On Kleene Algebras for weighted computation[⋆]

Leandro Gomes[1], and Alexandre Madeira[1,2]and Luís S. Barbosa[1]

[1] HASLab INESC TEC - Univ. Minho, Portugal
[2] CIDMA - Univ. Aveiro, Portugal

Abstract. Kleene algebra with tests (KAT) was introduced as an algebraic structure to model and reason about classic imperative programs, i.e. sequences of discrete actions guarded by Boolean tests.
This paper introduces two generalisations of this structure able to express programs as weighted transitions and tests with outcomes in a not necessary bivalent truth space, namely graded Kleene algebra with tests (GKAT) and Heyting Kleene algebra with tests (HKAT).
On these contexts, in analogy to Kozen's encoding of Propositional Hoare Logic (PHL) in KAT [10], we discuss the encoding of a graded PHL in HKAT and of its while-free fragment in GKAT.

1 Introduction

1.1 Roadmap

Kleene algebra is pervasive in computer science: it arises in relational algebra, semantics and logics of programs, automata and formal language theory, and design and analysis of algorithms. In the specific context of program calculi, the axiomatisation of Kleene algebra forms a purely equational system to manipulate programs [8]. Its applications typically deal with conventional, imperative programming constructs such as conditional and loops. In order to reason equationally about them, a notion of test is required, which lead D. Kozen to define - *Kleene algebra with tests* (KAT) [9], which plays a major role in reasoning about programs.

Hoare logic (HL) was the first formal system proposed for verification of programs. Introduced in as early as 1969, its wide influence transformed Hoare's work in a cornerstone of program correctness, a reference for most current research in the area. HL encompasses a syntax to reason about partial correctness assertions (PCA) of the form $\{b\}p\{c\}$, also called a Hoare triple, and a deductive system to reason about them [5]. In a PCA, b and c stand for predicates, representing pre and post conditions, respectively, and p is a program statement.

In particular, *propositional Hoare logic* (PHL) can be seen as a fragment of HL, in which PCAs are reduced to static assertions about the underlying domain of computation [10]. In [10] the authors show that this fragment can be encoded

[⋆] This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within projects POCI-01-0145-FEDER-016692 and UID/MAT/04106/2013. The second author is also supported by the individual grant SFRH/BPD/103004/2014.

in a Kleene algebra with tests. The translation is based on equational logic, transforming PCAs into equations and the rules of inference into equational implications.

As originally presented, KAT is suitable to reason about classic imperative programs. In fact, such programs are particularly "well tractable": they represent a sequence of discrete steps, which can be modelled as atomic transition systems in a standard automaton. Moreover, the assertions about theses programs have an outcome in a bivalent truth space. However, current complex dynamic systems are based in new computing domains, namely probabilistic [15] or continuous [13], which entail the need for computing paradigms able to deal with quantitative program executions (eg. weighted, valued, probabilistic). Moreover, the assertions about these programs can have a graded outcome. In this context, the development of algebraic structures to model weighted computations becomes a must. This work builds on such motivations to introduce two generalisations of KAT able to express programs as weighted computations and tests as predicates evaluated in graded truth space - the *graded Kleene algebra with tests* (GKAT) and the *Heyting Kleene algebra with tests* (HKAT). GKAT, for example, has a myriad of interesting examples, from continuous Łukasiewicz lattice to the discrete finite hoops. HKAT, on the other hand, allows to address full imperative languages.

In analogy to KAT [10], we intend to encode PHL into GKAT, in the context of a research agenda to extend the classical area of program correctness. However, we can only partially generalise such an encoding. More specifically, we can only encode *while*-free programs. To achieve a complete encoding of Hoare logic, we propose to refine the basic structure, obtaining HKAT as a generalisation of the classical KAT. HKAT is, indeed, a subclass of GKAT. As a consequence, however, its set of examples is smaller. It includes, in particular, the lattice **3** to deal with partial programs and uncertainty on tests, and Gödel algebra, a well-known basic structure used in logics whose truth values are closed subsets of the interval $[0,1]$.

The remaining of the paper is organised as follows: Subsection 1.2 recaps some fundamental concepts needed to understand the definitions and results presented in this work. Section 2 introduces graded Kleene algebra with tests as a generalisation of KAT, including its axiomatisation, a few examples and proofs of basic properties. It also presents a partial encoding of classical PHL in GKAT. Section 3 introduces Heyting Kleene algebra with tests as another generalisation of the standard KAT and a refinement of GKAT, enjoying of a complete encoding of PHL. Section 4 sums up some related research, concludes, and enumerates some topics for future work.

1.2 Preliminaries

Definition 1. *A* Kleene algebra with tests *(KAT) is a tuple*

$$(K, T, +, ;, ^*, ^-, 0, 1)$$

where $T \subseteq K$, 0 and 1 are constants, + and ; are binary operators in K and T, * is a unary operator in K, and $^-$ is a unary operator defined only on T such that:

- $(K, +, ;, ^*, 0, 1)$ *is a Kleene algebra;*
- $(T, +, ;, ^-, 0, 1)$ *is a Boolean algebra;*
- $(T, +, ;, 0, 1)$ *is a subalgebra of* $(K, +, ;, 0, 1)$*.*

The elements of K, denoted by lower case letters p, q, r, s, x, y, z, stand for programs and the elements of T, denoted by a, b, c, d are called tests. Kleene algebra with tests induces an abstract programming language, where conditionals and **while** loops programming constructs are encoded as follows:

$$\textbf{if } b \textbf{ then } p \overset{\text{def}}{=} b; p + \bar{b}$$
$$\textbf{if } b \textbf{ then } p \textbf{ else } q \overset{\text{def}}{=} b; p + \bar{b}; q$$
$$\textbf{while } b \textbf{ do } p \overset{\text{def}}{=} (b; p)^*; \bar{b}$$

As stated in Section 1, Hoare logic allows to verify imperative programs by validating PCAs of the form $\{b\}p\{c\}$ through a deductive system [5]. The validity of a Hoare triple assures that whenever precondition b is met, after the execution of program p, if and when p halts, the postcondition c is guaranteed to hold. The set of logical rules is shown in Figure 1. Although the importance of HL for reasoning about program correctness is unquestionable, proofs in PHL can be more easily done in terms of purely equational calculation on KAT, as presented in [10]. In fact, it is shown that PHL can be encoded in KAT, in such a way that the inference rules *Composition*, *Conditional*, *While* and *Weakening* become derived theorems of KAT.

- *Composition rule*:

$$\frac{\{b\}p\{c\} \quad \{c\}q\{d\}}{\{b\}p; q\{d\}}$$

- *Conditional rule*:

$$\frac{\{b \wedge c\}p\{d\}, \{\neg b \wedge c\}q\{d\}}{\{c\} \textbf{ if } b \textbf{ then } p \textbf{ else } q \ \{d\}}$$

- *While rule*:

$$\frac{\{b \wedge c\}p\{c\}}{\{c\} \textbf{ while } b \textbf{ do } p\{\neg b \wedge c\}}$$

- *Weakening rule*:

$$\frac{b' \to b, \ \{b\}p\{c\}, \ c \to c'}{\{b'\} \ p\{c'\}}$$

Fig. 1. Hoare logic rules

As presented in [10], the PCA $\{b\}p\{c\}$ can be encoded in KAT as $b; p; \bar{c} = 0$, which is equivalent to $b; p = b; p; c$. The first equation means, intuitively, that the execution of p with precondition b and postcondition $\bar{c}$ does not halt. Equation $b; p = b; p; c$, on the other hand, states that the verification of the post condition c after the execution of $b; p$ is redundant.

Moreover, the inference rules of Hoare logic can be encoded in KAT, as shown below:

- *Composition*:

$$b; p = b; p; c \wedge c; q = c; q; d \Rightarrow b; p; q = b; p; q; d$$

- *Conditional*:

$$b; c; p = b; c; p; d \wedge \bar{b}; c; q = \bar{b}; c; q; d \Rightarrow c; (b; p + \bar{b}; q) = c; (b; p + \bar{b}; q); d$$

- *While*:

$$b; c; p = b; c; p; c \Rightarrow c; (b; p)^*; \bar{b} = c; (b; p)^*; \bar{b}; \bar{b}; c$$

- *Weakening*:

$$b' \leq b \wedge b; p = b; p; c \wedge c \leq c' \Rightarrow b'; p = b'; p; c'$$

2 Graded Kleene Algebra with Tests

2.1 The basic structure

The approach proposed here, to reason about program executions in a many-valued context, is based on redefining the interpretation of the assertions about programs. Since such assertions take the form of tests, we start by modifying the part of the axiomatisation of KAT that deals with properties of tests, i.e., the Boolean algebra $(T, +, \cdot, ^-, 0, 1)$.

Instead of having a Boolean outcome, as happens in KAT, tests are graded, taking values from a truth space with more than two possible outcomes (0 and 1). As a consequence, the expression $b; p$ represents a weighted execution of program p, conditioned by the value of b. In order to reason about computations in this graded setting, we introduce the following generalisation of KAT:

Definition 2. *A* graded Kleene algebra with tests *(GKAT) is a tuple*

$$(K, T, +, ;, ^*, \rightarrow, 0, 1)$$

where K and T are sets, with $T \subseteq K$, 0 and 1 are constants and $+$ and $;$ are binary operations in K and T, * is a unary operator in K, and $\rightarrow$ is an operator only defined in T, satisfying the axioms enumerated in Figure 2, where relation $\leq$ is induced by $+$ in the usual way: $p \leq q$ iff $p + q = q$. Note that $(T, +, ;, 0, 1)$ is a subalgebra of $(K, +, ;, 0, 1)$.

Again, programs are denoted by lower case letters p, q, r, s, x, y, z and tests by a, b, c, d. Note that, differently from what happens in KAT, negation is not explicitly denoted, although it can be derived as $a \rightarrow 0$, for $a \in T$. Indeed, this operator, along with a more relaxed subalgebra (which will replace the Boolean subalgebra of KAT) are introduced to support a proper truth space, for possible non bivalent interpretation of assertions.

Some operators in GKAT play a different role when acting on programs or tests. Such is the case of "+" and ";". The former one plays the role of

$$p + (q + r) = (p + q) + r \tag{1}$$
$$p + q = q + p \tag{2}$$
$$p + p = p \tag{3}$$
$$p + 0 = 0 + p = p \tag{4}$$
$$p; (q; r) = (p; q); r \tag{5}$$
$$p; 1 = 1; p = p \tag{6}$$
$$p; (q + r) = (p; q) + (p; r) \tag{7}$$
$$(p + q); r = (p; r) + (q; r) \tag{8}$$
$$p; 0 = 0; p = 0 \tag{9}$$
$$1 + p; p^* = p^* \tag{10}$$
$$1 + p^*; p \;=\; p^* \tag{11}$$
$$q + p; r \leq r \Rightarrow p^*; q \leq r \tag{12}$$
$$q + r; p \leq r \Rightarrow q; p^* \leq r \tag{13}$$
$$a; b \leq c \Leftrightarrow b \leq a \rightarrow c \tag{14}$$
$$a \rightarrow b \;\leq\; a \rightarrow (b + c) \tag{15}$$
$$b \;\leq\; a \rightarrow (a; b) \tag{16}$$
$$a + 1 \;=\; 1 + a = 1 \tag{17}$$
$$a; b \;=\; b; a \tag{18}$$
$$a + (a; b) \;=\; a \tag{19}$$

Fig. 2. Axiomatisation of graded Kleene algebra with tests

non-deterministic choice, when interpreting programs, and of logical disjunction, when acting on tests. The latter is taken as sequential composition of actions when applied to elements of K, and as a conjunction when applied to elements of T. In the domain of programs, the constants 0 and 1 refer to *halt* and *skip*, while when applied on tests, stand for false and true, respectively. However, there are operations specific to just one of these domains. For instance, while operation $*$ is taken as iterative execution of programs, operation $\rightarrow$ plays the role of logical implication over tests. Let us now discuss some instances of GKAT.

Example 1. (**2** - the Boolean lattice). Our first example is the well known binary structure

$$\mathbf{2} = (\{\top, \bot\}, \{\top, \bot\}, \vee, \wedge, ^*, \rightarrow, \bot, \top)$$

with the standard interpretation of Boolean connectives. Operator * maps each element of $\{\top, \bot\}$ to $\top$ and $\rightarrow$ is defined as logical implication.

Example 2. A second example is provided by the three-element linear lattice, which introduces an explicit denotation for "unknown" (or undefined).

$$\mathbf{3} = (\{\top, u, \bot\}, \{\top, u, \bot\}, \vee, \wedge, ^*, \rightarrow, \bot, \top)$$

where

$\vee$	$\bot$	u	$\top$
$\bot$	$\bot$	u	$\top$
u	u	u	$\top$
$\top$	$\top$	$\top$	$\top$

$\wedge$	$\bot$	u	$\top$
$\bot$	$\bot$	$\bot$	$\bot$
u	$\bot$	u	u
$\top$	$\bot$	u	$\top$

$\rightarrow$	$\bot$	u	$\top$
$\bot$	$\top$	$\top$	$\top$
u	$\bot$	$\top$	$\top$
$\top$	$\bot$	u	$\top$

*	
$\bot$	$\top$
u	$\top$
$\top$	$\top$

Example 3. For a fixed, finite set A, let us consider the structure

$$\mathbf{2^A} = (P(A), P(A), \cup, \cap, ^*, \rightarrow, \emptyset, A)$$

where $P(A)$ denotes the powerset of A, $\cup$ and $\cap$ are set union and intersection, respectively, * maps each set $X \in P(A)$ into A and $X \rightarrow Y = X^C \cup Y$, where $X^C = \{x \in A | x \notin X\}$.

Example 4. This example is based on the well-known Łukasiewicz arithmetic lattice.

$$\text{Ł} = ([0,1], [0,1], max, \odot, ^*, \rightarrow, 0, 1)$$

where $x \rightarrow y = min\{1, 1 - x + y\}$, $x \odot y = max\{0, x + y - 1\}$ and $*$ maps each point of the interval $[0,1]$ to 1.

Example 5. Let us consider now the example given by the standard Π-algebra

$$\boldsymbol{\Pi} = ([0,1], [0,1], max, ., ^*, \rightarrow, 0, 1)$$

where . is the usual multiplication for real numbers and

$$x \rightarrow y = \begin{cases} 1, & \text{if } x \leq y \\ y/x, & \text{if } y \leq x \end{cases}$$

with / being the usual division for real numbers and $*$ mapping each point of the interval $[0,1]$ to 1.

Example 6. Another example is provided by the Gödel algebra

$$G = ([0,1], [0,1], max, min, ^*, \rightarrow, 0, 1)$$

where

$$x \rightarrow y = \begin{cases} 1, & \text{if } x \leq y \\ y, & \text{if } y \leq x \end{cases}$$

and $*$ maps each point of the interval $[0,1]$ to 1.

Example 7. We consider now a GKAT endowing the finite *Wajsberg hoop* with a star operator, as presented in [1]. For a fixed natural k and a generator a, define the structure

$$\boldsymbol{W_k} = (W_k, W_k, +, ;, ^*, \rightarrow, 0, 1)$$

where $W_k = \{a^0, a^1, ..., a^{k-1}\}$, $1 = a^0$ and $0 = a^{k-1}$. Moreover, for any $m, n \leq k - 1, a^m + a^n = a^{min\{m,n\}}, a^m; a^n = a^{min\{m+n,k-1\}}, (a^m)^* = a^0$ and $a^m \rightarrow a^n = a^{max\{n-m,0\}}$.

Example 8. The $(min, +)$ Kleene algebra of [6], known as the tropical semiring, can be extended to a GKAT by adding residuation $\rightarrow$. First let R_+ denote the set $\{x \in \mathbb{R} | x \geq 0\}$ and let ∞ be a new element. Thus, define

$$\boldsymbol{R} = (R_+ \cup \{\infty\}, R_+ \cup \{\infty\}, min, +, ^*, \rightarrow, \infty, 0)$$

where, for any $x, y \in R_+ \cup \{\infty\}$, $x^* = 0$ and $x \rightarrow y = max\{y - x, 0\}$.

Note that in all examples considered, $T = K$, that is, the set of tests and the set of programs coincide.

For the purpose of this work, i.e., for reasoning about graded computations and assertions in a multi-valued truth space, Example 4 is particularly relevant. Indeed, this is a very well known model for fuzzy and multi-valued logics.

A main particularity of the GKAT axiomatization concerns rules (17)-(19), which form a weakened version of the axiomatization of a Boolean algebra. Note, however, that this is, in fact, a generalisation:

Lemma 1. *Any KAT is a GKAT.*

Proof. For a fixed KAT

$$\boldsymbol{A} = (K, T, +, ;, ^*, ^-, 0, 1)$$

let us consider the structure

$$\boldsymbol{M} = (K, T, +, ;, ^*, \rightarrow, 0, 1)$$

inheriting the operators $+$, $;$, $*$ and constants 0 and 1 from $\boldsymbol{A}$. Define $a \rightarrow 0 := \bar{a}$ and $a \rightarrow b := \bar{a} + b$, for $a, b \in T$.
Actually, axioms (14)-(16) hold for $\boldsymbol{M}$, for all $a, b, c \in T$. For (14), assume $a; b \leq c$, i.e. by definition of $\leq$, $a; b + c = c$. Then,

$$\begin{array}{l|l}
\quad a \rightarrow c & \\
= \qquad \{\text{ definition of } \rightarrow \text{ and hypothesis}\} & = \qquad \{\text{ (1), BA: } a + \bar{a} = 1\text{, (17) and (6)}\} \\
\quad \bar{a} + ((a; b) + c) & \quad \bar{a} + b + c \\
= \qquad \{\text{ BA } (+, ;)\text{-dist.}\} & = \qquad \{\text{ (2) and definition of } \rightarrow\} \\
\quad (\bar{a} + (a + c)); (\bar{a} + (b + c)) & \quad a \rightarrow c + b
\end{array}$$

Thus, $a; b \leq c \Rightarrow b \leq a \rightarrow c$. Now, assume $b \leq a \rightarrow c$, i.e. by definition of $\boldsymbol{M}$, $b; (\bar{a} + c) = b$, and reason

$$\begin{array}{l|l}
\quad a; b + c & \\
= \qquad \{\ b; (\bar{a} + c) = b\} & = \qquad \{\text{ BA comm, } a; \bar{a} = 0\text{, (9), (4) and (8)}\} \\
\quad a; (b; (\bar{a} + c)) + c & \quad (a; b + 1); c \\
= \qquad \{\text{ BA } (+, ;)\text{-dist.}\} & = \qquad \{\text{ (17) and (6)}\} \\
\quad a; b; \bar{a} + a; b; c + c & \quad c
\end{array}$$

Hence, $b \leq c \rightarrow c \Rightarrow a;b \leq c$. To prove (15), consider

$$\begin{aligned} & a \rightarrow b + a \rightarrow (b+c) \\ = & \quad \{ \text{ by definition of } \rightarrow \} \\ & (\bar{a}+b) + (\bar{a} + (b+c)) \\ = & \quad \{ \text{ (1), (2) and (3)} \} \\ & \bar{a} + (b+c) \\ = & \quad \{ \text{ by definition of } \rightarrow \} \\ & a \rightarrow (b+c) \end{aligned}$$

Axiom (16) is proved as follows:

$$\begin{aligned} & a \rightarrow (a;b) \\ = & \quad \{ \text{ definition of } \rightarrow \} \\ & \bar{a} + (a;b) \\ = & \quad \{ \text{ BA } (+,;)\text{-dist., BA: } a + \bar{a} = 1 \text{ and (6)} \} \\ & \bar{a} + b \\ = & \quad \{ \text{ definition of } \rightarrow \} \\ & a \rightarrow b \end{aligned}$$

We have that $b \leq a \rightarrow b$, so, by transitivity, $b \leq a \rightarrow (a;b)$, for all $a, b \in T$.

We concluded the proof that axioms (14)-(16) hold for any $a, b, c \in T$ in $\boldsymbol{M}$. Since axioms (1)-(13), (17)-(19) are axioms of $\boldsymbol{A}$, $\boldsymbol{M}$ is, indeed, a GKAT.

□

As stated above, while tests in KAT have an outcome of two possible values (0 and 1), GKAT deals with graded tests. This entails the need to weaken the Boolean subalgebra $(T, +, \cdot, ^*, 0, 1, ^-)$ of KAT. In any GKAT, for any test $a \in T$, $a;(a \rightarrow 0) = 0$, which follows immediately from definition of $\leq$ and axiom (14). However, it is not necessarily true that $a + (a \rightarrow 0) = 1$. In order to show this, let us consider the following GKAT structure over the set $\{0, n, m, 1\}$, where $\{0, m, 1\} \subseteq T$ and $n \in K$, in which the operation * maps all points to the top element of T, 1, and the remaining operations are defined as follows:

$+$	0	n	m	1
0	0	n	m	1
n	n	n	m	1
m	m	m	m	1
1	1	1	1	1

$;$	0	n	m	1
0	0	0	0	0
n	0	0	0	n
m	0	0	0	m
1	0	n	m	1

$\rightarrow$	0	n	m	1
0	1	1	1	1
n	m	1	1	1
m	m	m	1	1
1	0	n	m	1

In this structure, and considering $a = m$, we have $m + (m \rightarrow 0) = m + m = m \neq 1$. Recall that in KAT, a program execution is guarded by a test with only two possible outcomes: 0 or 1. Thus, an expression $b;p$ intuitively means that program p is executed when $b = 1$ and is not executed when $b = 0$.

It is therefore safe to state that GKAT has embedded a weakened Boolean subalgebra and, consequently, tests can assume other values besides 0 and 1, representing the truth degree of the statement "b is true". Consequently, the expression $b;p$ means that the execution of program p is guarded by that particular truth (graded) value.

2.2 Graded Propositional Hoare Logic

Kleene algebra with tests provides a theoretical basis to reason about classic imperative programs by using purely equational reasoning. Actually, its presentation in [10] aimed at the reduction of PHL to ordinary equations and quasi-equations, as mentioned in the introduction. In particular, the inference rules of Hoare logic are derived as theorems in KAT.

Following an analogous approach [10], mentioned in Subsection 1.2, we now encode propositional Hoare logic in GKAT. Since this new structure deals with graded tests, both the meaning of PCAs and the inference rules need to be adjusted. This reinterpretation unfolds a generalised version of classic Hoare logic, that we call here *graded propositional Hoare logic* (GPHL).

In the presence of graded tests, the interpretation of a triple $\{b\}p\{c\}$, and hence, the correctness of a program, relies on the idea that whenever $b;p$ executes with a truth degree b, if and when it halts, it is guaranteed that $(b;p);c$ holds with at least the same degree of truth. By other words, correctness of a program can only grow with execution. Therefore, the encoding in GKAT is captured by the following inequality:

$$b;p \leq b;p;c$$

However, the equivalence

$$b;p \leq b;p;c \Leftrightarrow b;p = b;p;c, \tag{20}$$

also holds in GKAT, following immediately from (7), (17) and (6). Note, also, that the equivalence

$$b;p = b;p;c \Leftrightarrow b;p \leq p;c$$

does not hold in GKAT, as it does in KAT.

The inference rules of Hoare logic can also be encoded in GKAT, as presented in the following theorem.

Theorem 1. *The following equational implications are theorems in GKAT.*

1. *Composition rule:*
   $b;p \leq b;p;c \wedge c;q \leq c;q;d \Rightarrow b;p;q = b;p;q;d$

2. *Conditional rule:*
   $b;c;p \leq b;c;p;d \wedge (b \rightarrow 0);c;q \leq (b \rightarrow 0);c;q;d$
   $\Rightarrow c;(b;p + (b \rightarrow 0);q) \leq c;(b;p + (b \rightarrow 0);q);d$

3. *Weakening rule:*
   $b' \leq b \wedge b;p \leq b;p;c \wedge c \leq c' \Rightarrow b';p \leq b';p;c'$

Proof. **1.** Let us assume that $b;p \leq b;p;c$ and $c;q \leq c;q;d$. By (20), these inequalities are equivalent to $b;p = b;p;c$ and $c;q = c;q;d$, respectively. So, we have

$$\begin{array}{ll|ll}
 & b;p;q & = & \{\ c;q = c;q;d\} \\
= & \{\ b;p = b;p;c\} & & b;p;c;q;d \\
 & b;p;c;q & = & \{\ b;p = b;p;c\} \\
 & & & b;p;q;d
\end{array}$$

2. Assume $b;c;p \leq b;c;p;d$ and $(b \to 0);c;q \leq (b \to 0);c;q;d$.
First of all, observe that, for any $p, q, r, s \in K$

$$p \leq q \;\&\; r \leq s \Rightarrow p + r \leq q + s \tag{21}$$

To prove this, assume that $p \leq q$ and $r \leq s$, i.e. $p + q = q$ and $r + s = s$. Then, by (1) and (2), $(p + r) + (q + s) = (p + q) + (r + s) = q + s$. So, by (21),

$$\begin{array}{ll}
 & b;c;p + (b \to 0);c;q \leq b;c;p;d + (b \to 0);c;q;d. \\
\Leftrightarrow & \qquad \{\ (18),\ (7) \text{ and } (8)\} \\
 & c;(b;p + (b \to 0);q) \leq c;(b;p + (b \to 0);q);d
\end{array}$$

3. Finally, for the Weakening rule, observe that, for all $b, c \in T$ and $p \in K$,

$$b;p \leq b;p;c \Rightarrow b;p;(c \to 0) \leq 0 \tag{22}$$

Using (20) to rewrite (22) as

$$b;p = b;p;c \Rightarrow b;p;(c \to 0) = 0 \tag{23}$$

and, assuming $b;p = b;p;c$, we have

$$\begin{array}{ll}
 & b;p;(c \to 0) \\
= & \qquad \{\ b;p = b;p;c \text{ assumption}\} \\
 & b;p;c;(c \to 0) \\
= & \qquad \{\ a;(a \to 0) = 0) \text{ and } (9)\} \\
 & 0
\end{array}$$

Using (23), the Weakening rule can be rewritten as

$$a \leq b \wedge b;p;(c \to 0) = 0 \wedge (d \to 0) \leq (c \to 0) \Rightarrow a;p;(d \to 0) = 0$$

which follows from the monotonicity of ";".

□

The attentive reader certainly noticed the absence of a While rule in the graded setting. In analogy with what was done before, such a rule would take the form:

$$b;c;p \leq b;c;p;c \Rightarrow c;(b;p)^*;(b \to 0) \leq c;(b;p)^*;(b \to 0);(b \to 0);c \tag{24}$$

However, this is not necessarily true for all $p \in K$ and $b, c \in T$.

To see this, consider the following GKAT structure over the set $\{0, n, m, 1\}$, in which the operator * maps all points to the top element 1 and the remaining operators are defined as follows:

$+$	0	n	m	1
0	0	n	m	1
n	n	n	m	1
m	m	m	m	1
1	1	1	1	1

$;$	0	n	m	1
0	0	0	0	0
n	0	0	0	n
m	0	0	0	m
1	0	n	m	1

$\rightarrow$	0	n	m	1
0	1	1	1	1
n	m	1	1	1
m	m	m	1	1
1	0	n	m	1

In this structure, $\{0, m, 1\} \subseteq T$ and $n \in K$. If $b = 0, c = m, p = 0$, the instantiation of $b; c; p \leq b; c; p; c$ becomes, using axioms (9) and (3),
$0; m; 0 + 0; m; 0; m = 0; m; 0; m \Leftrightarrow 0 = 0$
and that of $c; (b; p)^*; (b \rightarrow 0) \leq c; (b; p)^*; (b \rightarrow 0); (b \rightarrow 0); c$ becomes, by axioms (9), (10), (6) and (4),
$m; (0)^*; 1 + m; (0)^*; 1; 1; m = m; (0)^*; 1; 1; m \Leftrightarrow m = 0.$
Using these two equations, the equational implication which could represent the While rule (24) boils down to $0 = 0 \Rightarrow m = 0$, which is obviously false. In the next section we will discuss this problem, by presenting an alternative algebraic structure with complete Hoare logic encoding.

3 Heyting Kleene Algebra with Tests

3.1 The basic structure

By carefully looking at the while rule proof for the Hoare logic encoding in KAT it is easy to note that one cause for the failure of the analogous encoding in GKAT, mentioned in the previous section, is the impossibility of duplicating graded tests. Actually, in GKAT, we don't have that $b; b = b$, but only $b; b \leq b$. In fact, the duplication is a requirement for the proof of the While rule. The solution we propose here is to refine the GKAT structure with some additional properties, capturing two crucial aspects for the purpose of this work: allowing for a complete encoding of Hoare logic and, at the same time, capturing non-classical examples, with degrees of uncertainty in program executions and evaluation of tests. The idea is to use a Heyting algebra to model the tests, instead of the Boolean algebra implicit on KAT.

Definition 3. *A* Heyting Kleene algebra with tests *(HKAT) is a tuple*

$$(K, T, +, ;, ^*, \rightarrow, 0, 1)$$

where K and T are sets, with $T \subseteq K$, 0 and 1 are constants and $+$ and $;$ are binary operations in K and T, * is a unary operator in K, and $\rightarrow$ is an operator only defined in T, satisfying the axioms enumerated in Figure 2 plus three axioms from KAT, listed in Figure 3. The relation $\leq$ is induced by $+$ in the usual way $p \leq q$ iff $p + q = q$ such that:

- $(K, +, ;, ^*, 0, 1)$ *is a Kleene algebra;*
- $(T, +, ;, \rightarrow, 0, 1)$ *is a Heyting algebra;*
- $(T, +, ;, 0, 1)$ *is a subalgebra of* $(K, +, ;, 0, 1)$.

$$a; a = a \tag{25}$$

$$a; (a + b) = a \tag{26}$$

$$a + (b; c) = (a + b); (a + c) \tag{27}$$

Fig. 3. New axioms added to the axiomatisation of GKAT, to form the axiomatisation of Heyting Kleene algebra with tests

Note that, as in GKAT, negation is not explicitly denoted and can be derived as $a \rightarrow 0$.

Let us also enhance that HKAT is a subclass of GKAT. Examples 1, 2, 3 and 6 illustrate this structure. The set of examples discussed for GKAT and HKAT, as well as which ones are also KAT is summarised in Figure 4.

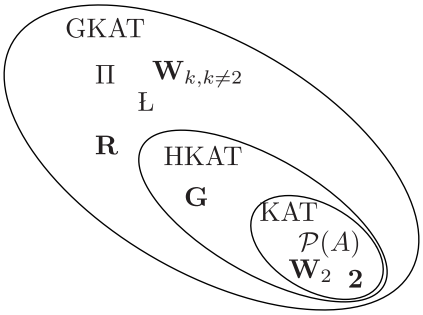

Fig. 4. Examples of KAT, GKAT and HKAT

In HKAT, we can think about the intuitive meaning of the execution of a program guarded by a test as an uncertain execution. For instance, in Example 2, if $b = u$, the expression $u; p$ means that we are not sure if program p could be executed or not.

Just as GKAT, HKAT is also a generalisation of KAT.

Lemma 2. *Any KAT is a HKAT.*

Proof. It suffices to show that axioms (14), (15) and (16) hold for all $a, b, c \in T$. The proof is the same as Lemma 1.

3.2 Heyting Propositional Hoare logic

Let us now discuss how to encode propositional Hoare logic in HKAT. We call this generalisation *Heyting propositional Hoare logic* (HPHL). Differently from what happens in GKAT, the three encodings proposed by D. Kozen for Hoare logic are equivalent in HKAT:

$$b; p = b; p; c \Leftrightarrow b; p \leq b; p; c \Leftrightarrow b; p \leq p; c$$

Hence, the inference rules of Hoare logic can be encoded in HKAT as in classical propositional Hoare logic.

Theorem 2. *The following equational implications are theorems in HKAT.*

1. *Composition rule:*
   $b; p = b; p; c \wedge c; q = c; q; d \Rightarrow b; p; q = b; p; q; d$
2. *Conditional rule:*
   $b; c; p = b; c; p; d \wedge (b \to 0); c; q = (b \to 0); c; q; d$
   $\Rightarrow c; (b; p + (b \to 0); q) = c; (b; p + (b \to 0); q); d$
3. *While rule:*
   $b; c; p = b; c; p; c \Rightarrow c; (b; p)^*; (b \to 0) = c; (b; p)^*; (b \to 0); (b \to 0); c$
4. *Weakening rule:*
   $b' \leq b \wedge b; p = b; p; c \wedge c \leq c' \Rightarrow b'; p = b'; p; c'$

Proof. The proofs for rules 1, 2 and 4 are as in Theorem 1. To prove rule 3, consider

$$c; b; p \leq c; b; p; c \Rightarrow c; (b; p)^* \leq c; (b; p)^*; c.$$

Assuming

$$c; b; p \leq c; b; p; c \tag{28}$$

by (13), it is enough to show

$$c + c; (b; p)^*; c; b; p \leq c; (b; p)^*; c$$

But

$$
\begin{array}{l|l}
\quad c + c; (b; p)^*; c; b; p & \leq \qquad \{ \text{ by distributivity} \} \\
\leq \qquad \{ \text{ by } (28) \} & \quad c; (1 + (b; p)^*; c; b; p); c \\
\quad c + c; (b; p)^*; c; b; p; c & \leq \qquad \{ \text{ by monotonicity} \} \\
\leq \qquad \{ \text{ by B.A} \} & \quad c; (1 + (b; p)^*; b; p); c \\
\quad c; 1; c + c; (b; p)^*; c; b; p; c & \leq \qquad \{ \text{ by } (11) \} \\
 & \quad c; (b; p)^*; c
\end{array}
$$

Note that, as in classical case, for both encodings of PHL previously discussed, the way to reason about the correctness of a program is settled in a bivalent truth space.

4 Conclusion and Further Work

This paper aimed at generalising Kleene algebra with tests, to reason equationally about graded computations and assertions evaluated in a multi-valued truth space. Moreover, the propositional fragment of classic Hoare logic was revisited.

A similar attempt is discussed in [15], which introduces a complete theory of probabilistic KAT to deal with regular programs with probabilities. However, instead of focusing on the possible range of values for tests, or in adding an uncertainty concretisation to them, which have an immediate consequence on program executions, the authors add a new operator $+_\alpha$ to the algebraic structure, where α is a probability value. Thus, in their work, a *probabilistic Kleene algebra with Tests* is defined as

$$(K, T, +, +_\alpha, \cdot, ^*, 0, 1, ^-)$$

where expression $p +_\alpha q$ represents the probabilistic choice between executing a program p with probability α or a program q with probability $1 - \alpha$. More related work on this matter include references [3] and [12]. However, the main ideas behind these approaches is to introduce probabilities at the syntactic level, namely a new choice operator. Our approach, on the other hand, opted by redefining the notions of test and program execution.

The approach taken in this paper for GKAT, of adding a residual as a logical implication to capture a multi-valued setting, is based on previous work reported in [11], where an action lattice is adopted as the basic algebraic structure to generate many-valued dynamic logics.

Originally derived from *action algebras* [7], an action lattice entails both a generic space of computations, with choice, composition and iteration, and, supported by residuation, a proper truth space for a non bivalent interpretation of the assertions (as a residuated lattice). V. Pratt thought about residuation as a pure technicality to obtain a finitely-based equational variety [14]. Subsequently, the work of D. Kozen [7] extended this notion by adding and axiomatizing a meet operation, in order to recover the closure under matricial formation typical of the Kleene algebras [2].

The attentive reader may wonder about the lack of concrete illustrations for the introduced formalism, like simple imperative programs as in [10]. Note, however, that programs are interpreted here as weighted relations and tests as truth degrees. Hence, as it happens in propositional Hoare logic derived from standard KAT, there is no first-order structure to interpret program variables. Consequently, there is no assignment rule neither for GPHL nor for HPHL, as presented here. Extending the formalism in this direction, in order to deal with imperative fuzzy programs is, naturally, in our agenda.

Another important aspect to note is that, since both GKAT and HKAT are generalisations of KAT, as stated by lemmas 1 and 2, all the classical models of KAT, namely relational algebra over a set, languages over an alphabet and traces are naturally examples of our structures. In all these cases, the set of programs K and the set of tests T do not coincide, contrary to what happens with all the examples presented in this paper. Nevertheless, since the introduced structures intend to formalise over fuzzy programs, we want to go a step further: the tentative to formalise fuzzy relations and fuzzy languages, as they are presented in [4], as models of GKAT and HKAT is a priority in our agenda.

In all variants of dynamic logic discussed in the literature, even when some forms of structured computations are taken into consideration, the validity of assertions (for example, of Hoare triples annotating a program) is always stated in classical terms. This means that, even when the object of reasoning is e.g. a fuzzy program or a quantum system, the validity of an assertion over it is discussed in classical, two-valued logic.

In this work we assume, as in classic PHL, that a PCA is valid if $b; p = b; p; c$. In GKAT, this expression states that, after the execution of p guarded by the truth degree of precondition b, a state is reached where the truth degree of the post condition does not modify the value of the execution. In HKAT, for the case considered in example 2, the variation from the classical case comes when $b = u$. Thus, the expression $b; p$ can be interpreted as “we are not sure if program p can be executed”. Due to the nature of the expression (an equality relation), this is clearly tied to the classical, two-valued logic: despite the graded nature of the computations, their correctness is evaluated in a bivalent truth space.

This limitation motivates an approach to be addressed in future work: the intention is to go a step further and resort to the same algebraic structure used to specify the computing paradigm, in order to give semantics to the logic used to reason about it. This makes it possible to discuss the validity of an assertion over a fuzzy or a quantum program in terms of a logic capturing itself fuzzy or quantum reasoning, respectively.

References

1. W. J. Blok and I. Ferreirim. On the structure of hoops. *Algebra Universalis*, 43:233–257, 2000.

2. J. Conway. *Regular Algebra and Finite Machines.* Dover Publications, 1971.
3. J. den Hartog and E. P. de Vink. Verifying probabilistic programs using a Hoare like logic. *Int. J. Found. Comput. Sci.*, 13(3):315–340, 2002.
4. R. Guilherme. A coalgebraic approach to fuzzy automata. Master's thesis, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, Lisboa, 2016.
5. C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.
6. D. Kozen. *The design and analysis of algorithms.* Springer-Verlag New York, 1992.
7. D. Kozen. On action algebras. In *Logic and the Flow of Information*, Amsterdam, 1993.
8. D. Kozen. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Information and Computation*, 110(May 1994):366–390, 1994.
9. D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.
10. D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Transactions on Computational Logic (TOCL)*, 1(212):1–14, 2000.
11. A. Madeira, R. Neves, and M. A. Martins. An exercise on the generation of many-valued dynamic logics. *Journal of Logical and Algebraic Methods in Programming*, 1:1–29, 2016.
12. A. McIver, E. Cohen, and C. Morgan. Using probabilistic kleene algebra for protocol verification. In R. A. Schmidt, editor, *Relations and Kleene Algebra in Computer Science, 9th International Conference on Relational Methods in Computer Science and 4th International Workshop on Applications of Kleene Algebra, RelMiCS/AKA 2006, Manchester, UK, August 29-September 2, 2006, Proceedings*, volume 4136 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2006.
13. A. Platzer. *Logical Analysis of Hybrid Systems - Proving Theorems for Complex Dynamics.* Springer, 2010.
14. V. R. Pratt. Action logic and pure induction. In J. van Eijck, editor, *Logics in AI, European Workshop, JELIA '90, Amsterdam, The Netherlands, September 10-14, 1990, Proceedings*, volume 478 of *Lecture Notes in Computer Science*, pages 97–120. Springer, 1990.
15. R. Qiao, J. Wu, Y. Wang, and X. Gao. Operational semantics of probabilistic Kleene algebra with tests. *Proceedings - IEEE Symposium on Computers and Communications*, pages 706–713, 2008.