Local Completeness Logic on Kleene Algebra with Tests

Marco Milanese[0000−0002−6215−7359] and Francesco Ranzato[0000−0003−0159−0068]

Dipartimento di Matematica, University of Padova, Italy

Abstract. Local Completeness Logic (LCL) has been put forward as a program logic for proving both the correctness and incorrectness of program specifications. LCL is an abstract logic, parameterized by an abstract domain that allows combining over- and under-approximations of program behaviors. It turns out that LCL instantiated to the trivial singleton abstraction boils down to O'Hearn incorrectness logic, which allows us to prove the presence of program bugs. It has been recently proved that suitable extensions of Kleene algebra with tests (KAT) allow representing both O'Hearn incorrectness and Hoare correctness program logics within the same equational framework. In this work, we generalize this result by showing how KATs extended either with a modal diamond operator or with a top element are able to represent the local completeness logic LCL. This is achieved by studying how these extended KATs can be endowed with an abstract domain so as to define the validity of correctness/incorrectness LCL triples and to show that the LCL proof system is logically sound and, under some hypotheses, complete.

Keywords: Local Completeness Logic · Incorrectness Logic · Complete Abstract Interpretation · Kleene Algebra with Tests.

1 Introduction

Kleene algebra [7] with tests (KAT) [17] allows an equational reasoning on programs and their properties. Programs are modeled as elements of a KAT, so that their properties can be algebraically derived through the general equational theory of KATs. KATs feature sound, complete, and decidable equational theories and have found successful applications in several different contexts, most notably in network programming [1, 2, 12, 30, 31]. The foundational study of Kozen [18] has shown that the reasoning of Hoare correctness logic [16] can be encoded and formulated equationally within a KAT. Later work by Desharnais, Möller and Struth [10, 24] extended KAT with a domain (KAD) to express the modal operators of propositional dynamic logic [11], thus enabling a more natural way of reasoning through a map from actions to propositions. The expressive power of KAD has been recently substantiated by Möller, O'Hearn and Hoare [23], who have shown how to encode both Hoare [16] correctness and O'Hearn [25] incorrectness program logics in a unique class of KAD where a backward diamond modality is exploited to encode strongest postconditions. Furthermore, very recently, Zhang, De Amorim and Gaboardi [33, Theorem 1] have shown that O'Hearn incorrectness logic cannot be formulated within a conventional KAT, but, at the same time, a full fledged modal KAT is not needed. In fact, [33] proves that a KAT including

a greatest element, called TopKAT, is capable to encode both Hoare and O'Hearn logic in a purely equational fashion. Moreover, [33] provides a PSPACE algorithm to decide TopKAT equality, based on a reduction to Cohen et al. [6]'s algorithm for KAT.

This stream of works made it possible to reason equationally on both program correctness and incorrectness in the same algebraic framework. For example, in the KAD framework where a backward diamond modality $\langle a|p$ plays the role of strongest postcondition of a KAT element a (viz., a program) for a KAT test p (viz., a precondition), the validity of a Hoare correctness triple $\{p\}\, a\, \{q\}$ is determined by the inequality $\langle a|p \leq q$, while the validity of an O'Hearn incorrectness triple $[p]\ a\ [q]$ boils down to $q \leq \langle a|p$. Moreover, if a KAT test s plays the role of specification for a program a and a Hoare triple $\{p\}\, a\, \{q\}$ is provable, then a can be proved correct through the inequality $q \leq s$. Vice versa, if $[p]\ a\ [q]$ is a provable incorrectness triple, then incorrectness of a can be verified as $q \leq \neg s$.

The Problem. Recently, Bruni et al. [4] put forward a novel program logic, called local completeness logic LCL, which is parameterized by an abstract domain [8, 9] of program stores and simultaneously combines over- and under-approximations of program behaviours. This program logic leverages the notion of *locally complete* abstract interpretation, meaning that the abstract interpretation of atomic program commands, such as variable assignments and Boolean guards, is complete (i.e. with no false alarm) *locally* on the preconditions, as opposed to standard completeness [13, 29] which must be satisfied *globally* for all the preconditions. While a global completeness program logic was proposed in [14], Bruni et al. [4] design a proof system for inferring that a program analysis is locally complete. It turns out [4, Section VI] that the instantiation of this LCL program logic to the trivial store abstraction with a unique "don't know" value abstracting any concrete store property, boils down to O'Hearn incorrectness logic [25]. Moreover, Bruni et al. [5] also show that abstract interpretations can be made locally complete through minimal domain refinements that repair the lack of local completeness in a given program analysis.

In the original definition of LCL in [4] program properties are represented as elements of a concrete domain C and program semantics as functions of type $C \to C$. Although straightforward, this approach determines a specific type of program semantics. Vice versa, by exploiting a KAT, program properties are represented as tests and programs as generic elements of the KAT. Hence, a KAT based formulation becomes agnostic w.r.t. the underlying semantics and can therefore admit multiple different models of computation (e.g., trace-based semantics, or even models not related to program semantics as shown by the language-theoretic example in Section 3.5). Furthermore, KAT is a particularly suitable formalism for compositionally reasoning on programs as all its basic composition operations on programs (concatenation, choice and Kleene iteration) are directly modeled within the algebra: this allows us to represent composite programs and tests as elements of the KAT and, in particular, to check for their equality and inclusion directly in the algebra. Thus, following the KAT-based model of incorrectness logic advocated by Möller, O'Hearn and Hoare [23], this paper pushes forward this line of work by studying an algebraic formulation of LCL program logic, with the objective of showing that there is no need to leverage particular semantic properties of programs to reason on their local completeness.

Contributions. In this work, we show that the local completeness logic LCL can be made fully algebraic in a suitable KAT, yet preserving all its noteworthy logical properties proved in [4]. For this purpose we show that:

- Our proof systems are logically sound and complete (likewise [4], completeness needs some additional hypotheses).
- By instantiating the algebraic version of the LCL logic to the trivial domain abstracting any concrete value to "don't know" we exactly obtain O'Hearn incorrectness program logic [25], thus retrieving its logical soundness and completeness as consequences of our results.
- Triples of O'Hearn incorrectness logic carry two postconditions, corresponding to normal and erroneous program termination. While the original local completeness logic LCL in [4] only considers normal termination, we propose a generalization that also supports erroneous termination. Moreover, we use the KAD construction of [23] to generalize our logical soundness and completeness results to incorrectness triples.

In particular, we study two different formulations of LCL given: (1) in a KAD, the KAT model used in [23], and (2) in a TopKAT, the KAT model employed in [33]. In both frameworks, we put forward a suitable notion of abstract domain of KAT that, correspondingly, induces a sound abstract semantics for KAT programs (i.e., KAT terms). Our local completeness logic on KAT, called LCK, turns out to be logically sound w.r.t. this abstract semantics, meaning that a provable LCK triple $[p]\ a\ [q]$ for an abstract domain A on a KAT K satisfies:

(i) q is below the strongest postcondition in K of the program term a for the precondition p;
(ii) the program term a is locally complete for the precondition p in the abstract domain A;
(iii) the approximations in A of q and of the strongest postcondition of a for p coincide.

The proofs of all the results have been omitted and can be found in the full version of the paper [22].

2 Background on Kleene Algebra with Tests

A Kleene algebra with tests (KAT) is a purely algebraic structure that provides an elegant equational framework for program reasoning. A KAT consists of actions, playing the role of programs, and tests, interpreted as pre/postconditions and Boolean guards. KAT elements can be combined with three basic operations: nondeterministic choice $a_1 + a_2$, sequential composition $a_1; a_2$, and Kleene iteration a^*. A standard model of KAT used to represent computations is the relational model, in which KAT elements are binary relations on some set, thus modeling programs as a relation between input and output states. Further models of KAT include regular languages over a finite alphabet, square matrices over another Kleene algebra, and Kleene algebra modulo theories [15]. In the following, we briefly recall some basics of KAT. For more details, the reader is referred to [7, 10, 17].

An *idempotent-semiring* (*i-semiring*) is a tuple $(A, +, \cdot, 0, 1)$ where: (1) $(A, +, 0)$ is a commutative monoid with an idempotent addition, i.e., for all $a \in A$, $a + a = a$; (2) $(A, \cdot, 1)$ is a monoid, where the multiplication symbol $\cdot$ is often omitted, such that, for any $a \in A$, $0 \cdot a = a \cdot 0$; (3) multiplication distributes over addition (in both arguments). In an i-semiring A, the relation $a \leq b \triangleq a + b = b$ is a partial order, referred to as the natural ordering, that we will implicitly use throughout the paper. Note that the addition $+$ is the join w.r.t. this natural ordering.

A *test-semiring* is a tuple $(A, \mathsf{test}(A), +, \cdot, \neg, 0, 1)$ where: (1) $(A, +, \cdot, 0, 1)$ is an i-semiring; (2) $\mathsf{test}(A) \subseteq A$, and $(\mathsf{test}(A), \vee, \wedge, \neg, 0, 1)$ is a Boolean subalgebra of A with greatest element 1 and least element 0, complement $\neg$, where the meet $\wedge$ and join $\vee$ of the Boolean algebra $\mathsf{test}(A)$ coincide, resp., with multiplication $\cdot$ and addition $+$.

A *Kleene algebra* is a tuple $(K, +, \cdot, ^*, 0, 1)$ where: (1) $(K, +, \cdot, 0, 1)$ is an i-semiring; (2) $(\cdot)^* : K \to K$ is a unary operation, called Kleene star or iteration, satisfying the following conditions:

$$1 + aa^* \leq a^* \qquad 1 + a^*a \leq a^* \qquad (*\text{-unfold})$$
$$b + ac \leq c \Rightarrow a^*b \leq c \qquad b + ca \leq c \Rightarrow ba^* \leq c \qquad (*\text{-induction})$$

Definition 2.1 (KAT [17]). A *Kleene algebra with tests* (*KAT*) is a two-sorted algebra $(K, \mathsf{test}(K), +, \cdot, ^*, \neg, 0, 1)$ such that $(K, \mathsf{test}(K), +, \cdot, \neg, 0, 1)$ is a test-semiring and $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra.
A KAT K is *countably-test-complete* (*CTC*) if any countable subset of $\mathsf{test}(K)$ admits least upper bound (lub).
A KAT is **-continuous*, referred to as KAT*, if it satisfies the following condition: for all $a, b, c \in K$, $ab^*c = \bigvee_{n \in \mathbb{N}} ab^n c$ (this equation implicitly assumes that the lub $\bigvee_{n \in \mathbb{N}} ab^n c$, w.r.t. the natural ordering of K, exists).

A *relational KAT* [19] on a carrier set X is determined by a set $K \subseteq \wp(X \times X)$ of binary relations on X with tests $\mathsf{test}(K) \subseteq \wp(\{(x, x) \mid x \in X\})$, where addition is union, multiplication is composition of relations, the additive identity is the empty relation, the multiplicative identity is $\{(x, x) \mid x \in X\}$, the Kleene star is the reflexive-transitive closure, and test complement is set complementation w.r.t. the multiplicative identity.

Informally, a backward diamond $\langle \cdot | \cdot$ on a KAT allows us to compute strongest postconditions of programs, that is, $\langle a | p$ can be interpreted as $\mathsf{post}[a]p$.

Definition 2.2 (bdKAT [23]). A *backward-diamond KAT* (*bdKAT*) is a two-sorted algebra $(K, \mathsf{test}(K), +, \cdot, ^*, \neg, 0, 1, \langle |)$ such that:

(1) $(K, \mathsf{test}(K), +, \cdot, ^*, \neg, 0, 1)$ is a KAT;
(2) $\langle \cdot | \cdot : K \to (\mathsf{test}(K) \to \mathsf{test}(K))$ is a *backward-diamond* operator satisfying the following conditions: for all $a, b \in K$ and $p, q \in \mathsf{test}(K)$,

$$\langle a | p \leq q \Leftrightarrow pa \leq aq \qquad \text{(bd1)}$$
$$\langle ab | p = \langle b | (\langle a | p) \qquad \text{(bd2)}$$

□

The axiom (bd1) is equivalent to requiring that $\langle a|p$ is the least test in K satisfying $pa \leq aq$ (the original definition of Kleene algebra with domain in [10] is of this form). Moreover, $pa \leq aq$ in (bd1) is equivalent to $pa = paq$ (see [10, Lemma 3.4]).

Definition 2.3 (TopKAT [21]). A *KAT with top* (*TopKAT*) is a KAT K that contains a largest element $\top \in K$, that is, for all $a \in K$, $a \leq \top$. □

3 Local Completeness Logic in KAT

We investigate how the local completeness program logic LCL [4] can be interpreted on a KAT. To achieve this, we need to address the following tasks:

- To define a notion of abstract domain of a KAT, with the aim of abstracting the set of program predicates, namely tests of a KAT;
- To establish a concrete semantics and a corresponding sound abstract semantics of programs on KATs;
- To adapt the local completeness proof system to attain valid triples on a KAT;
- To prove logical soundness and completeness w.r.t. a KAT of this new proof system.

3.1 Program Properties in KAT

Program properties can be broadly classified as intensional and extensional. The former relate to how programs are written, while the latter concern the input-output relation of a program, i.e., its strongest postcondition denotational semantics. Local completeness logic LCL relies on an abstract interpretation of programs which crucially depends on intensional properties of programs, meaning that even if two programs share the same denotation, they could well have different abstract semantics. Thus, we expect that an appropriate definition of abstract semantics based on a KAT model should also be intensional. Given two elements a and b of a modal bdKAT playing the role of programs, we therefore expect that their backward diamond functions might coincide, i.e. $\langle a| = \langle b|$, even if a and b encode different programs, i.e. $a \neq b$. However, as shown by the following remark for the basic relational model of KAT, it might happen that for certain classes of KAT models the backward diamond interpretation is injective.

Proposition 3.1. *Let K be a relational KAT on a set X where* $\mathsf{test}(K) = \wp(\{(x, x) \mid x \in X\})$. *Then, for all $a, b \in K$, $\langle a| = \langle b| \Leftrightarrow a = b$.*

This means that, at least for some fundamental KAT models, KAT elements are equal iff they are extensionally equal, or, equivalently, they carry exclusively extensional program properties. In this case, when a program is encoded with a KAT element all the intensional properties are lost and it is indistinguishable from any other program with the same denotational semantics. Therefore, an abstract interpretation-based semantics can not be defined directly on KAT elements.

3.2 KAT Language

As a consequence of the discussion in Section 3.1, the concrete semantics cannot be directly defined on KAT elements. A solution is to define it on an inductive language. Actually, in a language of programs, two elements are equal iff they are syntactically equal, or, in other terms, if the corresponding programs are written in the same way. This property makes a language an ideal basis upon which a semantics can be defined, because this brings the chance of depending on intensional properties.

A natural choice for defining this language of programs is the so-called *KAT language*, as originally defined by Kozen and Smith [19, Section 2.3], because it contains all and only the operators of a KAT, so that the interpretation of language terms as KAT elements is the most natural one. This language is inductively defined from two disjoint sets of primitive actions and tests through the basic elements/operations $0, 1, +, \cdot, ^*$ of KATs. More precisely, given a set Σ of *primitive actions* and a set B of *primitive tests* such that $\Sigma \cap B = \varnothing$, the corresponding *KAT language* $T_{\Sigma,B}$ of terms is defined as follows:

$$\begin{aligned} \mathrm{Atom} \ni \mathtt{a} &::= a \in \Sigma \mid p \in B \\ T_{\Sigma,B} \ni \mathtt{t} &::= \mathtt{a} \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{t}_1 + \mathtt{t}_2 \mid \mathtt{t}_1 \cdot \mathtt{t}_2 \mid \mathtt{t}^* \end{aligned}$$

For simplicity, we assume that $\mathtt{0}$ and $\mathtt{1}$ are primitive tests in B, so that $\mathtt{0}, \mathtt{1} \in \mathrm{Atom}$. The notation $\mathrm{Atom}(\mathtt{t}) \subseteq \mathrm{Atom}$ will denote the set of atoms occurring in a term $\mathtt{t} \in T_{\Sigma,B}$. Notice that a KAT language $T_{\Sigma,B}$ is an equivalent representation of the language of regular commands used in [4, 25] for their program logics.

Given a KAT K, an evaluation of atoms in K is a mapping $u : \mathrm{Atom} \to K$ such that $\mathtt{p} \in B \Rightarrow u(\mathtt{p}) \in \mathsf{test}(K)$. An evaluation u induces an interpretation of terms $[\![\cdot]\!]_u : T_{\Sigma,B} \to K$, which is inductively defined as expected:

$$\begin{aligned} [\![\mathtt{a}]\!]_u &\triangleq u(\mathtt{a}) & [\![\mathtt{t}_1 + \mathtt{t}_2]\!]_u &\triangleq [\![\mathtt{t}_1]\!]_u + [\![\mathtt{t}_2]\!]_u \\ [\![\mathtt{t}_1 \cdot \mathtt{t}_2]\!]_u &\triangleq [\![\mathtt{t}_1]\!]_u \cdot [\![\mathtt{t}_2]\!]_u & [\![\mathtt{t}^*]\!]_u &\triangleq [\![\mathtt{t}]\!]_u^* \end{aligned}$$

In turn, the concrete semantic function

$$[\![\cdot]\!]^K : T_{\Sigma,B} \to \big(\mathsf{test}(K) \to \mathsf{test}(K)\big)$$

models the strongest postcondition of a program, i.e. of a language term, for a given precondition, i.e. a KAT test. This is therefore defined in terms of the backward diamond of a bdKAT as follows:

$$[\![\mathtt{t}]\!]^K p \triangleq \langle [\![\mathtt{t}]\!]_u | p. \tag{1}$$

We will often use $[\![\mathtt{t}]\!]$ to denote a concrete semantics, by omitting the superscript K when it is clear from the context.

3.3 Kleene Abstractions

An abstract domain is used in abstract interpretation for approximating store properties, i.e., sets of program stores form the concrete domain, likewise in our KAT model, the role of concrete domain is played by the set of tests $\mathsf{test}(K)$ of a KAT K, ordered by the natural ordering induced by K.

Definition 3.2 (Kleene Abstract Domain). A poset $(A, \leq_A)$ is a *Kleene abstract domain* of a bdKAT K if:

(i) There exists a Galois insertion, defined by a concretization map $\gamma : A \to \mathsf{test}(K)$ and an abstraction map $\alpha : \mathsf{test}(K) \to A$, of the poset $(A, \leq_A)$ into the poset $(\mathsf{test}(K), \leq_K)$;
(ii) A is countably-complete, i.e., any countable subset of A admits a lub. □

The abstract semantic function $[\![\cdot]\!]^\sharp_A : T_{\Sigma,B} \to (A \to A)$ defines how abstract preconditions are transformed into abstract postconditions. Likewise store-based abstract interpretation, this abstract semantics is inductively defined as follows:

$$\begin{array}{ll} [\![\mathtt{a}]\!]^\sharp_A p^\sharp \triangleq \alpha([\![\mathtt{a}]\!]^K \gamma(p^\sharp)) & [\![\mathtt{t}_1 + \mathtt{t}_2]\!]^\sharp_A p^\sharp \triangleq [\![\mathtt{t}_1]\!]^\sharp_A p^\sharp + [\![\mathtt{t}_2]\!]^\sharp_A p^\sharp \\ [\![\mathtt{t}_1 \cdot \mathtt{t}_2]\!]^\sharp_A p^\sharp \triangleq [\![\mathtt{t}_2]\!]^\sharp_A ([\![\mathtt{t}_1]\!]^\sharp_A p^\sharp) & [\![\mathtt{t}^*]\!]^\sharp_A p^\sharp \triangleq \bigvee_{n\in\mathbb{N}} ([\![\mathtt{t}]\!]^\sharp_A)^n p^\sharp \end{array} \tag{2}$$

It is worth remarking that condition (ii) of Definition 3.2 ensures that the abstract semantics of the Kleene star in (2) is well defined. It turns out that $[\![\cdot]\!]^\sharp_A$ is a sound (and monotonic) abstract semantics.

Theorem 3.3 (Soundness of bdKAT Abstract Semantics). *Let A be a Kleene abstraction of a CTC bdKAT K and $T_{\Sigma,B}$ be a language interpreted on K. For all $p^\sharp, q^\sharp \in A$, $p \in \mathsf{test}(K)$, and $\mathtt{t} \in T_{\Sigma,B}$:*

$$p^\sharp \leq_A q^\sharp \Rightarrow [\![\mathtt{t}]\!]^\sharp_A p^\sharp \leq_A [\![\mathtt{t}]\!]^\sharp_A q^\sharp \qquad \text{(monotonicity)}$$
$$\alpha([\![\mathtt{t}]\!]^K p) \leq_A [\![\mathtt{t}]\!]^\sharp_A \alpha(p) \qquad \text{(soundness)}$$

3.4 Local Completeness Logic on bdKAT

Given a Kleene abstract domain A, we will slightly abuse notation by using

$$A \triangleq \gamma \circ \alpha : \mathsf{test}(K) \to \mathsf{test}(K)$$

as a function (indeed, this is the upper closure operator on tests induced by the Galois insertion defining A). Let us recall the notions of global vs. local completeness. If $f : \mathsf{test}(K) \to \mathsf{test}(K)$ is any test transformer then:

- A is *globally complete* for f, denoted $\mathbb{C}^A(f)$, iff $A \circ f = A \circ f \circ A$;
- A is *locally complete* for f on a concrete test $p \in \mathsf{test}(K)$, denoted $\mathbb{C}^A_p(f)$, iff $A \circ f(p) = A \circ f \circ A(p)$.

It is known [14] that global completeness is hard to achieve in practice, even for simple programs. Moreover, a complete and compositional (i.e., inductively defined on program structure) abstract interpretation is even harder to design [3]. This motivated to study a local notion of completeness in abstract interpretation [4] as a pragmatic and more attainable weakening of standard global completeness.

In our local completeness logic on a Kleene algebra, a triple $[p]\ \mathtt{t}\ [q]$, where p and q are tests and $\mathtt{t}$ is a language term, will be valid when:

$$\frac{\mathtt{a} \in \Sigma \cup B \qquad \mathbb{C}_p^A([\![\mathtt{a}]\!])}{\vdash_A^{\mathrm{K}} [p]\ \mathtt{a}\ [[\![\mathtt{a}]\!]p]}\ \text{(transfer)}$$

$$\frac{p' \leq p \leq A(p') \qquad \vdash_A^{\mathrm{K}} [p']\ \mathtt{t}\ [q'] \qquad q \leq q' \leq A(q')}{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}\ [q]}\ \text{(relax)}$$

$$\frac{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}_1\ [r] \qquad \vdash_A^{\mathrm{K}} [r]\ \mathtt{t}_2\ [q]}{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}_1 \cdot \mathtt{t}_2\ [q]}\ \text{(seq)} \qquad \frac{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}_1\ [q_1] \qquad \vdash_A^{\mathrm{K}} [p]\ \mathtt{t}_2\ [q_2]}{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}_1 + \mathtt{t}_2\ [q_1 + q_2]}\ \text{(join)}$$

$$\frac{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}\ [r] \qquad \vdash_A^{\mathrm{K}} [p + r]\ \mathtt{t}^*\ [q]}{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}^*\ [q]}\ \text{(rec)} \qquad \frac{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}\ [q] \qquad q \leq A(p)}{\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}^*\ [p + q]}\ \text{(iterate)}$$

Fig. 1. Proof system LCK_A.

(1) q is an under-approximation of the concrete semantics of $\mathtt{t}$ from a precondition p;
(2) A is locally complete for $[\![\mathtt{t}]\!]$ on the precondition p;
(3) q and $[\![\mathtt{t}]\!]p$ have the same over-approximation in A.

Definition 3.4 (Triple Validity). Let K a CTC bdKAT, A be a Kleene abstraction of K, and $T_{\Sigma,B}$ be a KAT language interpreted on K. For all $p, q \in \mathsf{test}(K)$ and $\mathtt{t} \in T_{\Sigma,B}$, a triple $[p]\ \mathtt{t}\ [q]$ is valid in A, denoted by $\models_A^{\mathrm{K}} [p]\ \mathtt{t}\ [q]$, if

(i) $q \leq_K [\![\mathtt{t}]\!]^K p$;
(ii) $[\![\mathtt{t}]\!]_A^\sharp \alpha(p) = \alpha(q) = \alpha([\![\mathtt{t}]\!]^K p)$. □

The local completeness proof system in [4] can be adapted to our algebraic framework, yielding the set of rules denoted by LCK_A in Figure 1. The only syntactic difference concerns the usage of elements of $\mathsf{test}(K)$ as pre/postconditions and the language of terms $T_{\Sigma,B}$ playing the role of programs.

It turns out that the logic LCK_A is logically sound (we use "logical" soundness to avoid overloading the soundness of abstract semantics).

Theorem 3.5 (Logical Soundness of $\vdash_A^{\mathrm{K}}$). *If* $\vdash_A^{\mathrm{K}} [p]\ \mathtt{t}\ [q]$ *then*

(i) $q \leq_K [\![\mathtt{t}]\!]p$;
(ii) $[\![\mathtt{t}]\!]_A^\sharp \alpha(p) = \alpha(q) = \alpha([\![\mathtt{t}]\!]p)$.

Analogously to what happens for LCL, we can prove that LCK_A is logically complete under these two additional hypotheses:

(A) The following infinitary rule is added to LCK_A:

$$\frac{\forall n \in \mathbb{N}.\ \vdash_A^{\mathrm{K}} [p_n]\ \mathtt{t}\ [p_{n+1}]}{\vdash_A^{\mathrm{K}} [p_0]\ \mathtt{t}^*\ [\bigvee_{n\in\mathbb{N}} p_n]}\ \text{(limit)}$$

Let us point out that the lub $\bigvee_{n\in\mathbb{N}} p_n$ always exists in K, as a consequence of the CTC requirement on K.

(B) The concrete semantics of the primitive actions and tests occurring in the program are globally complete.

It can be proved that the rule (limit) preserves logical soundness (see the full version [22]).

Theorem 3.6 (Logical Completeness of $\vdash^{\mathrm{K}}_{A}$). *Assume that conditions* (A) *and* (B) *hold. If* $\models^{\mathrm{K}}_{A} [p]\ \mathtt{t}\ [q]$ *then* $\vdash^{\mathrm{K}}_{A} [p]\ \mathtt{t}\ [q]$.

Summing up, this shows that the local completeness logic LCL introduced in [4] can be made *fully algebraic* by means of a natural interpretation on modal KATs with a backward diamond operator, still preserving its logical soundness and completeness, which are proved by using just the algebraic axioms of this class of KATs. Hence, this shows that there is no need to leverage particular semantic properties of programs to reason on their local completeness.

3.5 An Example of a Language-Theoretic KAT

To give an example digressing from programs and showing the generality of the KAT-based approach, we describe a language-theoretic model of Kleene algebra, early introduced by Kozen and Smith [19, Section 3].

Let $\Sigma = \{\mathtt{u}\}$ and $B = \{\mathtt{b}_1, \mathtt{b}_2\}$ be, resp., the sets of primitive actions and tests. An *atom* is a string c_1c_2, where $c_i \in \{\mathtt{b}_i, \overline{\mathtt{b}}_i\}$. If $c_i = \mathtt{b}_i$, where $i \in \{1, 2\}$, then $\mathtt{b}_i$ appears positively in the atom c_1c_2, while if $c_i = \overline{\mathtt{b}}_i$ it appears negatively. A *guarded string* is either a single atom or a string $\alpha_0\mathtt{a}_1\alpha_1...\mathtt{a}_n\alpha_n$, where α_i are atoms and $\mathtt{a}_i \in \Sigma$. If we are only interested in the first (last) atom of a guarded string $\alpha\mathtt{a}_1\alpha_1...\mathtt{a}_n\beta$, we may refer to it through the syntax αx ($x\beta$). Concatenation of guarded strings is given by a coalesced product operation $\diamond$, which is partially defined as follows:

$$x\alpha \diamond \beta y \triangleq \begin{cases} x\alpha y & \text{if } \alpha = \beta \\ \text{undefined} & \text{otherwise} \end{cases}$$

The elements of this KAT $\mathcal{G}$ are sets of guarded strings. Thus, $+$ is set union, the product is defined as pointwise coalesced product:

$$A \cdot B \triangleq \{x \diamond y \mid x \in A, y \in B\},$$

while the Kleene iteration is: $A^* \triangleq \bigcup_{n\in\mathbb{N}} A^n$. The product identity corresponds to the whole set of atoms $1_{\mathcal{G}} \triangleq \{\mathtt{b}_1\mathtt{b}_2, \mathtt{b}_1\overline{\mathtt{b}}_2, \overline{\mathtt{b}}_1\mathtt{b}_2, \overline{\mathtt{b}}_1\overline{\mathtt{b}}_2\}$, while $0_{\mathcal{G}}$ is the empty set. The set of tests is $\mathsf{test}(\mathcal{G}) \triangleq \wp(1_{\mathcal{G}})$.

It turns out that $\mathcal{G}$ is a bdKAT, whose backward diamond is as follows: for all $a \in \mathcal{G}$ and $p \in \mathsf{test}(\mathcal{G})$,

$$\langle a|p = \{\beta \mid x\beta \in pa\}. \tag{3}$$

Proof. Let $r = \{\beta \mid x\beta \in pa\}$. It can be proved (see the full version [22]) that in a bdKAT K, for all $p \in \mathsf{test}(K)$ and $a \in K$, (bd1) holds iff $\langle a|p$ is the least (w.r.t. the natural ordering) $q \in \mathsf{test}(K)$ such that $pa = paq$. Therefore, it is enough to show that r is the least $q \in \mathsf{test}(\mathcal{G})$ satisfying $pa = paq$.

Let $x\beta \in pa$. By definition, $\beta \in r$ means that $x\beta \diamond \beta = x\beta$ is contained in par. This therefore means that $pa \leq par$. The opposite inequality is trivial since r is a test, hence $r \leq 1_{\mathcal{G}}$, and by monotonicity of $\cdot$, we have that $pa \geq par$, thus implying $pa = par$. Assume now, by contradiction, that there exists $t \in \mathsf{test}(\mathcal{G})$ such that $pa = pat$, $t \leq r$ and $t \neq r$. This means that there is at least an atom β in r which is not in t. By definition of r, there is a guarded string $x\beta \in pa$. Since $pa = pat$, the last atom of all the guarded strings in pa must be in t, but this does not hold for $x\beta$ as $\beta \notin t$. □

Let us consider the evaluation function $G : \mathrm{Atom} \to \mathcal{G}$ as defined in [19]:

$$G(\mathtt{a}) \triangleq \{\alpha \mathtt{a} \beta \mid \alpha, \beta \in 1_{\mathcal{G}}\},$$
$$G(\mathtt{b}) \triangleq \{\alpha \in 1_{\mathcal{G}} \mid \mathtt{b} \text{ appears positively in } \alpha\}.$$

We consider the abstract domain $A \triangleq \{\top, e, o, \bot\}$ determined by the following abstraction $\alpha : \mathsf{test}(\mathcal{G}) \to A$ and concretization $\gamma : A \to \mathsf{test}(\mathcal{G})$ maps:

$$\alpha(p) \triangleq \begin{cases} \bot & \text{if } p = \emptyset \\ e & \text{if } \emptyset \subsetneq p \subseteq \{\mathtt{b}_1\mathtt{b}_2, \overline{\mathtt{b}}_1\overline{\mathtt{b}}_2\} \\ o & \text{if } \emptyset \subsetneq p \subseteq \{\overline{\mathtt{b}}_1\mathtt{b}_2, \mathtt{b}_1\overline{\mathtt{b}}_2\} \\ \top & \text{otherwise} \end{cases} \qquad \gamma(p^\sharp) \triangleq \begin{cases} \emptyset & \text{if } p^\sharp = \bot \\ \{\mathtt{b}_1\mathtt{b}_2, \overline{\mathtt{b}}_1\overline{\mathtt{b}}_2\} & \text{if } p^\sharp = e \\ \{\overline{\mathtt{b}}_1\mathtt{b}_2, \mathtt{b}_1\overline{\mathtt{b}}_2\} & \text{if } p^\sharp = o \\ 1_{\mathcal{G}} & \text{if } p^\sharp = \top \end{cases}$$

By counting, in an atom, the number of primitive tests that appear positively we obtain an integer that may be even or odd. Hence, this abstract domain A represents the property of being even e or odd o of all the atoms occurring in a test $p \in \mathsf{test}(\mathcal{G})$.

By using our logic LCK, we study the correctness of the program $\mathtt{r} \triangleq (\mathtt{u} \cdot \mathtt{b}_1)^* \in \mathcal{G}$, assuming a precondition $p \triangleq \{\mathtt{b}_1\mathtt{b}_2, \overline{\mathtt{b}}_1\overline{\mathtt{b}}_2\} \in \mathsf{test}(\mathcal{G})$ and a specification $\mathrm{Spec} \triangleq p = \gamma(e)$. Let us define two auxiliary tests: $q \triangleq \{\mathtt{b}_1\mathtt{b}_2, \mathtt{b}_1\overline{\mathtt{b}}_2\}$, $s \triangleq \{\mathtt{b}_1\mathtt{b}_2, \mathtt{b}_1\overline{\mathtt{b}}_2, \overline{\mathtt{b}}_1\overline{\mathtt{b}}_2\}$. Using the equation (3), we can easily check the following local completeness equations:

$$\alpha(\llbracket \mathtt{u} \rrbracket A(s)) = \alpha(\llbracket \mathtt{u} \rrbracket 1_{\mathcal{G}}) = \alpha(1_{\mathcal{G}}) = \top = \alpha(1_{\mathcal{G}}) = \alpha(\llbracket \mathtt{u} \rrbracket s) \quad \Rightarrow \mathbb{C}^A_s(\llbracket \mathtt{u} \rrbracket)$$
$$\alpha(\llbracket \mathtt{u} \rrbracket A(p)) = \alpha(\llbracket \mathtt{u} \rrbracket p) = \alpha(1_{\mathcal{G}}) = \top = \alpha(1_{\mathcal{G}}) = \alpha(\llbracket \mathtt{u} \rrbracket p) \quad \Rightarrow \mathbb{C}^A_p(\llbracket \mathtt{u} \rrbracket)$$
$$\alpha(\llbracket \mathtt{b}_1 \rrbracket A(1_{\mathcal{G}})) = \alpha(\llbracket \mathtt{b}_1 \rrbracket 1_{\mathcal{G}}) = \alpha(q) = \top = \alpha(q) = \alpha(\llbracket \mathtt{b}_1 \rrbracket 1_{\mathcal{G}}) \quad \Rightarrow \mathbb{C}^A_{1_{\mathcal{G}}}(\llbracket \mathtt{b}_1 \rrbracket)$$

Therefore, we have the following derivation in LCK_A of the triple $[p]\ \mathtt{r}\ [s]$:

$$\dfrac{\dfrac{\dfrac{\mathbb{C}^A_p(\llbracket \mathtt{u} \rrbracket)}{\vdash^{\mathsf{K}}_A [p]\ \mathtt{u}\ [1_{\mathcal{G}}]}\text{(transfer)} \quad \dfrac{\mathbb{C}^A_{1_{\mathcal{G}}}(\llbracket \mathtt{b}_1 \rrbracket)}{\vdash^{\mathsf{K}}_A [1_{\mathcal{G}}]\ \mathtt{b}_1\ [q]}\text{(transfer)}}{\vdash^{\mathsf{K}}_A [p]\ \mathtt{u} \cdot \mathtt{b}_1\ [q]}\text{(seq)} \quad \dfrac{\dfrac{\mathbb{C}^A_s(\llbracket \mathtt{u} \rrbracket)}{\vdash^{\mathsf{K}}_A [s]\ \mathtt{u}\ [1_{\mathcal{G}}]}\text{(transfer)} \quad \dfrac{\mathbb{C}^A_{1_{\mathcal{G}}}(\llbracket \mathtt{b}_1 \rrbracket)}{\vdash^{\mathsf{K}}_A [1_{\mathcal{G}}]\ \mathtt{b}_1\ [q]}\text{(transfer)}}{\vdash^{\mathsf{K}}_A [s]\ \mathtt{u} \cdot \mathtt{b}_1\ [q]}\text{(seq)} \quad q \leq A(s)}{\vdash^{\mathsf{K}}_A [s]\ (\mathtt{u} \cdot \mathtt{b}_1)^*\ [s]}\text{(iterate)}}{\vdash^{\mathsf{K}}_A [p]\ (\mathtt{u} \cdot \mathtt{b}_1)^*\ [s]}\text{(rec)}$$

Here, in accordance with the soundness Theorem 3.5, we have that $s \subseteq \llbracket \mathtt{r} \rrbracket p \subseteq A(s)$. Observe that $A(s) \not\subseteq \mathrm{Spec}$ holds, meaning that an abstract interpretation-based analysis fails to prove that the program $\mathtt{r}$ is correct for Spec. However, unlike conventional abstract interpretation, LCK_A is capable to show that $s \smallsetminus \mathrm{Spec} = \{\mathtt{b}_1\overline{\mathtt{b}}_2\}$ is indeed a true alert, meaning that the program $\mathtt{r}$ is really incorrect and the failure to prove its correctness was not due to a false alarm.

$$\frac{\mathtt{a} \in \mathrm{Atom}}{\vdash_{\mathrm{UL}} [p]\ \mathtt{a}\ [[\![\mathtt{a}]\!]p]}\ \text{(transfer)}$$

$$\frac{}{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}\ [0]}\ \text{(empty)} \qquad \frac{p' \leq p \qquad \vdash_{\mathrm{UL}} [p']\ \mathtt{t}\ [q'] \qquad q \leq q'}{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}\ [q]}\ \text{(consequence)}$$

$$\frac{\vdash_{\mathrm{UL}} [p_1]\ \mathtt{t}\ [q_1] \qquad \vdash_{\mathrm{UL}} [p_2]\ \mathtt{t}\ [q_2]}{\vdash_{\mathrm{UL}} [p_1 + p_2]\ \mathtt{t}\ [q_1 + q_2]}\ \text{(disj)} \qquad \frac{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}_1\ [r] \qquad \vdash_{\mathrm{UL}} [r]\ \mathtt{t}_2\ [q]}{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}_1 \cdot \mathtt{t}_2\ [q]}\ \text{(seq)}$$

$$\frac{}{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}^*\ [p]}\ \text{(iterate zero)} \qquad \frac{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}^* \cdot \mathtt{t}\ [q]}{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}^*\ [q]}\ \text{(iterate non-zero)}$$

$$\frac{\forall n \in \mathbb{N}.\ \vdash_{\mathrm{UL}} [p_n]\ \mathtt{t}\ [p_{n+1}]}{\vdash_{\mathrm{UL}} [p_0]\ \mathtt{t}^*\ [\bigvee_{n\in\mathbb{N}} p_n]}\ \text{(back-v)} \qquad \frac{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}_i\ [q], \text{ with } i = 1 \text{ or } i = 2}{\vdash_{\mathrm{UL}} [p]\ \mathtt{t}_1 + \mathtt{t}_2\ [q]}\ \text{(choice)}$$

Fig. 2. Proof System UL.

3.6 Under-Approximation Logic

O'Hearn [25] incorrectness logic (IL) establishes two main novelties w.r.t. the seminal Hoare logic of program correctness [16]: (1) a valid postcondition of an incorrectness triple for a program P is an *under-approximation* of the strongest postcondition of P, rather than an over-approximation of Hoare logic; (2) incorrectness triples feature two postconditions: one corresponding to a "normal" program termination and one corresponding to an erroneous termination. Even if IL was originally defined with both those features, we first neglect the second one — i.e., we consider "normal" termination only — and we refer to the resulting program logic as Under-approximation Logic, denoted by UL. For the sake of clarity, Figure 2 recalls the UL proof system, adapted to our algebraic framework. We only focus on the "propositional" fragment of this logic, meaning that the roles of all the special program commands (i.e., `error`, `assume`, `skip`, `nondet` used in [25]) and variable manipulations commands of incorrectness logic are played by some corresponding elements in Atom. Hence, for all of them, the single rule (transfer) is unifying and enough for our purposes.

Analogously to what has been proved in [4, Section 6] for LCL, it turns out that the trivial abstraction, i.e., the abstract domain $A_{tr} \triangleq \{\top\}$ that approximates all the concrete elements to a single abstract element $\top$, allows us to show that the instantiation $\mathrm{LCK}_{A_{tr}}$, with the additional rule (limit), boils down to UL, namely, our LCK logic generalizes UL, even when both are interpreted on KATs.

Theorem 3.7 ($\mathrm{LCK}_{A_{tr}} \equiv \mathrm{UL}$)**.** *Let K be a CTC bdKAT. Assume that $\mathrm{LCK}_{A_{tr}}$ includes the rule* (limit)*. For any $p, q \in \mathsf{test}(K)$ and $\mathtt{t} \in T_{\Sigma,B}$:*

$$\vdash^{\mathrm{K}}_{A_{tr}} [p]\ \mathtt{t}\ [q] \quad \Leftrightarrow \quad \vdash_{\mathrm{UL}} [p]\ \mathtt{t}\ [q].$$

Moreover, since the abstraction map defining A_{tr} is $\alpha_{A_{tr}} = \lambda x.\top$, we have that condition (ii) of Definition 3.4 for the validity of a LCK triple trivially holds, that is,

$[\![\mathtt{t}]\!]^\sharp_A \alpha(p) = \top = \alpha(q) = \alpha([\![\mathtt{t}]\!]^K p)$. This therefore entails that

$$\models^{\mathrm{K}}_{A_{tr}} [p]\ \mathtt{t}\ [q] \quad \Leftrightarrow \quad \models_{\mathrm{UL}} [p]\ \mathtt{t}\ [q]. \tag{4}$$

This allows us to retrieve the soundness and completeness results of incorrectness logic [25] as a consequence of those for LCK.

Corollary 3.8. *Under the same hypotheses of Theorem 3.7, the proof system* UL *is sound and complete, that is,* $\vdash_{\mathrm{UL}} [p]\ \mathtt{t}\ [q] \Leftrightarrow \models_{\mathrm{UL}} [p]\ \mathtt{t}\ [q]$.

4 Incorrectness Logic in KAT

Incorrectness logic IL has been introduced by O'Hearn [25] as a natural under-approximating counterpart of the pivotal Hoare correctness logic [16], and quickly attracted a lot of research interest [20,26,27,28,32]. Incorrectness logic distinguishes two postconditions corresponding to normal and erroneous/abnormal program termination. Here, we generalize the algebraic formulation of our LCK logic to support abnormal termination. We follow the approach of Möller, O'Hearn and Hoare [23], namely, each language term is interpreted as a pair of KAT elements which model the normal and abnormal execution. The evaluation function has type $u : \mathrm{Atom} \to (K \times K)$, while the interpretation function has type $[\![\cdot]\!]_u : T_{\Sigma,B} \to (K \times K)$. As a shorthand $[\![\cdot]\!]_u$ can be subscripted with **ok** or **err** to denote, resp., its first normal and second erroneous component. The definition is as follows:

$$\begin{aligned}
[\![\mathtt{a}]\!]_u &\triangleq u(\mathtt{a}) \\
[\![\mathtt{t}_1 + \mathtt{t}_2]\!]_u &\triangleq ([\![\mathtt{t}_1]\!]_{u_{\mathbf{ok}}} + [\![\mathtt{t}_2]\!]_{u_{\mathbf{ok}}}, [\![\mathtt{t}_1]\!]_{u_{\mathbf{err}}} + [\![\mathtt{t}_2]\!]_{u_{\mathbf{err}}}) \\
[\![\mathtt{t}_1 \cdot \mathtt{t}_2]\!]_u &\triangleq ([\![\mathtt{t}_1]\!]_{u_{\mathbf{ok}}} \cdot [\![\mathtt{t}_2]\!]_{u_{\mathbf{ok}}}, [\![\mathtt{t}_1]\!]_{u_{\mathbf{err}}} + [\![\mathtt{t}_1]\!]_{u_{\mathbf{ok}}} \cdot [\![\mathtt{t}_2]\!]_{u_{\mathbf{err}}}) \\
[\![\mathtt{t}^*]\!]_u &\triangleq ([\![\mathtt{t}]\!]^*_{u_{\mathbf{ok}}}, [\![\mathtt{t}]\!]^*_{u_{\mathbf{ok}}} \cdot [\![\mathtt{t}]\!]_{u_{\mathbf{err}}})
\end{aligned} \tag{5}$$

Following the original definition of IL, the precondition encodes an **ok** condition only, while the postcondition contains both an **ok** and an **err** component. Hence, the latter is given by a pair $(p, q) \in \mathsf{test}(K) \times \mathsf{test}(K)$, typically denoted by $\mathbf{ok} : p, \mathbf{err} : q$. The concrete semantics $[\![\cdot]\!] : T_{\Sigma,B} \to (\mathsf{test}(K) \to (\mathsf{test}(K) \times \mathsf{test}(K)))$ is defined as

$$[\![\mathtt{t}]\!]p \triangleq \mathbf{ok} : \langle [\![\mathtt{t}]\!]_{u_{\mathbf{ok}}} | p, \mathbf{err} : \langle [\![\mathtt{t}]\!]_{u_{\mathbf{err}}} | p$$

To refer to one of its components, $[\![\cdot]\!]$ can be subscripted with **ok** or **err** , e.g., $[\![\mathtt{t}]\!]_{\mathbf{ok}}\, p$.

Given a Kleene abstract domain A on K, the corresponding abstract semantics $[\![\cdot]\!]^\sharp_A : T_{\Sigma,B} \to (A \to (A \times A))$ is defined as follows:

$$\begin{aligned}
[\![\mathtt{a}]\!]^\sharp_A p^\sharp &\triangleq \mathbf{ok} : \alpha([\![\mathtt{a}]\!]_{\mathbf{ok}}\, \gamma(p^\sharp)), \mathbf{err} : \alpha([\![\mathtt{a}]\!]_{\mathbf{err}}\, \gamma(p^\sharp)) \\
[\![\mathtt{t}_1 + \mathtt{t}_2]\!]^\sharp_A p^\sharp &\triangleq \mathbf{ok} : [\![\mathtt{t}_1]\!]^\sharp_{A_{\mathbf{ok}}} p^\sharp + [\![\mathtt{t}_2]\!]^\sharp_{A_{\mathbf{ok}}} p^\sharp, \mathbf{err} : [\![\mathtt{t}_1]\!]^\sharp_{A_{\mathbf{err}}} p^\sharp + [\![\mathtt{t}_2]\!]^\sharp_{A_{\mathbf{err}}} p^\sharp \\
[\![\mathtt{t}_1 \cdot \mathtt{t}_2]\!]^\sharp_A p^\sharp &\triangleq \mathbf{ok} : [\![\mathtt{t}_2]\!]^\sharp_{A_{\mathbf{ok}}} ([\![\mathtt{t}_1]\!]^\sharp_{A_{\mathbf{ok}}} p^\sharp), \mathbf{err} : [\![\mathtt{t}_1]\!]^\sharp_{A_{\mathbf{err}}} p^\sharp + [\![\mathtt{t}_2]\!]^\sharp_{A_{\mathbf{err}}} ([\![\mathtt{t}_1]\!]^\sharp_{A_{\mathbf{ok}}} p^\sharp) \\
[\![\mathtt{t}^*]\!]^\sharp_A p^\sharp &\triangleq \mathbf{ok} : \bigvee_{n\in\mathbb{N}} ([\![\mathtt{t}]\!]^\sharp_{A_{\mathbf{ok}}})^n p^\sharp, \mathbf{err} : [\![\mathtt{t}]\!]^\sharp_{A_{\mathbf{err}}} \bigvee_{n\in\mathbb{N}} ([\![\mathtt{t}]\!]^\sharp_{A_{\mathbf{ok}}})^n p^\sharp
\end{aligned} \tag{6}$$

The **ok** part coincides with the semantics of LCK, while the **err** component puts in place some differences. In particular, the composition exhibits a short-circuiting behavior, meaning that an error in the first command aborts the execution without executing the second one, while the Kleene star allows an error to occur after some error-free iterations. It is straightforward to check that this definition of abstract semantics is monotonic and sound.

The proof system LCK can be extended with incorrectness triples. In particular, a triple $[p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r]$ is valid if the standard validity conditions hold for both **ok** and **err** .

Definition 4.1 (Incorrectness Triple). Let K be a CTC bdKAT K and $T_{\Sigma,B}$ be a language interpreted on K. An *incorrectness triple* is either $[p]\ \mathtt{t}\ [\mathbf{ok} : q]$ or $[p]\ \mathtt{t}\ [\mathbf{err} : r]$, where $p, q, r \in \mathsf{test}(K)$ and $\mathtt{t} \in T_{\Sigma,B}$.
Let A be a Kleene abstract domain on K with abstraction map $\alpha : \mathsf{test}(K) \to A$.

- The triple $[p]\ \mathtt{t}\ [\mathbf{ok} : q]$ is valid if: (1) $q \leq [\![\mathtt{t}]\!]_{\mathbf{ok}}\, p$, and (2) $[\![\mathtt{t}]\!]^{\sharp}_{A_{\mathbf{ok}}}\, \alpha(p) = \alpha(q) = \alpha([\![\mathtt{t}]\!]_{\mathbf{ok}}\, p)$.
- The triple $[p]\ \mathtt{t}\ [\mathbf{err} : r]$ is valid if: (1) $r \leq [\![\mathtt{t}]\!]_{\mathbf{err}}\, p$, and (2) $[\![\mathtt{t}]\!]^{\sharp}_{A_{\mathbf{err}}}\, \alpha(p) = \alpha(r) = \alpha([\![\mathtt{t}]\!]_{\mathbf{err}}\, p)$.
- A triple $[p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r]$ is valid when both $[p]\ \mathtt{t}\ [\mathbf{ok} : q]$ and $[p]\ \mathtt{t}\ [\mathbf{err} : r]$ are valid. In particular, if $q = r$ then the triple $[p]\ \mathtt{t}\ [\boldsymbol{\epsilon} : q]$ is valid. □

The proof system LCIL_A defining the local completeness incorrectness logic is given in Figure 3.

Theorem 4.2 (Logical Soundness of LCIL_A). *The triples provable in LCIL_A are valid.*

Furthermore, it turns out that LCIL_A is logically complete.

Theorem 4.3 (Logical Completeness of LCIL_A). *Let A be a Kleene abstract domain on a CTC bdKAT K and $T_{\Sigma,B}$ be a language interpreted on K. Assume that the atoms in $\mathtt{t} \in T_{\Sigma,B}$ are globally complete, i.e., for all $\mathtt{a} \in \mathrm{Atom}(\mathtt{t})$, $\mathbb{C}^A([\![\mathtt{a}]\!]_{\mathbf{ok}})$ and $\mathbb{C}^A([\![\mathtt{a}]\!]_{\mathbf{err}})$ hold. If $[p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r]$ is valid, then it is provable in LCIL_A.*

Example 4.4. Consider a relational bdKAT $K \triangleq \wp(\mathbb{Z} \times \mathbb{Z})$ on the set of integers $\mathbb{Z}$, where $\mathbf{1}_K \triangleq \{\langle z, z\rangle \mid z \in \mathbb{Z}\}$ and $\mathbf{0}_K \triangleq \varnothing$, and the standard integer interval abstraction Int [8, 9]. Let us consider a language with primitive actions $\Sigma \triangleq \{\mathtt{x} := \mathtt{x} + 1, \mathtt{err}\}$. The evaluation function $u : \Sigma \cup B \to K_{\mathbf{ok}} \times K_{\mathbf{err}}$ is defined as expected:

$$u(\mathtt{x} := \mathtt{x} + 1) = (\{\langle z, z + 1\rangle \mid z \in \mathbb{Z}\}, \mathbf{0}_K), \qquad u(\mathtt{err}) = (\mathbf{0}_K, \mathbf{1}_K).$$

We study the correctness of the program $\mathtt{r} \equiv ((\mathtt{x} := \mathtt{x} + 1) + \mathtt{err})^*$, for the precondition $p \triangleq \{\langle 0, 0\rangle, \langle 2, 2\rangle\}$ and the specification $\mathrm{Spec} \triangleq (\mathbf{ok} : \{\langle z, z\rangle \mid z \geq 0\}, \mathbf{err} : \mathbf{0}_K)$. Let us define an auxiliary sequence of tests $p_n \triangleq \{\langle n, n\rangle, \langle n + 2, n + 2\rangle\}$ and $s \triangleq \{\langle z, z\rangle \mid z \geq 0\}$.

$$\frac{\mathtt{a} \in \mathrm{Atom} \qquad \mathbb{C}^A_p(\llbracket \mathtt{a} \rrbracket_{\mathbf{ok}}) \qquad \mathbb{C}^A_p(\llbracket \mathtt{a} \rrbracket_{\mathbf{err}})}{\vdash^{\mathrm{K}}_A [p]\ \mathtt{a}\ [\mathbf{ok} : \llbracket \mathtt{a} \rrbracket_{\mathbf{ok}}\, p][\mathbf{err} : \llbracket \mathtt{a} \rrbracket_{\mathbf{err}}\, p]}\ \text{(transfer)}$$

$$\frac{p' \leq p \leq A(p') \qquad \vdash^{\mathrm{K}}_A [p']\ \mathtt{t}\ [\boldsymbol{\epsilon} : q'] \qquad q \leq q' \leq A(q)}{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}\ [\boldsymbol{\epsilon} : q]}\ \text{(relax)}$$

$$\frac{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_1\ [\mathbf{ok} : r] \qquad \vdash^{\mathrm{K}}_A [r]\ \mathtt{t}_2\ [\mathbf{ok} : q]}{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_1 \cdot \mathtt{t}_2\ [\mathbf{ok} : q]}\ \text{(seq-ok)}$$

$$\frac{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_1\ [\mathbf{ok} : q][\mathbf{err} : r] \qquad \vdash^{\mathrm{K}}_A [q]\ \mathtt{t}_2\ [\mathbf{err} : s]}{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_1 \cdot \mathtt{t}_2\ [\mathbf{err} : r + s]}\ \text{(seq-err)}$$

$$\frac{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}^*\ [\mathbf{ok} : q] \qquad \vdash^{\mathrm{K}}_A [q]\ \mathtt{t}\ [\mathbf{err} : r]}{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}^*\ [\mathbf{err} : r]}\ \text{(rec-err)}$$

$$\frac{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_1\ [\boldsymbol{\epsilon} : q_1] \qquad \vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_2\ [\boldsymbol{\epsilon} : q_2]}{\vdash^{\mathrm{K}}_A [p]\ \mathtt{t}_1 + \mathtt{t}_2\ [\boldsymbol{\epsilon} : q_1 + q_2]}\ \text{(join)}$$

$$\frac{\forall n \in \mathbb{N}.\ \vdash^{\mathrm{K}}_A [p_n]\ \mathtt{t}\ [\mathbf{ok} : p_{n+1}]}{\vdash^{\mathrm{K}}_A [p_0]\ \mathtt{t}^*\ [\mathbf{ok} : \bigvee_{n \in \mathbb{N}} p_n]}\ \text{(limit)}$$

Fig. 3. Proof system LCIL_A.

We can easily check the local completeness of the atoms by exploiting the following characterization (see the full version [22]) of the backward diamond operator in a relational KAT K on a set X where $\mathsf{test}(K) = \wp(\{(x,x) \mid x \in X\})$: for all $a \in K$ and $p \in \mathsf{test}(K)$,

$$\langle a | p = \{(y,y) \mid \exists x \in X.\ (x,x) \in p, (x,y) \in a\}.$$

We therefore have the following derivation in $\mathrm{LCIL}_{\mathrm{Int}}$ for $\mathtt{r}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbb{C}^{\mathrm{Int}}_{p_n}(\llbracket \mathtt{x} := \mathtt{x}+1 \rrbracket_{\mathbf{ok}}) \qquad \mathbb{C}^{\mathrm{Int}}_{p_n}(\llbracket \mathtt{x} := \mathtt{x}+1 \rrbracket_{\mathbf{err}})}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [p_n]\ \mathtt{x} := \mathtt{x}+1\ [\mathbf{ok} : p_{n+1}]}\ \text{(transfer)} \qquad \dfrac{\mathbb{C}^{\mathrm{Int}}_{p_n}(\llbracket \mathtt{err} \rrbracket_{\mathbf{ok}}) \qquad \mathbb{C}^{\mathrm{Int}}_{p_n}(\llbracket \mathtt{err} \rrbracket_{\mathbf{err}})}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [p_n]\ \mathtt{err}\ [\mathbf{ok} : 0]}\ \text{(transfer)}}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [p_n]\ (\mathtt{x} := \mathtt{x}+1) + \mathtt{err}\ [\mathbf{ok} : p_{n+1}]}\ \text{(choice)}}{\dagger}\ \text{(limit)}$$

$$\dfrac{\dfrac{\mathbb{C}^{\mathrm{Int}}_{s}(\llbracket \mathtt{x} := \mathtt{x}+1 \rrbracket_{\mathbf{ok}}) \qquad \mathbb{C}^{\mathrm{Int}}_{s}(\llbracket \mathtt{x} := \mathtt{x}+1 \rrbracket_{\mathbf{err}})}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [s]\ \mathtt{x} := \mathtt{x}+1\ [\mathbf{err} : 0]}\ \text{(transfer)} \qquad \dfrac{\mathbb{C}^{\mathrm{Int}}_{s}(\llbracket \mathtt{err} \rrbracket_{\mathbf{ok}}) \qquad \mathbb{C}^{\mathrm{Int}}_{s}(\llbracket \mathtt{err} \rrbracket_{\mathbf{err}})}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [s]\ \mathtt{err}\ [\mathbf{err} : s]}\ \text{(transfer)}}{\ddagger}\ \text{(choice)}$$

$$\dfrac{\dfrac{\dagger}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [p_0]\ ((\mathtt{x} := \mathtt{x}+1) + \mathtt{err})^*\ [\mathbf{ok} : s]}\ \text{(limit)} \qquad \dfrac{\ddagger}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [s]\ (\mathtt{x} := \mathtt{x}+1) + \mathtt{err}\ [\mathbf{err} : s]}\ \text{(choice)}}{\vdash^{\mathrm{K}}_{\mathrm{Int}} [p_0]\ ((\mathtt{x} := \mathtt{x}+1) + \mathtt{err})^*\ [\mathbf{err} : s]}\ \text{(rec-err)}$$

By soundness of $\mathrm{LCIL}_{\mathrm{Int}}$ in Theorem 4.2, the program $\mathtt{r}$ satisfies the **ok** part of Spec because

$$\llbracket \mathtt{r} \rrbracket_{\mathbf{ok}}\, p \subseteq \mathrm{Int}(s) = s \subseteq s = \mathrm{Spec}_{\mathbf{ok}}\ .$$

However, the **err** part is not satisfied as $\mathrm{Int}(s) = s \not\subseteq \varnothing = \mathbf{0}_K = \mathrm{Spec}_{\mathbf{err}}$. Moreover, $\mathrm{LCIL}_{\mathrm{Int}}$ also catches true alerts as $s \smallsetminus \mathrm{Spec}_{\mathbf{err}} = s$. □

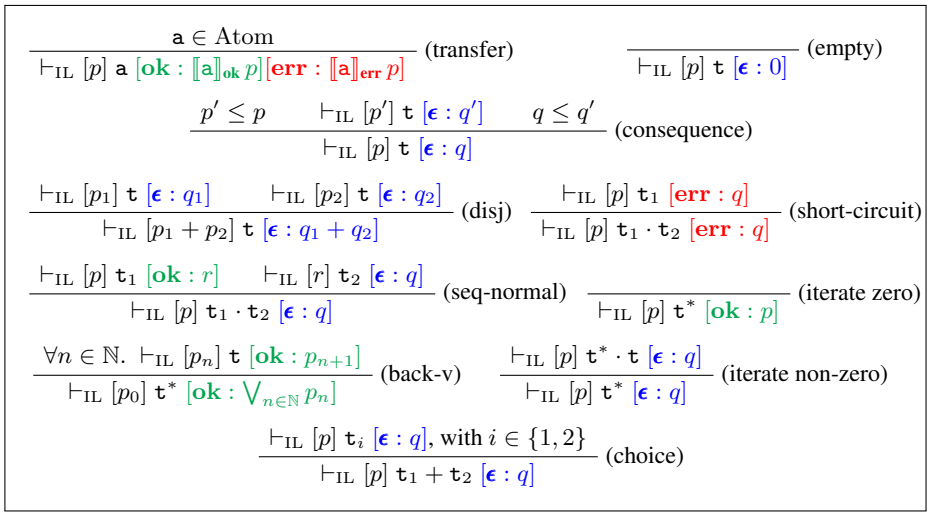

Fig. 4. Proof system IL.

4.1 Relationship with Incorrectness logic

Section 3.6 has shown that LCK yields a generalization of UL. The same can be done for IL, i.e., we prove that LCIL_A with incorrectness triples generalizes the incorrectness logic of [25]. For the sake of clarity, we recall in Figure 4 an algebraic version of IL. Analogously to the reduction of Theorem 3.7, this generalization is obtained by letting $A = A_{tr}$, where A_{tr} is the trivial abstract domain.

Theorem 4.5. *Let K be a CTC bdKAT and $T_{\Sigma,B}$ a language interpreted on K. For any $p, q \in \mathsf{test}(K)$, $\mathtt{t} \in T_{\Sigma,B}$,*

$$\vdash^{\mathrm{K}}_{A_{tr}} [p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r] \quad \Leftrightarrow \quad \vdash_{\mathrm{IL}} [p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r].$$

The abstraction map $\alpha = \lambda x.\top$ of A_{tr} makes the validity of a triple trivially true. In particular, $[\![\mathtt{t}]\!]^\sharp_{A_{tr}\,\mathbf{ok}}\, \alpha(p) = \top = \alpha(q) = \alpha([\![\mathtt{t}]\!]_{\mathbf{ok}}\, p)$ and $[\![\mathtt{t}]\!]^\sharp_{A_{tr}\,\mathbf{err}}\, \alpha(p) = \top = \alpha(q) = \alpha([\![\mathtt{t}]\!]_{\mathbf{err}}\, p)$ hold. As a consequence, we obtain that

$$\models^{\mathrm{K}}_{A_{tr}} [p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r] \quad \Leftrightarrow \quad \models_{\mathrm{IL}} [p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r] \tag{7}$$

By this equivalence (7) and Theorems 4.2 and 4.3, we can thus retrieve the logical soundness and completeness of IL as a consequence of the one of $\mathrm{LCIL}_{A_{tr}}$.

Corollary 4.6. *Let K be a CTC bdKAT and $T_{\Sigma,B}$ a language interpreted on K. For any $p, q \in \mathsf{test}(K)$, $\mathtt{t} \in T_{\Sigma,B}$, $\vdash_{\mathrm{IL}} [p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r] \Leftrightarrow \models_{\mathrm{IL}} [p]\ \mathtt{t}\ [\mathbf{ok} : q][\mathbf{err} : r]$.*

5 Local Completeness Logic in TopKAT

We have shown in Section 3 how KAT extended with a modal backward-diamond operator allows us to interpret and represent the local completeness program logic.

This result follows the approach by Moller, O'Hearn and Hoare [23], who leverage a backward-diamond operator in their KAT interpretation of correctness/incorrectness logics. On the other hand, Zhang, de Amorim and Gaboardi [33] have recently shown that incorrectness logic can be formulated for a standard KAT, provided that it contains a top element, thus giving rise to a so-called TopKAT. In particular, [33] observed that a TopKAT is enough to express the codomain of relational KATs. In this section, we take a similar path in studying an alternative formulation of local completeness logic based on a TopKAT.

5.1 Abstracting TopKATs

We expect that the base case of abstract semantics $[\![\mathtt{a}]\!]^\sharp_A p^\sharp$ for a basic action $\mathtt{a} \in \mathrm{Atom}$ is defined as best correct approximation in A of the concrete semantics of $\mathtt{a}$ on the concretization of $p^\sharp$. In a bdKAT this is achieved in definition (2) through its backward-diamond operator, which is crucially used in (1) to define the strongest postcondition as $[\![\mathtt{a}]\!]^K \gamma(p^\sharp) = \langle [\![\mathtt{a}]\!]_u | \gamma(p^\sharp)$. Zhang et al. [33] observed that in a relational model of KAT, the codomain inclusion $\mathrm{cod}(q) \subseteq \mathrm{cod}(pa)$ defining the meaning of an under-approximation triple $[p]\, a\, [q]$ can be expressed in a TopKAT as the inequality $\top q \leq \top pa$, thus hinting that this latter condition could be taken as definition of validity of incorrectness triples in a TopKAT. We follow here a similar approach by considering the element $\top p [\![\mathtt{a}]\!]_u$ as a proxy for strongest postconditions in a TopKAT. It is worth noticing that while in a bdKAT a strongest postcondition $\langle [\![\mathtt{a}]\!]_u | p$ is always a test, in a TopKAT K, given $p \in \mathsf{test}(K)$ and a term $\mathtt{t} \in T_{\Sigma,B}$, it is not guaranteed that there exists a test $q \in \mathsf{test}(K)$ such that $\top p [\![\mathtt{t}]\!]_u = \top q$, as shown by the following example.

Example 5.1 (Strongest Postconditions in TopKAT). Consider the Kleene algebra $A_3 = \{0, 1, a\}$ consisting of 3 elements and characterized by Conway [7, Chapter 12]. This algebra can be lifted to a KAT by letting $\mathsf{test}(A_3) \triangleq \{0, 1\}$ and defining the KAT operators as follows:

$+$	0	1	a
0	0	1	a
1	1	1	1
a	a	1	a

$\cdot$	0	1	a
0	0	0	0
1	0	1	a
a	0	a	0

$0^* \triangleq 1$	$1^* \triangleq 1$	$a^* \triangleq 1$

We have that $1 \geq a$ and $1 \geq 0$, because $1 + a = 1$ and $1 + 0 = 1$, so that A_3 is a TopKAT with $\top = 1$. Moreover, $\top \cdot 1 \cdot a = 1 \cdot 1 \cdot a = a$, whereas there exists no $q \in \mathsf{test}(A_3)$ satisfying $\top \cdot q = a$. Indeed, $\top \cdot 1 = 1 \cdot 1 = 1 \neq a$ and $\top \cdot 0 = 0 \neq a$. □

In general, the lack of such a $q \in \mathsf{test}(K)$ implies that the abstract domain cannot be defined as an abstraction of the set of topped-tests $\{\top p \mid p \in \mathsf{test}(K)\}$, because in this case we could miss the abstraction $\alpha(\top p [\![\mathtt{a}]\!]_u)$. To settle this issue, an abstract domain must provide an approximation of the larger set

$$\mathsf{top}(K) \triangleq \{\top a \mid a \in K\}$$

which contains all the multiplicative elements of type $\top a$.

Definition 5.2 (Top Kleene Abstract Domain). A poset $(A, \leq)$ is a *top Kleene abstract domain* of a TopKAT K if:

(i) There exists a Galois insertion, defined by $\gamma : A \to \mathtt{top}(K)$ and $\alpha : \mathtt{top}(K) \to A$, of the poset $(A, \leq_A)$ into the poset $(\mathtt{top}(K), \leq_K)$;
(ii) A is countably-complete. □

The abstract semantic function $[\![\cdot]\!]^\sharp_A : T_{\Sigma,B} \to (A \to A)$ on a top Kleene abstraction A can be therefore defined for the base case $\mathtt{a} \in \mathrm{Atom}$ as

$$[\![\mathtt{a}]\!]^\sharp_A p^\sharp \triangleq \alpha(\gamma(p^\sharp)[\![\mathtt{a}]\!]_u),$$

while the remaining inductive cases are defined as in (2) for Kleene abstractions. The monotonicity and soundness properties of this abstract semantics hold, provided that the TopKAT is *-continuous[1], which is referred to as TopKAT*.

Theorem 5.3 (Soundness of TopKAT Abstract Semantics). *Let A be a Kleene abstraction of a TopKAT* K and $T_{\Sigma,B}$ be a language interpreted on K. For all $p^\sharp, q^\sharp \in A$, $a \in K$ and $\mathtt{t} \in T_{\Sigma,B}$:*

$$p^\sharp \leq_A q^\sharp \Rightarrow [\![\mathtt{t}]\!]^\sharp_A p^\sharp \leq_A [\![\mathtt{t}]\!]^\sharp_A q^\sharp \qquad \text{(monotonicity)}$$
$$\alpha(\top a[\![\mathtt{t}]\!]_u) \leq_A [\![\mathtt{t}]\!]^\sharp_A \alpha(\top a) \qquad \text{(soundness)}$$

5.2 Local Completeness Logic on TopKAT

Completeness and triple validity are adapted to the TopKAT framework as follows. Given a Top Kleene abstract domain A on a TopKAT* K, A is defined to be locally complete for $a \in K$ on an element $b \in K$, denoted by $\mathbb{C}^A_b(a)$, when

$$A(\top ba) = A(A(\top b)a)$$

holds. Moreover, A is globally complete for a, denoted by $\mathbb{C}^A(a)$, when it is locally complete for any $b \in K$.

Likewise, a triple $[a]\ \mathtt{t}\ [b]$, with $a, b \in K$ and $\mathtt{t} \in T_{\Sigma,B}$, is valid, denoted by $\models^{\mathrm{TK}}_A [a]\ \mathtt{t}\ [b]$, when:

$$(1)\ \top b \leq \top a[\![\mathtt{t}]\!]_u; \quad (2)\ [\![\mathtt{t}]\!]^\sharp_A \alpha(\top a) = \alpha(\top b) = \alpha(\top a[\![\mathtt{t}]\!]_u).$$

The corresponding proof system, denoted by LCTK_A, has the same rules of LCK_A in Figure 1 except (transfer), (relax) and (iterate) which are modified as follows:

$$\frac{\mathtt{c} \in \mathrm{Atom} \qquad \mathbb{C}^A_a([\![\mathtt{c}]\!]_u)}{\vdash^{\mathrm{TK}}_A [a]\ \mathtt{c}\ [a[\![\mathtt{c}]\!]_u]}\ \text{(transfer)}$$

$$\frac{\top a' \leq \top a \leq A(\top a') \qquad \vdash^{\mathrm{TK}}_A [a']\ \mathtt{t}\ [b'] \qquad \top b \leq \top b' \leq A(\top b)}{\vdash^{\mathrm{TK}}_A [a]\ \mathtt{t}\ [b]}\ \text{(relax)}$$

[1] This condition plays a role similar to the CTC condition for bdKATs.

$$\frac{\vdash_A^{\mathrm{TK}} [a]\ \mathtt{t}\ [b] \qquad \top b \leq A(\top a)}{\vdash_A^{\mathrm{TK}} [a]\ \mathtt{t}^*\ [a+b]}\ \text{(iterate)}$$

This incarnation LCTK_A of local completeness logic for TopKAT* turns out to be logically sound and, under additional hypotheses, complete.

Theorem 5.4 (Logical Soundness of $\vdash_A^{\mathrm{TK}}$). *If* $\vdash_A^{\mathrm{TK}} [a]\ \mathtt{t}\ [b]$ *then*

(i) $\top b \leq \top a[\![\mathtt{t}]\!]_u$;
(ii) $[\![\mathtt{t}]\!]_A^\sharp \alpha(\top a) = \alpha(\top b) = \alpha(\top a[\![\mathtt{t}]\!]_u)$.

Logical completeness needs the following additional conditions:

(a) Likewise LCK_A, the same infinitary rule for Kleene star:

$$\frac{\forall n \in \mathbb{N}.\ \vdash_A^{\mathrm{TK}} [a_n]\ \mathtt{t}\ [a_{n+1}]}{\vdash_A^{\mathrm{TK}} [a_0]\ \mathtt{t}^*\ [\bigvee_{n\in\mathbb{N}} a_n]}\ \text{(limit)}$$

where we assume that:
- $\bigvee_{n\in\mathbb{N}} a_n$ always exists. Let us remark that for bdKAT, such explicit condition was not needed, as it was entailed by the CTC requirement on the KAT.
- $\top$ distributes over $\bigvee_{n\in\mathbb{N}} a_n$, i.e., $\top \bigvee_{n\in\mathbb{N}} a_n = \bigvee_{n\in\mathbb{N}} \top a_n$.

It turns out that this additional rule (limit) is sound (see the full version [22]).

(b) Global completeness of all the primitive actions and tests occurring in the program.

Theorem 5.5 (Logical Completeness of $\vdash_A^{\mathrm{TK}}$). *Assume that conditions* (a) *and* (b) *hold. If* $\models_A^{\mathrm{TK}} [a]\ \mathtt{t}\ [b]$ *then* $\vdash_A^{\mathrm{TK}} [a]\ \mathtt{t}\ [b]$.

Let us describe an example of derivation in LCTK_A.

Example 5.6. Consider a relational KAT $K = \wp(\mathbb{Z}\times\mathbb{Z})$ on the set of integers $\mathbb{Z}$, where $\mathbf{1}_K \triangleq \{(z,z) \mid z \in \mathbb{Z}\}$ and $\mathbf{0}_K \triangleq \varnothing$. Notice that $\mathbb{Z}\times\mathbb{Z} \in K$ is the top element $\top$ of K, meaning that K is a TopKAT. Let us consider a language with primitive actions $\Sigma = \{\mathtt{x} := \mathtt{x}+1\}$ and primitive tests $B = \{\mathtt{x} \geq 0, \mathtt{x} < 0\}$. The evaluation function $u : \Sigma \cup B \to K$ is defined as expected by the following relations:

$$u(\mathtt{x} := \mathtt{x}+1) \triangleq \{(z, z+1) \mid z \in \mathbb{Z}\},$$
$$u(\mathtt{x} \geq 0) \triangleq \{(z,z) \mid z \in \mathbb{Z}, z \geq 0\},$$
$$u(\mathtt{x} < 0) \triangleq \{(z,z) \mid z \in \mathbb{Z}, z < 0\}.$$

Consider the following sign abstraction $\mathrm{Sign} \triangleq \{\mathbb{Z}, \mathbb{Z}_{\leq 0}, \mathbb{Z}_{\neq 0}, \mathbb{Z}_{\geq 0}, \mathbb{Z}_{<0}, \mathbb{Z}_{=0}, \mathbb{Z}_{>0}, \varnothing\}$ of $\wp(\mathbb{Z})$, whose abstraction and concretization maps are straightforward. Let us verify that the program

$$\mathtt{r} \equiv \big((\mathtt{x} \geq 0) \cdot (\mathtt{x} := \mathtt{x}+1)\big)^* \cdot (\mathtt{x} < 0)$$

does not terminate with precondition $p \triangleq \{(0,0),(10,10)\}$, i.e., we prove the specification $\mathrm{Spec} \triangleq \varnothing$. Let us define the following auxiliary elements: $q \triangleq \{(1,1),(11,11)\}$, $s \triangleq p+q$, $t_{\geq 0} \triangleq \{(x,z) \mid x \in \mathbb{Z}, z \in \mathbb{Z}_{\geq 0}\}$, and observe that $\mathrm{Sign}(t_{\geq 0}) = t_{\geq 0}$.

The following local completeness conditions for the atoms hold:

$$\alpha(\mathrm{Sign}(\top p)[\![\mathtt{x} \geq \mathtt{0}]\!]_u) = \alpha(t_{\geq 0}[\![\mathtt{x} \geq \mathtt{0}]\!]_u) = \mathbb{Z}_{\geq 0} = \alpha(\top p[\![\mathtt{x} \geq \mathtt{0}]\!]_u),$$
$$\alpha(\mathrm{Sign}(\top p)[\![\mathtt{x} := \mathtt{x}+\mathtt{1}]\!]_u) = \alpha(t_{\geq 0}[\![\mathtt{x} := \mathtt{x}+\mathtt{1}]\!]_u) = \mathbb{Z}_{>0} = \alpha(\top p[\![\mathtt{x} := \mathtt{x}+\mathtt{1}]\!]_u),$$
$$\alpha(\mathrm{Sign}(\top s)[\![\mathtt{x} < \mathtt{0}]\!]_u) = \alpha(t_{\geq 0}[\![\mathtt{x} < \mathtt{0}]\!]_u) = \varnothing = \alpha(\top s[\![\mathtt{x} < \mathtt{0}]\!]_u).$$

Moreover, we also have that:

$$\top q = \{(x, z) \mid x \in \mathbb{Z}, z \in \{1, 11\}\} \leq t_{\geq 0} = \mathrm{Sign}(\top p).$$

The following derivation shows that the triple $[p]\ \mathtt{r}\ [\mathbf{0}_K]$ is provable in $\mathrm{LCTK}_{\mathrm{Sign}}$:

$$\dfrac{\dfrac{\dfrac{\mathbb{C}_p^{\mathrm{Sign}}([\![\mathtt{x} \geq \mathtt{0}]\!]_u)}{\vdash^{\mathrm{TK}}_{\mathrm{Sign}} [p]\ \mathtt{x} \geq \mathtt{0}\ [p]}\ \text{(transfer)} \quad \dfrac{\mathbb{C}_p^{\mathrm{Sign}}([\![\mathtt{x} := \mathtt{x}+\mathtt{1}]\!]_u)}{\vdash^{\mathrm{TK}}_{\mathrm{Sign}} [p]\ \mathtt{x} := \mathtt{x}+\mathtt{1}\ [q]}\ \text{(transfer)}}{\vdash^{\mathrm{TK}}_{\mathrm{Sign}} [p]\ (\mathtt{x} \geq \mathtt{0}) \cdot (\mathtt{x} := \mathtt{x}+\mathtt{1})\ [q]}\ \text{(seq)} \quad \top q \leq \mathrm{Sign}(\top p)}{\vdash^{\mathrm{TK}}_{\mathrm{Sign}} [p]\ ((\mathtt{x} \geq \mathtt{0}) \cdot (\mathtt{x} := \mathtt{x}+\mathtt{1}))^* [s]}\ \text{(iterate)} \quad \dfrac{\mathbb{C}_s^{\mathrm{Sign}}([\![\mathtt{x} < \mathtt{0}]\!]_u)}{\vdash^{\mathrm{TK}}_{\mathrm{Sign}} [s]\ \mathtt{x} < \mathtt{0}\ [\mathbf{0}_K]}\ \text{(transfer)}$$
$$\vdash^{\mathrm{TK}}_{\mathrm{Sign}} [p]\ ((\mathtt{x} \geq \mathtt{0}) \cdot (\mathtt{x} := \mathtt{x}+\mathtt{1}))^* \cdot (\mathtt{x} < \mathtt{0})\ [\mathbf{0}_K]\ \text{(seq)}$$

By Theorem 5.4, we have that $\top \mathbf{0}_K \subseteq \top p[\![\mathtt{r}]\!]_u \subseteq \mathrm{Sign}(\top \mathbf{0}_K) = \varnothing$, meaning that the program does not terminate, and Spec is satisfied as $\top p[\![\mathtt{r}]\!]_u = \varnothing = \top \mathrm{Spec}$. □

5.3 Relationship with Under-Approximation Logic

We have shown in Section 3.6 that the backward-diamond formulation of LCK generalizes UL. The same can be done for the TopKAT formulation. A TopKAT version of the UL proof system has been already proposed in [33, Figure 6]. The reduction here considered refers to such system, with the following minor differences:

- We consider only propositional fragments of the logic, meaning that the rules (assume) and (identity) are replaced by the following single (transfer) rule:

$$\frac{\mathtt{c} \in \mathrm{Atom}}{\vdash_{\mathrm{UL}} [a]\ \mathtt{c}\ [a[\![\mathtt{c}]\!]_u]}\ \text{(transfer)}$$

- The premises of the (consequence) rule in [33, Figure 6], $b \leq b'$ and $c' \leq c$, are relaxed to $\top b \leq \top b'$ and $\top c' \leq \top c$. Notice that the former implies the latter. Furthermore, the soundness proof of [33, Theorem 4] is not affected by this change, because $(\top b' \geq \top b \wedge \top c \geq \top c' \wedge \top bp \geq c) \Rightarrow \top b'p \geq \top bp \geq \top c \geq \top c'$, and, by [33, Theorem 3], it holds that $\top b'p \geq \top c'$ entails $\top b'p \geq c'$.
- The (limit) rules of LCTK_A and UL differ on the distributivity condition. We assume that distributivity also holds in UL.

By instantiating to the trivial abstract domain A_{tr}, it turns out that the two proof systems become equivalent.

Theorem 5.7 ($\mathrm{LCTK}_{A_{tr}} \equiv \mathrm{UL}$)**.** *Let K be a TopKAT*. For any $a, b \in K$, $\mathtt{t} \in T_{\Sigma,B}$:*

$$\vdash^{\mathrm{TK}}_{A_{tr}} [a]\ \mathtt{t}\ [b] \quad \Leftrightarrow \quad \vdash_{\mathrm{UL}} [a]\ \mathtt{t}\ [b].$$

In turn, the logical soundness and completeness of UL can be retrieved as a consequence of those of LCTK.

Corollary 5.8. *Under the same hypotheses of Theorem 5.7, the proof system* UL *is sound and complete, that is,* $\vdash_{\mathrm{UL}} [p]\ \mathtt{t}\ [q] \Leftrightarrow \models_{\mathrm{UL}} [p]\ \mathtt{t}\ [q]$.

Finally, let us mention that the full version [22] also shows how to define an incorrectness logic in TopKAT.

6 Conclusion

This work has shown that the abstract interpretation-based local completeness logic introduced in [4] can be generalized to and interpreted in Kleene algebra with tests. In particular, we proved that this can be achieved both for KATs extended with a modal backward diamond operator playing the role of strongest postcondition, and for KATs endowed with a top element. Our results generalize both the modal [23] and top [33] KAT approaches that encode Hoare correctness and O'Hearn incorrectness logic using different classes of KATs. In particular, our KAT-based logic leverages an abstract interpretation of KAT, a problem that was not studied so far.

Our plan for future work includes, but is not limited to, the following questions.

- For a KAT with top $\top$, following the technical idea underlying the approach by Zhang et al. [33], we defined an abstract domain as an approximation of all the algebraic elements of type $\top \cdot a$, where a is any element of the KAT (cf. Definition 5.2). Although this definition technically works, it is somehow artificial, because the elements $\top \cdot a$ do not carry a clear intuitive meaning. As an interesting future task, we would like to characterize under which conditions an element $\top \cdot a$ coincides with $\top \cdot p$ for some test $p \in \mathsf{test}(K)$, and if such test p is unique.
- This work is a first step towards an *algebraic and equational approach to abstract interpretation*. We envisage that the reasoning made by an abstract interpreter of programs could be made purely equational within a KAT equipped with a suitable collection of axioms. The ambition would be to conceive a notion of *abstract Kleene algebra* (AKA) making this slogan true: *AKA is for the abstract interpretation of programs what KAT is for concrete interpretation of programs*.

Acknowledgements. Francesco Ranzato has been partially funded by the *Italian Ministry of University and Research*, under the PRIN 2017 project no. 201784YSZ5 "AnalysiS of PRogram Analyses (ASPRA)", by *Facebook Research*, under a "Probability and Programming Research Award", and by an *Amazon Research Award* for "AWS Automated Reasoning".

References

1. Anderson, C.J., Foster, N., Guha, A., Jeannin, J.B., Kozen, D., Schlesinger, C., Walker, D.: NetKAT: Semantic foundations for networks. In: Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. p. 113–126. POPL '14, ACM (2014). https://doi.org/10.1145/2535838.2535862

2. Beckett, R., Greenberg, M., Walker, D.: Temporal NetKAT. In: Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016. pp. 386–401. ACM (2016). https://doi.org/10.1145/2908080.2908108
3. Bruni, R., Giacobazzi, R., Gori, R., Garcia-Contreras, I., Pavlovic, D.: Abstract extensionality: on the properties of incomplete abstract interpretations. Proc. ACM Program. Lang. **4**(POPL), 28:1–28:28 (2020). https://doi.org/10.1145/3371096
4. Bruni, R., Giacobazzi, R., Gori, R., Ranzato, F.: A Logic for Locally Complete Abstract Interpretations. In: Proceedings 36th ACM/IEEE Symposium on Logic in Computer Science (LICS 2021). pp. 1–13. IEEE (2021). https://doi.org/10.1109/LICS52264.2021.9470608
5. Bruni, R., Giacobazzi, R., Gori, R., Ranzato, F.: Abstract interpretation repair. In: Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation. p. 426–441. PLDI 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3519939.3523453
6. Cohen, E., Kozen, D., Smith, F.: The complexity of Kleene algebra with tests. Tech. rep., Cornell University, USA (1996), https://www.cs.cornell.edu/~kozen/Papers/ckat
7. Conway, J.: Regular Algebra and Finite Machines. Chapman and Hall mathematics series, Dover Publications (2012)
8. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 1977). pp. 238–252. ACM (1977). https://doi.org/10.1145/512950.512973
9. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL 1979). pp. 269–282. ACM (1979). https://doi.org/10.1145/567752.567778
10. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. ACM Trans. Comput. Logic **7**(4), 798–833 (oct 2006). https://doi.org/10.1145/1183278.1183285
11. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences **18**(2), 194–211 (Apr 1979). https://doi.org/10.1016/0022-0000(79)90046-1
12. Foster, N., Kozen, D., Milano, M., Silva, A., Thompson, L.: A coalgebraic decision procedure for NetKAT. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015. pp. 343–355. ACM (2015). https://doi.org/10.1145/2676726.2677011
13. Giacobazzi, R., Ranzato, F., Scozzari., F.: Making abstract interpretation complete. Journal of the ACM **47**(2), 361–416 (March 2000). https://doi.org/10.1145/333979.333989
14. Giacobazzi, R., Logozzo, F., Ranzato, F.: Analyzing program analyses. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015. pp. 261–273 (2015). https://doi.org/10.1145/2676726.2676987
15. Greenberg, M., Beckett, R., Campbell, E.: Kleene algebra modulo theories: A framework for concrete KATs. In: Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation. p. 594–608. PLDI 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3519939.3523722
16. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM **12**(10), 576–580 (oct 1969). https://doi.org/10.1145/363235.363259

17. Kozen, D.: Kleene algebra with tests. ACM Transactions on Programming Languages and Systems **19**(3), 427–443 (May 1997). https://doi.org/10.1145/256167.256195
18. Kozen, D.: On Hoare logic and Kleene algebra with tests. ACM Trans. Comput. Logic **1**(1), 60–76 (jul 2000). https://doi.org/10.1145/343369.343378
19. Kozen, D., Smith, F.: Kleene algebra with tests: Completeness and decidability. In: Proceedings 10th International Workshop on Computer Science Logic, CSL 1996, Annual Conference of the EACSL. Lecture Notes in Computer Science, vol. 1258, pp. 244–259. Springer (1996). https://doi.org/10.1007/3-540-63172-0_43
20. Le, Q.L., Raad, A., Villard, J., Berdine, J., Dreyer, D., O'Hearn, P.W.: Finding real bugs in big programs with incorrectness logic. Proc. ACM Program. Lang. **6**(OOPSLA1) (apr 2022). https://doi.org/10.1145/3527325
21. Mamouras, K.: Equational theories of abnormal termination based on kleene algebra. In: Proceedings 20th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2017. Lecture Notes in Computer Science, vol. 10203, pp. 88–105 (2017). https://doi.org/10.1007/978-3-662-54458-7_6
22. Milanese, M., Ranzato, F.: Local completeness logic on Kleene algebra with tests. arXiv e-prints arXiv:2205.08128 (2022). https://doi.org/10.48550/ARXIV.2205.08128
23. Möller, B., O'Hearn, P.W., Hoare, T.: On algebra of program correctness and incorrectness. In: Proceedings of the 19th International Conference on Relational and Algebraic Methods in Computer Science, RAMiCS 2021. Lecture Notes in Computer Science, vol. 13027, pp. 325–343. Springer (2021). https://doi.org/10.1007/978-3-030-88701-8_20
24. Möller, B., Struth, G.: Algebras of modal operators and partial correctness. Theoretical Computer Science **351**(2), 221–239 (Feb 2006). https://doi.org/10.1016/j.tcs.2005.09.069
25. O'Hearn, P.W.: Incorrectness logic. Proceedings of the ACM on Programming Languages **4**(POPL), 1–32 (Jan 2020). https://doi.org/10.1145/3371078
26. Poskitt, C.M.: Incorrectness logic for graph programs. In: Proceedings of the 14th International Conference on Graph Transformation, ICGT 2021. Lecture Notes in Computer Science, vol. 12741, pp. 81–101. Springer (2021). https://doi.org/10.1007/978-3-030-78946-6_5
27. Raad, A., Berdine, J., Dang, H., Dreyer, D., O'Hearn, P.W., Villard, J.: Local reasoning about the presence of bugs: Incorrectness separation logic. In: Proceedings 32nd International Conference on Computer Aided Verification, CAV 2020. Lecture Notes in Computer Science, vol. 12225, pp. 225–252. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_14
28. Raad, A., Berdine, J., Dreyer, D., O'Hearn, P.W.: Concurrent incorrectness separation logic. Proc. ACM Program. Lang. **6**(POPL), 1–29 (2022). https://doi.org/10.1145/3498695
29. Ranzato, F.: Complete abstractions everywhere. In: Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI 2013. Lecture Notes in Computer Science, vol. 7737, pp. 15–26. Springer (2013). https://doi.org/10.1007/978-3-642-35873-9_3
30. Smolka, S., Eliopoulos, S.A., Foster, N., Guha, A.: A fast compiler for NetKAT. In: Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015. pp. 328–341. ACM (2015). https://doi.org/10.1145/2784731.2784761
31. Smolka, S., Kumar, P., Foster, N., Kozen, D., Silva, A.: Cantor meets Scott: Semantic foundations for probabilistic networks. In: Proceedings of the 44th ACM SIGPLAN Sympo-

sium on Principles of Programming Languages. p. 557–571. POPL 2017, ACM (2017). https://doi.org/10.1145/3009837.3009843
32. Yan, P., Jiang, H., Yu, N.: On incorrectness logic for quantum programs. Proc. ACM Program. Lang. **6**(OOPSLA1) (apr 2022). https://doi.org/10.1145/3527316
33. Zhang, C., de Amorim, A.A., Gaboardi, M.: On Incorrectness Logic and Kleene Algebra with Top and Tests. Proceedings of the ACM on Programming Languages **6**(POPL), 1–30 (Jan 2022). https://doi.org/10.1145/3498690