401. Springer, 2015. `doi:10.1007/978-3-662-48899-7\_27`.

[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, 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`.

[6] 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`.

[7] 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`.

[8] 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`.

[9] 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`.

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

[11] 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`.

[12] 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`.

[13] 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.

[14] 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.

[15] 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

$$
\begin{array}{rcll}
c_1 +_{e_1} (c_2 +_{e_2} c_3) & = & (c_1 +_{e_1} c_2) +_{e_1+e_2} c_3 & (29) \\
c_1 +_e c_2 & = & c_1 +_e \neg\, e \cdot c_2 & (30) \\
e_1 \cdot (c_1 +_{e_2} c_2) & = & e_1 \cdot c_1 +_{e_2} e_1 \cdot c_2 & (31) \\
c +_e 0 & = & e \cdot c & (32) \\
c_1 +_0 c_2 & = & c_2 & (33) \\
e \cdot (c_1 +_e c_2) & = & ec_1 & (34)
\end{array}
\qquad
\begin{array}{rcll}
c^{(e)} & = & c^{(e)} \cdot \neg\, e & (35) \\
c^{(e)} & = & (e \cdot c)^{(e)} & (36) \\
c^{(0)} & = & 1 & (37) \\
c^{(1)} & = & 0 & (38) \\
e_1^{(e_2)} & = & \neg\, e_2 & (39) \\
c^{(e_2)} & = & c^{(e_1 e_2)} \cdot c^{(e_2)} & (40)
\end{array}
$$

Figure 4: Derivable GKAT facts

Equational theory for effects

The equational theory of GKAT that we will resort on ads to the base theory the following additional axioms:

$$x_1 \leftarrow t_1 \cdot x_2 \leftarrow t_2 = \begin{cases} x_2 \leftarrow t_2[t_1/x_1] \cdot x_1 \leftarrow t_1 & \text{if } x_1 \neq x_2 \text{ and } x_2 \notin FV(t_1) \\ x_1 \leftarrow t_2[t_1/x_1] & \text{if } x_1 = x_2 \end{cases} \tag{41}$$

$$x_1 \overset{\$}{\leftarrow} d_1 \cdot x_2 \overset{\$}{\leftarrow} d_2 = \begin{cases} x_2 \overset{\$}{\leftarrow} d_2 \cdot x_1 \overset{\$}{\leftarrow} d_1 & \text{if } x_1 \neq x_2 \\ x_1 \overset{\$}{\leftarrow} d_2 & \text{if } x_1 = x_2 \end{cases} \tag{42}$$

$$x_1 \leftarrow t \cdot x_2 \overset{\$}{\leftarrow} d = \begin{cases} x_2 \overset{\$}{\leftarrow} d \cdot x_1 \leftarrow t & \text{if } x_1 \neq x_2 \\ x_1 \overset{\$}{\leftarrow} d & \text{if } x_1 = x_2 \end{cases} \tag{43}$$

$$x_1 \overset{\$}{\leftarrow} d \cdot x_2 \leftarrow t = x_1 \leftarrow t \quad \text{if } x_1 = x_2 \tag{44}$$

Proposition .1. *The axioms (41)-(44) are valid in the Probabilistic model of Definition 2.1.*

Proof.

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,

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

□

"One-sided" probabilistic relational Hoare logic rules.

- *R-Assign left rule*:

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

- *R-Cond left rule*:

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

- *R-Cond right rule*:

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

Encodings of the rules above in BiGKAT.

- *R-Assign left rule*:

$$\varphi[v/x]\langle x \leftarrow v|\mathbf{skip}\rangle = \varphi[v/x]\langle x \leftarrow v|\mathbf{skip}\rangle\varphi \tag{45}$$

- *R-Cond-left rule*:

$$\begin{aligned}
& \phi \cdot \langle e| \cdot \langle c_1|c_1'\rangle = \phi \cdot \langle e| \cdot \langle c_1|c_1'\rangle \cdot \psi \quad \wedge \quad \phi \cdot \ \langle \neg\, e| \cdot \langle c_2|c_1'\rangle = \phi \cdot \ \langle \neg\, e| \cdot \ \langle c_2|c_1'\rangle \cdot \psi \\
& \Rightarrow \ \phi \cdot \langle c_1 +_e c_2|c_1\rangle = \phi \cdot \langle c_1 +_e c_2|c_1'\rangle \cdot \psi
\end{aligned} \tag{46}$$

- *R-Cond-right rule*:

$$\begin{aligned}
& \phi \cdot |e\rangle \cdot \langle c_1|c_1'\rangle = \phi \cdot |e\rangle \cdot \langle c_1|c_1'\rangle \cdot \psi \quad \wedge \quad \phi \cdot \ |\neg\, e\rangle \cdot \langle c_1|c_2'\rangle = \phi \cdot \ |\neg\, e\rangle \cdot \ \langle c_1|c_2'\rangle \cdot \psi \\
& \Rightarrow \ \phi \cdot \langle c_1|c_1 +_e c_2\rangle = \phi \cdot \langle c_1|c_1 +_e c_2\rangle \cdot \psi
\end{aligned} \tag{47}$$

Proof of Lemma 3.1.
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

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

Proof of Lemma 3.2.
To prove the equality, first note that

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

by axiom (14) and Boolean algebra.
Using this observation, we reason for (28)

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

Proof of Proposition 3.1.

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

Proof of Lemma 3.4.

$$
\begin{array}{ll}
 & \langle c^{(e)} | c'^{(e')} \rangle \\
= & \quad \{ \text{ (11)} \} \\
 & \langle cc^{(e)} +_{\langle e|} 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ } \langle \text{ } | \text{ is homomorphism} \} \\
 & (\langle cc^{(e)} | +_{\langle e|} \langle 1|)(|c'^{(e')}\rangle) \\
= & \quad \{ \text{ (5)} \} \\
 & \langle cc^{(e)} | c'^{(e')} \rangle +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (15)} \} \\
 & (|c'^{(e')}\rangle \cdot \langle cc^{(e)}|) +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (11)} \} \\
 & (|c'c'^{(e')}\rangle +_{|e'\rangle} |1\rangle)\langle cc^{(e)}| +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (5)} \} \\
 & (\langle cc^{(e)} | c'c'^{(e')} \rangle +_{|e'\rangle} \langle cc^{(e)} | 1 \rangle) +_{\langle e|} \langle 1 | c'^{(e')} \rangle \\
= & \quad \{ \text{ (3)} \} \\
 & \langle cc^{(e)} | c'c'^{(e')} \rangle +_{\langle e|e'\rangle} (\langle cc^{(e)} | 1 \rangle +_{\langle e|} \langle 1 | c'^{(e')} \rangle) \\
= & \quad \{ \text{ homomorphism} \} \\
 & \langle c|c'\rangle\langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e'\rangle} (\langle cc^{(e)} | 1 \rangle +_{\langle e|} \langle 1 | c'^{(e')} \rangle) \\
= & \quad \{ \text{ (4) and (2)} \} \\
 & \langle c|c'\rangle\langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e'\rangle} (\langle ecc^{(e)} | 1 \rangle +_{\langle e|} \langle \neg\, e | c'^{(e')} \rangle) \\
= & \quad \{ \text{ Lemma 3.1 and (4)} \} \\
 & \langle c|c'\rangle\langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e'\rangle} (\langle c^{(e)} | 1 \rangle +_{\langle e|} \langle \neg\, e | c'^{(e')} \rangle) \\
= & \quad \{ \text{ (2) and (4)} \} \\
 & \langle c|c'\rangle\langle c^{(e)} | c'^{(e')} \rangle +_{\langle e|e'\rangle} (\langle \neg\, e| + |\neg\, e'\rangle)(\langle c^{(e)} | 1 \rangle +_{\langle e|} \langle \neg\, e | c'^{(e')} \rangle)
\end{array}
$$

$$
\begin{aligned}
&= \quad \{ \text{ fact u5'} \} \\
&\quad \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) \} \\
&\quad \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.} \} \\
&\quad \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.} \} \\
&\quad \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) \} \\
&\quad \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} \} \\
&\quad \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 \{ \text{(2), (4), (fact u5') and B.A.} \} \\
&\quad \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')} \} \\
&\quad \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 \{ \text{(2) and (4)} \} \\
&\quad \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 (27).

Proof of Lemma 3.5.

$$
\begin{aligned}
&\quad \phi(\langle c^{(e)}|\neg\, e'\rangle +_{\langle e|} \langle \neg\, e|c'^{(e')}\rangle) \\
&= \quad \{ \text{homomorfism and (4)} \} \\
&\quad \phi(\langle e|\neg\, e'\rangle\langle c^{(e)}| +_{\langle e|} \langle \neg\, e|c'^{(e')}\rangle) \\
&= \quad \{ \text{fact u5'} \} \\
&\quad \phi\langle e|\neg\, e'\rangle\langle c^{(e)}| +_{\langle e|} \phi\langle \neg\, e|c'^{(e')}\rangle \\
&= \quad \{ \text{(16) and (7)} \} \\
&\quad 0 +_{\langle e|} \phi\langle \neg\, e|c'^{(e')}\rangle \\
&= \quad \{ \text{(2) and fact u6} \} \\
&\quad \langle \neg\, e|\phi\langle \neg\, e|c'^{(e')}\rangle \\
&= \quad \{ \text{B.A.} \} \\
&\quad \phi\langle \neg\, e|c'^{(e')}\rangle
\end{aligned}
$$

$$
\begin{array}{ll}
= & \{\ (11)\} \\
& \phi\langle \neg\ e | c'\, c'^{(e')} +_{e'} 1\rangle \\
= & \{\ (2),\ (4) \text{ and } (10)\} \\
& \phi\langle \neg\ e | e'\, c'\, c'^{(e')} +_{e'} (\neg\ e')\rangle \\
= & \{\ \text{fact u5' and homomorfism}\} \\
& \phi(\langle \neg\ e | e'\, c'\, c'^{(e')}\rangle +_{|e'\rangle} \langle \neg\ e | \neg\ e'\rangle) \\
= & \{\ \text{homomorfism and (fact u5')}\} \\
& \phi\langle \neg\ e | e'\rangle | c'\, c'^{(e')}\rangle +_{|e'\rangle} \phi\langle \neg\ e | \neg\ e'\rangle \\
= & \{\ (16)\} \\
& 0 +_{|e'\rangle} \phi\langle \neg\ e | \neg\ e'\rangle \\
= & \{\ (2) \text{ and fact u6}\} \\
& |\neg\ e'\rangle \phi\langle \neg\ e | \neg\ e'\rangle \\
= & \{\ \text{B.A.}\} \\
& \langle \neg\ e | \neg\ e'\rangle \phi
\end{array}
$$

Proof of Theorem 3.1.
R-Seq rule:

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

R-Cond rule:

$$
\begin{array}{ll}
& \phi \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle \\
= & \{\ \text{B.A. and (28)}\} \\
& \phi \cdot (\langle e | e'\rangle +_{e'} \langle \neg\ e | \neg\ e'\rangle) \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle \\
= & \{\ \text{(U5,U5')}\} \\
& \phi \cdot \langle e | e'\rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle +_{e'} \phi \cdot \langle \neg\ e | \neg\ e'\rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle \\
= & \{\ (24),\ (25)\} \\
& \phi \cdot \langle e | e'\rangle \cdot \langle c_1 | c_1'\rangle +_{e'} \phi \cdot \langle \neg\ e | \neg\ e'\rangle \cdot \langle c_2 | c_2'\rangle \\
= & \{\ \text{premises}\} \\
& \phi \cdot \langle e | e'\rangle \cdot \langle c_1 | c_1'\rangle \cdot \psi +_{e'} \phi \cdot \langle \neg\ e | \neg\ e'\rangle \cdot \langle c_2 | c_2'\rangle \cdot \psi \\
= & \{\ (24),\ (25)\} \\
& \phi \cdot \langle e | e'\rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle \cdot \psi +_{e'} \phi \cdot \langle \neg\ e | \neg\ e'\rangle \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle \cdot \psi \\
= & \{\ \text{(U5,U5')}\} \\
& \phi \cdot (\langle e | e'\rangle +_{e'} \langle \neg\ e | \neg\ e'\rangle) \cdot (\langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle) \cdot \psi \\
= & \{\ (28)\} \\
& \phi \cdot \langle c_1 +_e c_2 | c_1' +_{e'} c_2\rangle \cdot \psi
\end{array}
$$

R-Cond right rule:

$$
\begin{aligned}
& \phi\langle c_1|c_1' +_e c2'\rangle \\
= & \quad \{ \text{ (14)}\} \\
& \phi(|e\rangle +_{|e\rangle} |\neg\ e\rangle)\langle c_1|c_1' +_e c2'\rangle \\
= & \quad \{ \text{ (5)}\} \\
& \phi|e\rangle\langle c_1|c_1' +_e c2'\rangle +_{|e\rangle} \phi|\neg\ e\rangle\langle c_1|c_1' +_e c2'\rangle \\
= & \quad \{ \text{ (15) and (34)}\} \\
& \phi\langle c_1|ec_1'\rangle +_{|e\rangle} \phi\langle c_1|\neg\ ec_2'\rangle \\
= & \quad \{ \text{ premises}\} \\
& \phi\langle c_1|ec_1'\rangle\psi +_{|e\rangle} \phi\langle c_1|\neg\ ec_2'\rangle\psi \\
= & \quad \{ \text{ (15), (34), (5) and (14) reverse steps}\} \\
& \phi(|e\rangle +_{|e\rangle} |\neg\ e\rangle)\langle c_1|c_1' +_e c2'\rangle\psi \\
= & \quad \{ \text{ B.A.}\} \\
& \phi\langle c_1|c_1' +_e c2'\rangle\psi
\end{aligned}
$$

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

$$
\begin{aligned}
& \phi\langle c^{(e)}|c'^{e'}\rangle \\
= & \quad \{ \text{ Lemma 3.4 (27)}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}(\langle c^{(\langle e|)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(|e'\rangle)}\rangle) \\
= & \quad \{ \text{ premise and Lemma 3.3}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}\phi(\langle c^{(\langle e|)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(|e'\rangle)}\rangle) \\
= & \quad \{ \text{ Lemma 3.5 (28)}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}\langle\neg\ e|\neg\ e'\rangle\phi \\
= & \quad \{ \text{ B.A.}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}\langle\neg\ e|\neg\ e'\rangle\phi\phi \\
= & \quad \{ \text{ Lemma 3.5 (28) reverse direction}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}\phi(\langle c^{(\langle e|)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(|e'\rangle)}\rangle)\phi \\
= & \quad \{ \text{ Lemma 3.3 (reverse direction)}\} \\
& \phi\langle c|c'\rangle^{\langle e|e'\rangle}(\langle c^{(\langle e|)}|\neg\ e'\rangle +_{\langle e|} \langle\neg\ e|c'^{(|e'\rangle)}\rangle)\phi \\
= & \quad \{ \text{ Lemma 3.4 (27) reverse direction}\} \\
& \phi\langle c^{(e)}|c'^{(e')}\rangle\phi \\
= & \quad \{ \text{ (fact w4)}\} \\
& \phi\langle c^{(e)}\neg\ e|c'^{(e')}\neg\ e'\rangle\phi \\
= & \quad \{ \text{ Def. 3.1}\} \\
& \phi\langle c^{(e)}|c'^{(e')}\rangle\langle\neg\ e|\neg\ e'\rangle\phi
\end{aligned}
$$

R-Sub rule:

$$
\begin{aligned}
& \phi' \cdot \langle c | c' \rangle \\
= & \quad \{ \text{ ax (14)} \} \\
& \phi' \cdot \phi \cdot \langle c | c' \rangle \\
= & \quad \{ \text{ premise} \} \\
& \phi' \cdot \phi \cdot \langle c | c' \rangle \cdot \psi \\
= & \quad \{ \text{ ax (14)} \} \\
& \phi' \cdot \phi \cdot \langle c | c' \rangle \cdot \psi \cdot \psi' \\
= & \quad \{ \text{ premise} \} \\
& \phi' \cdot \phi \cdot \langle c | c' \rangle \cdot \psi' \\
= & \quad \{ \text{ ax (14)} \} \\
& \phi' \cdot \langle c | c' \rangle \cdot \psi'
\end{aligned}
$$

R-Case rule:

$$
\begin{aligned}
& \phi \cdot \langle c | c' \rangle \\
= & \quad \{ \text{ B.A., ax (14)} \} \\
& \phi \cdot (\phi' +_{\phi'} \phi') \cdot \langle c | c' \rangle \\
= & \quad \{ \text{ (U5), (U5') of GKAT} \} \\
& \phi \cdot \phi' \cdot \langle c | c' \rangle +_{\phi'} \phi \cdot \neg \phi' \cdot \langle c | c' \rangle \\
= & \quad \{ \text{ premises} \} \\
& \phi \cdot \phi' \cdot \langle c | c' \rangle \cdot \psi +_{\phi'} \phi \cdot \neg \phi' \cdot \langle c | c' \rangle \cdot \psi \\
= & \quad \{ \text{ (U5') of GKAT} \} \\
& \phi \cdot (\phi' \cdot \langle c | c' \rangle \cdot \psi +_{\phi'} \neg \phi' \cdot \langle c | c' \rangle \cdot \psi) \\
= & \quad \{ \text{ (U5) of GKAT} \} \\
& \phi \cdot ((\phi' +_{\phi'} \neg \phi') \cdot \langle c | c' \rangle \cdot \psi) \\
= & \quad \{ \text{ (B.A.), ax (14)} \} \\
& \phi \cdot \langle c | c' \rangle \cdot \psi
\end{aligned}
$$

Example 1.

$$
\begin{aligned}
& [y = y'] \langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x'=tt]} d_2') \cdot c_2' \rangle \qquad (48) \\
= & \quad \{ \ e +_e \neg \, e = 1 \} \\
& [y = y'] ([x = x'] + \neg \, [x = x']) \langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x'=tt]} d_2') \cdot c_2' \rangle \\
= & \quad \{ \ (31) \} \\
& [y = y'] [x = x'] \langle (d_1 +_{[x=tt]} d_2) \cdot c_2 | (d_1' +_{[x=tt]} d_2') \cdot c_2' \rangle \\
= & \quad \{ \ (5) \} \\
& [y = y'] [x = x'] \langle (d_1 \cdot c_2) +_{[x=tt]} (d_2 \cdot c_2) | (d_1' \cdot c_2') +_{[x=tt]} (d_2' \cdot c_2') \rangle
\end{aligned}
$$

- **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 $b \xleftarrow{\$} dmybool$. 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 (47).
By rule (20) 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:

$$\begin{aligned}
&[y = y']\langle d_1 \cdot c_2 | d_1' \cdot c_2'\rangle \\
= &\quad \{ \text{ abbreviation and homomorfism} \} \\
&[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool'\rangle\langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1\rangle \\
&\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b'\rangle \\
= &\quad \{ \text{ R-Rand} \} \\
&[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool'\rangle[y = y'][b = b'] \\
&bihomy \leftarrow y \; xor \; tt +_{[b=tt]} 1 y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1 bihomy \leftarrow y \; xor \; by' \leftarrow y' \; xor \; b' \\
= &\quad \{ (20) \} \\
&[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool'\rangle[y = y'][b = b'] \\
&\langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1\rangle[y = y'][b = b'] \\
&\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b'\rangle \\
= &\quad \{ \text{ R-Assign} \} \\
&[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool'\rangle[y = y'][b = b'] \\
&\langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1\rangle[y = y'][b = b'] \\
&\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b'\rangle[y = y'] \\
= &\quad \{ (19) \} \\
&[y = y']\langle b \xleftarrow{\$} dbool | b' \xleftarrow{\$} dbool'\rangle\langle y \leftarrow y \; xor \; tt +_{[b=tt]} 1 | y' \leftarrow y' \; xor \; tt +_{[b'=tt]} 1\rangle \\
&\langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b'\rangle[y = y'] \\
= &\quad \{ \text{ homomorfism and abbreviation} \} \\
&[y = y']\langle d_1 \cdot c_2 | d_1' \cdot c_2'\rangle[y = y']
\end{aligned}$$

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

- **subgoal (4)**:

$$
\begin{array}{ll}
 & [y = y'][x = tt][x' \neq tt]\langle d_2 \cdot c_2 | d_2' \cdot c_2' \rangle \\
= & \quad \{ \text{ Abbreviations} \} \\
 & [y = y'][x = tt][x' \neq tt]\langle b \leftarrow ff \cdot y \leftarrow y \; xor \; b | b' \leftarrow ff \cdot y' \leftarrow y' \; xor \; b' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
 & [y = y'][x = tt][x' \neq tt]\langle b \leftarrow ff | b' \leftarrow ff \rangle \langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle \\
= & \quad \{ \text{ R-Assign} \} \\
 & [y = y'][x = tt][x' \neq tt]\langle b \leftarrow ff | b' \leftarrow ff \rangle [y = y'][b = b'] \\
 & \langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y'] \\
= & \quad \{ (19) \} \\
 & [y = y'][x = tt][x' \neq tt]\langle b \leftarrow ff | b' \leftarrow ff \rangle \langle y \leftarrow y \; xor \; b | y' \leftarrow y' \; xor \; b' \rangle [y = y'] \\
= & \quad \{ \text{ homomorfism} \} \\
 & [y = y'][x = tt][x' \neq tt]\langle b \leftarrow ff \cdot y \leftarrow y \; xor \; b | b' \leftarrow ff \cdot y' \leftarrow y' \; xor \; b' \rangle [y = y']
\end{array}
$$

Example 2.
Regarding the invariant

$$\varphi \hat{=} ((pos \neq pos') \rightarrow (pos = i + \Sigma(H) \wedge pos' = i' - \Sigma(H')) \wedge (P(H) \rightarrow (pos = pos')) \wedge (i = i') \wedge (k = k')$$

we use the additional abbreviations $\psi_1 \hat{=} pos \neq pos' \rightarrow (pos = i + \Sigma(H) \wedge pos' = i' - \Sigma(H'))$ and $\psi_2 \hat{=} P(H) \rightarrow (pos = pos')$, to facilitate the writing of the proof.
The first step is to prove the two distinct cases $pos = pos'$ and $pos \neq pos'$:

- **1st case**:

$$
\begin{array}{ll}
 & [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle d | d' \rangle \\
= & \quad \{ \text{ homomorfism} \} \\
 & [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
 & \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \; +_b \; pos - -) | (pos' +\!\!+ \; +_{b'} \; pos' - -) \rangle \\
 & \langle i \leftarrow i + 1 | i' \leftarrow i' + 1 \rangle \\
= & \quad \{ (18) \} \\
 & [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle [b = b'] \\
 & \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \; +_b \; pos - -) | (pos' +\!\!+ \; +_{b'} \; pos' - -) \rangle \\
 & \langle i \leftarrow i + 1 | i' \leftarrow i' + 1 \rangle \\
= & \quad \{ (20) \text{ and } (19) \} \\
 & [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
 & \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \; +_b \; pos - -) | (pos' +\!\!+ \; +_{b'} \; pos' - -) \rangle \\
 & \langle i \leftarrow i + 1 | i' \leftarrow i' + 1 \rangle [pos = pos'] \\
= & \quad \{ \} \\
 & [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
 & \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \; +_b \; pos - -) | (pos' +\!\!+ \; +_{b'} \; pos' - -) \rangle \\
 & \langle i \leftarrow i + 1 | i' \leftarrow i' + 1 \rangle (\psi_1 \wedge \psi_2)(i = i' \wedge k = k') \\
= & \quad \{ \text{ notation} \} \\
 & [\varphi \wedge (i < k \wedge i' < k') \wedge (pos = pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
 & \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \; +_b \; pos - -) | (pos' +\!\!+ \; +_{b'} \; pos' - -) \rangle \\
 & \langle i \leftarrow i + 1 | i' \leftarrow i' + 1 \rangle \varphi
\end{array}
$$

- **2nd case**:

$$
\begin{aligned}
& [\varphi \wedge (i < k \wedge i' < k') \wedge (pos \neq pos')]\langle d | d' \rangle \\
= & \quad \{ \text{ homomophism} \} \\
& [\varphi \wedge (i < k \wedge i' < k') \wedge (pos \neq pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
& \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \ +_b \ \ pos --) | (pos' +\!\!+ \ +_{b'} \ \ pos' --) \rangle \\
& \langle i \leftarrow i+1 | i' \leftarrow i'+1 \rangle \\
= & \quad \{ \text{ (18)} \} \\
& [\varphi \wedge (i < k \wedge i' < k') \wedge (pos \neq pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
& [pos = start + \Sigma(H) pos' = start' + \Sigma(H) \wedge \neg\, P(H)][b \neq b'] \\
& \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \ +_b \ \ pos --) | (pos' +\!\!+ \ +_{b'} \ \ pos' --) \rangle \\
& \langle i \leftarrow i+1 | i' \leftarrow i'+1 \rangle \\
= & \quad \{ \ b \neq b' \} \\
& [\varphi \wedge (i < k \wedge i' < k') \wedge (pos \neq pos')]\langle b \xleftarrow{\$} \{0,1\} | b' \xleftarrow{\$} \{0,1\} \rangle \\
& [pos = start + \Sigma(H) pos' = start' + \Sigma(H) \wedge \neg\, P(H)][b = tt \wedge b' = ff \vee b = ff \wedge b' = tt] \\
& \langle H \leftarrow b :: H | H' \leftarrow b' :: H' \rangle \langle (pos +\!\!+ \ +_b \ \ pos --) | (pos' +\!\!+ \ +_{b'} \ \ pos' --) \rangle \\
& \langle i \leftarrow i+1 | i' \leftarrow i'+1 \rangle
\end{aligned}
$$

  Let us analyse the two following subcases:

  - case $[b = tt \wedge b' = ff]$: program on the left:

$$
\begin{aligned}
& [\neg\, P(H)][pos = start + \Sigma(H)][b = tt]\langle H \leftarrow b :: H | \langle pos +\!\!+ \ +_b \ \ pos --| \langle i \leftarrow i+1| \\
= & [\neg\, P(H)][pos = start + \Sigma(H)][b = tt]\langle H \leftarrow b :: H | \langle pos +\!\!+ \ +_b \ \ pos --| \langle i \leftarrow i+1| \\
& [pos = start + \Sigma(H) \wedge (\neg\, P(H)) \vee [\Sigma(H) = n]]
\end{aligned}
$$

    and analogously for the program on the right. We can obsevre that $\neg\, P(H) \Rightarrow \psi_2$ and $\big(\Sigma(H) = n \wedge start' - start = 2n \wedge pos = start + \Sigma(H) \wedge pos' = start' - \Sigma(H)\big) \Rightarrow \psi_2$ and by Boolean algebra

$$
\begin{aligned}
& [\neg\, P(H)] + [\Sigma(H) = n][start' - start = 2n][pos = start + \Sigma(H)][pos' = start' - \Sigma(H)] \\
= & [\neg\, P(H)]\psi_2 + [\Sigma(H) = n][start' - start = 2n][pos = start + \Sigma(H)][pos' = start' - \Sigma(H)]\psi_2 \\
= & \big([\neg\, P(H)] + [\Sigma(H) = n][start' - start = 2n][pos = start + \Sigma(H)][pos' = start' - \Sigma(H)]\big)\psi_2
\end{aligned}
$$

  - case $[b = ff \wedge b' = tt]$: symmetric

  Hence, we conclude $[\varphi \wedge (i < k \wedge i' < k')]\langle d | d' \rangle = [\varphi \wedge (i < k \wedge i' < k')]\langle d | d' \rangle \varphi$.
  The next step is to apply the Weak Whl rule (21), for which we observe that the three premises hold:

- first premise: $\varphi \Rightarrow [(i < k)] = [(i' < k')]$
- second premise: proven above
- third premise: expressions d and d' start with an action, hence $E(d) = E(d') = 0$

  Now we can apply the R-Weak Whl rule (21) and obtain
  $\varphi\langle d^{(i<k)} | d'^{(i'<k')} \rangle = \varphi\langle d^{(i<k)} | d'^{(i'<k')} \rangle \varphi[\neg\, (i < k)]$
and then we reason for programs c, c':

$$
\begin{aligned}
& [start' - start = 2n]\langle c | c' \rangle \\
= & \quad \{ \text{ (19)} \} \\
& [start' - start = 2n]\langle c | c' \rangle \varphi[\neg\, (i < k)] \\
= & \quad \{ \text{ arithmetic} \} \\
& [start' - start = 2n]\langle c | c' \rangle [P(H_1) \rightarrow (pos = pos')]
\end{aligned}
$$