Kleene Algebra with Observations

Tobias Kappé
University College London, UK
tkappe@cs.ucl.ac.uk

Paul Brunet
University College London, UK

Jurriaan Rot
University College London, UK
Radboud University, Nijmegen, The Netherlands

Alexandra Silva
University College London, UK

Jana Wagemaker
University College London, UK

Fabio Zanasi
University College London, UK

Abstract

Kleene algebra with tests (*KAT*) is an algebraic framework for reasoning about the control flow of sequential programs. Generalising KAT to reason about concurrent programs is not straightforward, because axioms native to KAT in conjunction with expected axioms for concurrency lead to an anomalous equation. In this paper, we propose *Kleene algebra with observations* (*KAO*), a variant of KAT, as an alternative foundation for extending KAT to a concurrent setting. We characterise the free model of KAO, and establish a decision procedure w.r.t. its equational theory.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Concurrent Kleene algebra, Kleene algebra with tests, free model, axiomatisation, decision procedure

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.41

Related Version A full version of the paper is available at `https://arxiv.org/abs/1811.10401`.

Funding ERC Starting Grant ProFoundNet (679127)
Jurriaan Rot: Marie Curie Fellowship (795119)
Alexandra Silva: Leverhulme Prize (PLP–2016–129)
Fabio Zanasi: EPSRC grant (EP/R020604/1)

Acknowledgements We are grateful to Jean-Baptiste Jeannin, Dan Frumin and Damien Pous individually, for their input which helped contextualise this work.

1 Introduction

The axioms of *Kleene algebra* (*KA*) [22, 9] correspond well to program composition [14], making them a valuable tool for studying equivalences between programs from an algebraic perspective. An extension of Kleene algebra known as *Kleene algebra with tests* (*KAT*) [24] adds primitives for conditional branching, and is particularly useful when proving validity of program transformations, such as optimisations applied by a compiler [28, 41].

As a matter of fact, KAT is sufficiently abstract to express not only program behaviour, but also program specifications; consequently, its laws can be used to compare programs to specifications [25, 1]. What makes this connection especially powerful is that KA (resp. KAT)

is known to be *sound* and *complete* with respect to a language model [5, 30, 23, 29], meaning that an equation is valid in any KA (resp. KAT) precisely when it holds in the corresponding language model. Practical algorithms for deciding language equivalence [17, 6, 34] enable checking equations in KA or KAT, and hence automated verification becomes feasible [10].

More recently, Kleene algebra has been extended with a parallel composition operator, yielding *concurrent Kleene algebra* (*CKA*) [15, 13, 16]. Crucially, CKA includes the *exchange law*, which encodes *interleaving*, i.e., the (partial) sequentialisation of threads. Like its predecessors, CKA can be applied to verify (concurrent) programs by reasoning about equivalences [16]. The equational theory of CKA has also been characterised in terms of a language-like semantics [32, 21], where equivalence is known to be decidable [7].

Since both KAT and CKA are conservative extensions of KA, this prompted Jipsen and Moshier [19] to study a marriage between the two, dubbed *concurrent Kleene algebra with tests* (*CKAT*). The aim of CKAT is to extend CKA with Boolean guards, and thus arrive at a new algebraic perspective on verification of concurrent programs with conditional branching.

The starting point of this paper is the realisation that CKAT is not a suitable model of concurrent programs. This is because for any test p and CKAT-term e, one can prove $p \cdot e \cdot \overline{p} \equiv_{\mathsf{CKAT}} 0$, an equation that appears to have no reasonable interpretation for programs. The derivation goes as follows:

$$0 \leqq_{\mathsf{KAT}} p \cdot e \cdot \overline{p} \leqq_{\mathsf{CKA}} e \parallel (p \cdot \overline{p}) \equiv_{\mathsf{KAT}} e \parallel 0 \equiv_{\mathsf{CKA}} 0 \ .$$

As we shall see, this is possible because of the interplay between the exchange law and the fact that KAT identifies conjunction of tests with their sequential composition. For sequential programs, this identification is perfectly reasonable. In the context of concurrency with interleaving, however, actions from another thread may be scheduled in between two sequentially composed tests, whereas the conjunction of tests executes atomically. Indeed, an action scheduled between the tests might very well change the result of the second test.

It thus appears that, to reason algebraically about programs with both tests and concurrency, one needs a perspective on conditional branching where the conjunction of two tests is not necessarily the same as executing one test after the other. The remit of this paper is to propose an alternative to KAT, which we call *Kleene algebra with observations* (*KAO*), that makes exactly this distinction. We claim that, because of this change, KAO is more amenable to a sensible extension with primitives for concurrency. Establishing the meta-theory of KAO turns out to be a technically demanding task. We therefore devote this paper to such foundations, and leave development of concurrent KAO to follow-up work.

Concretely, we characterise the equational theory of KAO in terms of a language model (Section 5). Furthermore, we show that we can decide equality of these languages (and hence the equational theory of KAO) by deciding language equivalence of non-deterministic finite automata (Section 6). Both proofs show a clear separation of concerns: their kernel is idiomatic to KAO, and some well-known results from KA complete the argument.

For space reasons, detailed proofs are only included in the full version of this paper [20]; here, we sketch the main insights needed to prove the core propositions and theorems.

2 Preliminaries

We start by outlining some concepts and elementary results.

Boolean algebra. We use 2 to denote the set $\{0, 1\}$. The *powerset* (i.e., set of subsets) of a set S is denoted 2^S. We fix a finite set Ω of symbols called *observables*.

The *propositional terms* over Ω, denoted $\mathcal{T}_P$, are generated by the grammar

$$p, q ::= \bot \mid \top \mid o \in \Omega \mid p \vee q \mid p \wedge q \mid \overline{p} \ .$$

We write $\equiv_{\mathsf{BA}}$ for the smallest congruence on $\mathcal{T}_P$ that satisfies the axioms of Boolean algebra, i.e., such that for all $p, q, r \in \mathcal{T}_P$ the following hold:

$$p \vee \bot \equiv_{\mathsf{BA}} p \qquad p \vee q \equiv_{\mathsf{BA}} q \vee p \qquad p \vee \overline{p} \equiv_{\mathsf{BA}} \top \qquad p \vee (q \vee r) \equiv_{\mathsf{BA}} (p \vee q) \vee r$$

$$p \wedge \top \equiv_{\mathsf{BA}} p \qquad p \wedge q \equiv_{\mathsf{BA}} q \wedge p \qquad p \wedge \overline{p} \equiv_{\mathsf{BA}} \bot \qquad p \wedge (q \wedge r) \equiv_{\mathsf{BA}} (p \wedge q) \wedge r$$

$$p \vee (q \wedge r) \equiv_{\mathsf{BA}} (p \vee q) \wedge (p \vee r) \qquad p \wedge (q \vee r) \equiv_{\mathsf{BA}} (p \wedge q) \vee (p \wedge r) \ .$$

The set of *atoms*, denoted $\mathcal{A}$, is defined as 2^{Ω}. The semantics of propositional terms is given by the map $[\![-]\!]_{\mathsf{BA}} : \mathcal{T}_P \to 2^{\mathcal{A}}$, as follows:

$$[\![\bot]\!]_{\mathsf{BA}} = \emptyset \qquad [\![o]\!]_{\mathsf{BA}} = \{\alpha \in \mathcal{A} : o \in \alpha\} \qquad [\![p \vee q]\!]_{\mathsf{BA}} = [\![p]\!]_{\mathsf{BA}} \cup [\![q]\!]_{\mathsf{BA}}$$

$$[\![\top]\!]_{\mathsf{BA}} = \mathcal{A} \qquad [\![\overline{p}]\!]_{\mathsf{BA}} = \mathcal{A} \setminus [\![p]\!]_{\mathsf{BA}} \qquad [\![p \wedge q]\!]_{\mathsf{BA}} = [\![p]\!]_{\mathsf{BA}} \cap [\![q]\!]_{\mathsf{BA}} \ .$$

We also write $p \leqq_{\mathsf{BA}} q$ as a shorthand for $p \vee q \equiv_{\mathsf{BA}} q$.

It is known that $[\![-]\!]_{\mathsf{BA}}$ characterises $\equiv_{\mathsf{BA}}$ (c.f. [4, Chapter 5.9]), in the following sense:

▶ **Theorem 2.1** (Completeness for BA). *Let $p, q \in \mathcal{T}_P$; now $p \equiv_{\mathsf{BA}} q$ if and only if $[\![p]\!]_{\mathsf{BA}} = [\![q]\!]_{\mathsf{BA}}$.*

When $\alpha \in \mathcal{A}$, we write π_α for the Boolean term $\bigwedge_{o \in \alpha} o \wedge \bigwedge_{o \in \Omega \setminus \alpha} \overline{o}$, in which $\bigwedge$ is the obvious generalisation of $\wedge$ for some (arbitrary) choice of bracketing and order on terms. The following is then straightforward to prove.

▶ **Lemma 2.2.** *For all $\alpha \subseteq \Omega$ it holds that $[\![\pi_\alpha]\!]_{\mathsf{BA}} = \{\alpha\}$.*

Kleene algebra. A *word* over a set Δ is a sequence of symbols $d_0 \cdots d_{n-1}$ from Δ. The *empty word* is denoted ϵ. A set of words is called a *language*. Words can be *concatenated*: if w and x are words, then wx is the word where the symbols of w precede those of x. If L and L' are languages over Δ, then $L \cdot L'$ is the language of pairwise concatenations from L and L', i.e., $\{wx : w \in L, x \in L'\}$. We write $L^\star$ for the *Kleene closure* of L, which is the set $\{w_0 \cdots w_{n-1} : w_0, \ldots, w_{n-1} \in L\}$. This makes $\Delta^\star$ the set of all words over Δ.

We fix a finite set of symbols Σ called the *alphabet*. The *rational terms* over Σ, denoted $\mathcal{T}_R$, are generated by the grammar

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^\star \ .$$

We write $\equiv_{\mathsf{KA}}$ for the smallest congruence on $\mathcal{T}_R$ that satisfies the axioms of Kleene algebra, i.e., such that for all $e, f, g \in \mathcal{T}_R$ the following hold:

$$e + 0 \equiv_{\mathsf{KA}} e \qquad e + e \equiv_{\mathsf{KA}} e \qquad e + f \equiv_{\mathsf{KA}} f + e \qquad e + (f + g) \equiv_{\mathsf{KA}} (e + f) + g$$

$$e \cdot 1 \equiv_{\mathsf{KA}} e \qquad e \equiv 1 \cdot e \qquad e \cdot 0 \equiv_{\mathsf{KA}} 0 \qquad 0 \equiv 0 \cdot e \qquad e \cdot (f \cdot g) \equiv_{\mathsf{KA}} (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv_{\mathsf{KA}} e \cdot f + e \cdot g \qquad 1 + e \cdot e^\star \equiv_{\mathsf{KA}} e^\star \qquad e + f \cdot g \leqq_{\mathsf{KA}} g \implies f^\star \cdot e \leqq_{\mathsf{KA}} g$$

$$(e + f) \cdot g \equiv_{\mathsf{KA}} e \cdot g + f \cdot g \qquad 1 + e^\star \cdot e \equiv_{\mathsf{KA}} e^\star \qquad e + f \cdot g \leqq_{\mathsf{KA}} f \implies e \cdot g^\star \leqq_{\mathsf{KA}} f \ ,$$

in which $e \leqq_{\mathsf{KA}} f$ is a shorthand for $e + f \equiv_{\mathsf{KA}} f$.

The semantics of rational terms is given by $[\![-]\!]_{\mathsf{KA}} : \mathcal{T}_R \to 2^{\Sigma^\star}$, in the following sense:

$$[\![0]\!]_{\mathsf{KA}} = \emptyset \qquad [\![a]\!]_{\mathsf{KA}} = \{a\} \qquad [\![e + f]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}} \cup [\![f]\!]_{\mathsf{KA}}$$
$$[\![1]\!]_{\mathsf{KA}} = \{\epsilon\} \qquad [\![e^\star]\!]_{\mathsf{KA}} = [\![e]\!]^\star_{\mathsf{KA}} \qquad [\![e \cdot f]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}} \cdot [\![f]\!]_{\mathsf{KA}} \ .$$

It is furthermore known that $[\![-]\!]_{\mathsf{KA}}$ characterises $\equiv_{\mathsf{KA}}$ [5, 30, 23], as follows:

▶ **Theorem 2.3** (Completeness for KA). *Let $e, f \in \mathcal{T}_R$; now $e \equiv_{\mathsf{KA}} f$ if and only if $[\![e]\!]_{\mathsf{KA}} = [\![f]\!]_{\mathsf{KA}}$.*

We also work with matrices and vectors of rational terms. Let Q be a finite set. A *Q-vector* is a function $x : Q \to \mathcal{T}_R$; a *Q-matrix* is a function $M : Q \times Q \to \mathcal{T}_R$.

Let $e \in \mathcal{T}_R$, let x and y be Q-vectors, and let M be a Q-matrix. *Addition* of vectors is defined pointwise, i.e., $x + y$ is the Q-vector given by $(x + y)(q) = x(q) + y(q)$. We can also *scale* vectors, writing $x \fatsemi e$ for the Q-vector given by $(x \fatsemi e)(q) = x(q) \cdot e$.

Multiplication of a Q-vector by a Q-matrix yields a Q-vector, as expected:

$$(M \cdot x)(q) = \sum_{q' \in Q} M(q, q') \cdot x(q') \ .$$

Here, $\sum$ is the usual generalisation of $+$ for some (arbitrary) choice of bracketing and order on terms; the empty sum is defined to be 0.

We write $x \equiv_{\mathsf{KA}} y$ when x and y are pointwise equivalent, i.e., for all $q \in Q$ we have $x(q) \equiv_{\mathsf{KA}} y(q)$; we extend $\leqq_{\mathsf{KA}}$ to Q-vectors as before. Matrices over rational terms again obey the axioms of Kleene algebra [23]. As a special case, we can obtain the following:

▶ **Lemma 2.4.** *Let M be a Q-matrix. Using the entries of M and applying the operators of Kleene algebra, we can construct a Q-matrix $M^\star$, which has the following property. Let y be any Q-vector; now $M^\star \cdot y$ is the least (w.r.t. $\leqq_{\mathsf{KA}}$) Q-vector x such that $M \cdot x + y \leqq_{\mathsf{KA}} x$.*

Automata and bisimulations. We briefly recall *bisimulation up to congruence* for language equivalence of automata, from [6]. This will be used in Section 6.

A *non-deterministic automaton* (*NDA*) over an alphabet Σ is a triple (X, o, d) where $o\colon X \to 2$ is an output function, and $d\colon X \times \Sigma \to X$ a transition function. A non-deterministic finite automaton (NFA) is an NDA where X is finite. It will be convenient to characterise the semantics of an NDA (X, o, d) recursively as the unique map $\ell : X \to 2^{\Sigma^\star}$ such that

$$\ell(x) = \{\epsilon : o(x) = 1\} \cup \bigcup_{x' \in d(x,a)} \{a\} \cdot \ell(x') \ .$$

This coincides with the standard definition of language acceptance for NDAs. The *determinisation* of an NDA (X, o, d) is the deterministic automaton $(2^X, \overline{o}, \overline{d})$ [35], where

$$\overline{o}(V) = \begin{cases} 1 & \text{if } \exists s \in V \text{ s.t. } o(s) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \overline{d}(V, a) = \bigcup_{s \in V} d(s, a) \ .$$

The *congruence closure* of a relation $R \subseteq 2^X \times 2^X$, denoted R^c, is the least equivalence relation such that $R \subseteq R^c$, and if $(C, D) \in R^c$ and $(E, F) \in R^c$ then $(C \cup E, D \cup F) \in R^c$. A *bisimulation up to congruence* for an NDA (X, o, d) is a relation $R \subseteq 2^X \times 2^X$ such that for all $(V, W) \in R$, we have $\bar{o}(V) = \bar{o}(W)$ and furthermore for all $a \in \Sigma$, we have $(\bar{d}(V, a), \bar{d}(W, a)) \in R^c$. Bisimulations up to congruence give a proof technique for language equivalence of states of non-deterministic automata, as a consequence of the following [6].

▶ **Theorem 2.5.** *Let (X, o, d) be an NDA. For all $U, V \in 2^X$, we have $\bigcup_{x \in U} \ell(x) = \bigcup_{x \in V} \ell(x)$ if and only if there exists a bisimulation up to congruence R for (X, o, d) such that $(U, V) \in R$.*

In [6], it is shown how the construction of bisimulations up to congruence leads to a very efficient algorithm for language equivalence.

3 The problem with CKAT

Our motivation for adjusting the axioms of KAT is based on the fact that conjunction and sequential composition are distinct when interleaving is involved, because sequential composition leaves a "gap" between tests that might be used by another thread, possibly changing the outcome of the second test. We now formalise this, by detailing how combining KAT and CKA to obtain CKAT (as in [19]) leads to the absurd equation $p \cdot e \cdot \overline{p} \equiv_{\mathsf{CKAT}} 0$.

Kleene algebra with tests. To obtain KAT, we enrich rational terms with propositions; concretely, the set of *guarded rational terms* over Σ and Ω, denoted $\mathcal{T}_{GR}$, is generated by

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid p \in \mathcal{T}_P \mid e + f \mid e \cdot f \mid e^\star \ .$$

We define $\equiv_{\mathsf{KAT}}$ as the smallest congruence on $\mathcal{T}_{GR}$ which contains $\equiv_{\mathsf{BA}}$ and obeys the axioms of $\equiv_{\mathsf{KA}}$ (e.g., $e + e \equiv_{\mathsf{KAT}} e$). Furthermore, $\equiv_{\mathsf{KAT}}$ should relate constants and operators on propositional subterms: for all $p, q \in \mathcal{T}_P$ it holds that[1]

$$\bot \equiv_{\mathsf{KAT}} 0 \qquad \top \equiv_{\mathsf{KAT}} 1 \qquad p \vee q \equiv_{\mathsf{KAT}} p + q \qquad p \wedge q \equiv_{\mathsf{KAT}} p \cdot q \ .$$

Guarded rational terms relate to programs by viewing actions as statements and propositions as assertions. The last axiom is therefore not strange: if we assert $\mathtt{x} = \mathtt{1}$ followed by $\mathtt{y} = \mathtt{1}$ then surely, if $\mathtt{x}$ and $\mathtt{y}$ remain the same, this is equivalent to asserting $\mathtt{x} = \mathtt{1} \wedge \mathtt{y} = \mathtt{1}$.

Concurrent Kleene algebra. To obtain CKA, we add a parallel composition operator to KA. Concretely, the set of *series-rational terms* [33] over Σ, denoted $\mathcal{T}_{SR}$, is generated by

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e \parallel f \mid e^\star \ .$$

We define $\equiv_{\mathsf{CKA}}$ as the smallest congruence on $\mathcal{T}_{SR}$ which obeys the axioms that generate $\equiv_{\mathsf{KA}}$, as well as the following for all $e, f, g, h \in \mathcal{T}_{SR}$:

$$e \parallel f \equiv_{\mathsf{CKA}} f \parallel e \qquad e \parallel 1 \equiv_{\mathsf{CKA}} e \qquad e \parallel 0 \equiv_{\mathsf{CKA}} 0 \qquad (e + f) \parallel g \equiv_{\mathsf{CKA}} e \parallel g + f \parallel g$$

$$e \parallel (f \parallel g) \equiv_{\mathsf{CKA}} (e \parallel f) \parallel g \qquad (e \parallel f) \cdot (g \parallel h) \leqq_{\mathsf{CKA}} (e \cdot g) \parallel (f \cdot h) \ ,$$

in which $e \leqq_{\mathsf{CKA}} f$ is an abbreviation for $e + f \equiv_{\mathsf{CKA}} f$.

Here, 0 annihilates parallel composition because 0 corresponds to the program without valid traces; hence, running e in parallel with 0 also cannot yield any traces. Distributivity of $\parallel$ over $+$ witnesses that a choice within a thread can also be made outside that thread.

The last axiom, called the *exchange law* [15], encodes interleaving. Intuitively, it says that when programs $e \cdot g$ and $f \cdot h$ run in parallel, their behaviour includes running their heads in parallel (i.e., $e \parallel f$) followed by their tails (i.e., $g \parallel h$). Taken to its extreme, the exchange law says that the behaviour of a program includes its complete linearisations.

[1] This is a slightly contrived definition of KAT; one usually presents the constants and operators using the same symbols [24]. To contrast KAO and KAT it is helpful to make this identification explicit.

Concurrent Kleene algebra with tests. To define CKAT [19], we choose *guarded series-rational terms* over Σ and Ω, denoted $\mathcal{T}_{GSR}$, as those generated by the grammar

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid p \in \mathcal{T}_P \mid e + f \mid e \cdot f \mid e \parallel f \mid e^\star \ .$$

Now, $\equiv_{\mathsf{CKAT}}$ is the least congruence on $\mathcal{T}_{GSR}$ obeying the axioms of $\equiv_{\mathsf{KAT}}$ and $\equiv_{\mathsf{CKA}}$.

If we view guarded series-rational terms as representations of programs, and use $\equiv_{\mathsf{CKAT}}$ to reason about them, then we should expect to obtain sensible statements about programs, since the axioms that underpin CKAT seem reasonable in that context.

To test this hypothesis, consider the guarded series-rational term $p \cdot e \cdot \overline{p}$, which represents a program that first performs an assertion p, then runs a program described by e, and asserts the negation of p. There are choices of p and e that should make $p \cdot e \cdot \overline{p}$ describe a program with valid behaviour; for instance, p could assert that $\mathtt{x} = \mathtt{1}$, and e could be the program $\mathtt{x} \leftarrow \mathtt{0}$. Hence, by our hypothesis, $p \cdot e \cdot \overline{p}$ should not, in general, be equivalent to 0, the program without any valid behaviour. Unfortunately, the opposite is true.

▶ **Antinomy 3.1.** *Let $e \in \mathcal{T}_{GSR}$ and $p \in \mathcal{T}_P$; now $p \cdot e \cdot \overline{p} \equiv_{\mathsf{CKAT}} 0$.*

Proof. By the axiom that 1 is the unit of $\parallel$ and the exchange law, we derive that

$$p \cdot e \cdot \overline{p} \equiv_{\mathsf{CKAT}} (p \parallel 1) \cdot (1 \parallel e) \cdot \overline{p} \leqq_{\mathsf{CKAT}} ((p \cdot 1) \parallel (1 \cdot e)) \cdot \overline{p} \ .$$

By the same reasoning, and the axiom that 1 is the unit of $\cdot$, we have that

$$((p \cdot 1) \parallel (1 \cdot e)) \cdot \overline{p} \equiv_{\mathsf{CKAT}} (p \parallel e) \cdot (\overline{p} \parallel 1) \leqq_{\mathsf{CKAT}} (p \cdot \overline{p}) \parallel (e \cdot 1) \ .$$

Applying the unit, and using the identification of $\wedge$ and $\cdot$ on tests, we find that

$$(p \cdot \overline{p}) \parallel (e \cdot 1) \equiv_{\mathsf{CKAT}} (p \cdot \overline{p}) \parallel e \equiv_{\mathsf{CKAT}} (p \wedge \overline{p}) \parallel e \equiv_{\mathsf{CKAT}} \bot \parallel e \ .$$

Since $\bot$ and 0 are identified, and 0 annihilates parallel composition, we have that

$$\bot \parallel e \equiv_{\mathsf{CKAT}} 0 \parallel e \equiv_{\mathsf{CKAT}} 0 \ .$$

By the above, we have $p \cdot e \cdot \overline{p} \leqq_{\mathsf{CKAT}} 0$. Since $0 \leqq_{\mathsf{CKAT}} p \cdot e \cdot \overline{p}$, the claim follows. ◀

Another way of contextualising the above is to consider that propositional Hoare logic can be encoded in KAT [26]. Concretely, a propositional Hoare triple $\{p\}S\{q\}$ is valid if $p \cdot e_S \cdot \overline{q} \equiv_{\mathsf{KAT}} 0$, where e_S is a straightforward encoding of S. It stands to reason that sequential programs should similarly encode in CKAT. However, if we apply that line of reasoning to the above, then *any* Hoare triple of the form $\{p\}S\{p\}$ is valid, which would mean that *any property is an invariant of any program.*

4 Kleene algebra with observations

We now propose KAO as an alternative way of embedding Boolean guards in rational terms. This approach should prevent Antinomy 3.1, and thus make KAO more suitable for an extension with concurrency. We start by motivating how we adapt KAT, before proceeding with a language model and a generalisation of partial derivatives to KAO.

4.1 From tests to observations

The root of the problem in Antinomy 3.1 is the axiom $p \cdot q \equiv_{\mathsf{KAT}} p \wedge q$, telling us that $(p \cdot \overline{p}) \parallel e \equiv_{\mathsf{cKAT}} (p \wedge \overline{p}) \parallel e$. Interpreted as programs, these are different: in one, e can be interleaved between p and $\overline{p}$, which the other does not allow.

To fix this problem, we propose a new perspective on Boolean guards. Rather than considering a guard a *test*, which in KAT entails an assertion valid from the last action to the next, we consider a guard an *observation*, i.e., an assertion valid at that particular point in the execution of the program. This weakens the connection between conjunction and concatenation: if a program observes p followed by q, there is no guarantee that p and q can be observed simultaneously. Hence, we drop the equivalence between conjunction and concatenation of tests. We call this system *weak Kleene algebra with observations*; like KAT, its axioms are based on the axioms of Boolean algebra and Kleene algebra, as follows.

▶ **Definition 4.1** (WKAO axioms). *We define $\equiv_{\mathsf{WKAO}}$ as the smallest congruence on $\mathcal{T}_{GR}$ that contains $\equiv_{\mathsf{BA}}$ and also obeys the axioms of $\equiv_{\mathsf{KA}}$. Furthermore, $\bot \equiv_{\mathsf{WKAO}} 0$ and for all $p, q \in \mathcal{T}_P$ it holds that $p \vee q \equiv_{\mathsf{WKAO}} p + q$.*

Since conjunction and concatenation no longer coincide, we have also dropped the axiom that relates their units. This can be justified from our shift in perspective: $\top$, i.e., the observation that always succeeds, is an action that leaves some record in the behaviour of the program, whereas 1, i.e., the program that does nothing, has no such obligation.[2]

As hinted before, it may happen that when a program observes p followed by q, both these assertions are true simultaneously, i.e., their conjunction could have been observed; hence, the behaviour of observing p *and* q should be contained in the behaviour of observing p *and then* q. For instance, consider the pseudo-programs $S = \texttt{assert}\ p \wedge q;\ e$ and $S' = \texttt{assert}\ p;\ \texttt{assert}\ q;\ e$. Here, S asserts that p and q hold at the same time before proceeding with e, while S' first asserts p, and then q. The behaviour of S should be included in that of S': if p and q are simultaneously observable, then the first two observations might take place simultaneously. Encoding this in an additional axiom leads to KAO proper.

▶ **Definition 4.2** (KAO axioms). *We define $\equiv_{\mathsf{KAO}}$ as the smallest congruence on $\mathcal{T}_{GR}$ that contains $\equiv_{\mathsf{WKAO}}$, and furthermore satisfies the* contraction law*: for all $p, q \in \mathcal{T}_P$ it holds that $p \wedge q \leqq_{\mathsf{KAO}} p \cdot q$ – where $e \leqq_{\mathsf{KAO}} f$ is a shorthand for $e + f \equiv_{\mathsf{KAO}} f$.*

We briefly return to our example programs. If we encode S as $(p \wedge q) \cdot e$ and S' as $p \cdot q \cdot e$, then the behaviour of S is contained in that of S', because $(p \wedge q) \cdot e \leqq_{\mathsf{KAO}} p \cdot q \cdot e$.

▶ Remark 4.3. In the presence of the other axioms, $p \wedge q \leqq_{\mathsf{KAO}} p \cdot q$ is equivalent to $p \leqq_{\mathsf{KAO}} p \cdot p$. Indeed, the second inequality may be inferred from the first, since $p \equiv_{\mathsf{KAO}} p \wedge p$. The converse implication is obtained as follows: $p \wedge q \leqq_{\mathsf{KAO}} (p \wedge q) \cdot (p \wedge q) \leqq_{\mathsf{KAO}} p \cdot q$, using that $p \wedge q \leqq_{\mathsf{KAO}} p, q$. While the contraction law is more convenient to compare terms, the axiom $p \leqq_{\mathsf{KAO}} p \cdot p$ will serve implicitly as the basis for the contraction relation $\preceq$ defined in the next section.

4.2 A language model

The Kleene algebra variants encountered thus far have models based on rational languages, which characterise the equivalences derivable using the axioms. To find a model for guarded rational terms which corresponds to KAO, we start with a model of weak KAO:

[2] We refer to [20, Appendix E] for further algebraic details on the consequences of identifying these units.

▶ **Definition 4.4** (WKAO semantics). *Let* $\Gamma = \mathcal{A} \cup \Sigma$. *Languages over* Γ *are called* observation languages. *We define* $[\![-]\!]_{\mathsf{WKAO}} : \mathcal{T}_{GR} \to 2^{\Gamma^\star}$ *as follows:*

$$[\![0]\!]_{\mathsf{WKAO}} = \emptyset \qquad [\![a]\!]_{\mathsf{WKAO}} = \{a\} \qquad [\![e + f]\!]_{\mathsf{WKAO}} = [\![e]\!]_{\mathsf{WKAO}} \cup [\![f]\!]_{\mathsf{WKAO}} \qquad [\![e^\star]\!]_{\mathsf{WKAO}} = [\![e]\!]^\star_{\mathsf{WKAO}}$$
$$[\![1]\!]_{\mathsf{WKAO}} = \{\epsilon\} \qquad [\![p]\!]_{\mathsf{WKAO}} = [\![p]\!]_{\mathsf{BA}} \qquad [\![e \cdot f]\!]_{\mathsf{WKAO}} = [\![e]\!]_{\mathsf{WKAO}} \cdot [\![f]\!]_{\mathsf{WKAO}} \ ,$$

where it is understood that $[\![p]\!]_{\mathsf{BA}} \subseteq \mathcal{A} \subseteq \Gamma \subseteq \Gamma^\star$.

Observation languages are a model for guarded rational terms w.r.t. $\equiv_{\mathsf{WKAO}}$:

▶ **Lemma 4.5** (WKAO soundness). *Let* $e, f \in \mathcal{T}_{GR}$. *Now, if* $e \equiv_{\mathsf{WKAO}} f$, *then* $[\![e]\!]_{\mathsf{WKAO}} = [\![f]\!]_{\mathsf{WKAO}}$.

More work is necessary to get a model of guarded rational terms w.r.t. $\equiv_{\mathsf{KAO}}$. In particular, $[\![-]\!]_{\mathsf{WKAO}}$ does not preserve the contraction law: if $o_1, o_2 \in \Omega$, then

$$[\![o_1 \wedge o_2]\!]_{\mathsf{WKAO}} = \{\alpha \in \mathcal{A} : o_1, o_2 \in \alpha\} \qquad [\![o_1 \cdot o_2]\!]_{\mathsf{WKAO}} = \{\alpha\beta \in \mathcal{A} \cdot \mathcal{A} : o_1 \in \alpha, o_2 \in \beta\} \ ,$$

meaning that $[\![o_1 \wedge o_2]\!]_{\mathsf{WKAO}} \not\subseteq [\![o_1 \cdot o_2]\!]_{\mathsf{WKAO}}$, despite $o_1 \wedge o_2 \leqq_{\mathsf{KAO}} o_1 \cdot o_2$.[3]

To obtain a sound model, we need the counterpart of the contraction law on the level of observation languages, which works out as follows

▶ **Definition 4.6** (Contraction). *We define* $\preceq$ *as the smallest relation on* $\Gamma^\star$ *s.t.*

$$\frac{w, x \in \Gamma^\star \qquad \alpha \in \mathcal{A}}{w\alpha x \preceq w\alpha\alpha x} \ .$$

When $L \subseteq \Gamma^\star$, *we write* $L{\downarrow}$ *for the* $\preceq$*-closure of* L, *that is to say, the smallest observation language such that* $L \subseteq L{\downarrow}$, *and if* $w \preceq x$ *with* $x \in L{\downarrow}$, *then* $w \in L{\downarrow}$.

Thus, $L{\downarrow}$ contains all words obtained from words in L by contracting any number of repeated atoms (but not actions) into one. Applying this closure to the semantics encodes the intuition that repeated observations may also correspond to just one step in the execution.

With these tools, we can define a model of KAO as follows.

▶ **Definition 4.7** (KAO semantics). *We define* $[\![-]\!]_{\mathsf{KAO}} : \mathcal{T}_{GR} \to 2^{\Gamma^\star}$ *by* $[\![e]\!]_{\mathsf{KAO}} = [\![e]\!]_{\mathsf{WKAO}}{\downarrow}$.

Alternatively, we can describe $[\![-]\!]_{\mathsf{KAO}}$ compositionally, using the following.

▶ **Lemma 4.8.** *Let* $e, f \in \mathcal{T}_{GR}$. *The following hold: (i)* $[\![e + f]\!]_{\mathsf{KAO}} = [\![e]\!]_{\mathsf{KAO}} \cup [\![f]\!]_{\mathsf{KAO}}$, *and (ii)* $[\![e \cdot f]\!]_{\mathsf{KAO}} = ([\![e]\!]_{\mathsf{KAO}} \cdot [\![f]\!]_{\mathsf{KAO}}){\downarrow}$, *and (iii)* $[\![e^\star]\!]_{\mathsf{KAO}} = [\![e]\!]^\star_{\mathsf{KAO}}{\downarrow}$.

It is now straightforward to check that $[\![-]\!]_{\mathsf{KAO}}$ preserves the axioms of $\equiv_{\mathsf{KAO}}$.

▶ **Lemma 4.9** (KAO soundness). *Let* $e, f \in \mathcal{T}_{GR}$. *Now, if* $e \equiv_{\mathsf{KAO}} f$, *then* $[\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}}$.

4.3 Partial derivatives

Derivatives [8] are a powerful tool in Kleene algebra variants. In the context of programs, we can think of derivatives as an operational semantics; they tell us whether the term represents a program that can halt immediately (given by the *termination map*), as well as the program that remains to be executed once an action is performed (given by the *continuation map*).

[3] Incidentally, this shows that the contraction law is independent of the axioms that build $\equiv_{\mathsf{WKAO}}$, i.e., that those axioms are not sufficient to prove the contraction law.

Derivatives are typically compatible with the congruences used to reason about terms; what's more, they are closely connected to finite automata [8]. This makes them useful for reasoning about language models [26, 6, 34]. We therefore introduce a form of derivatives of guarded rational terms that works well with $\equiv_{\mathsf{KAO}}$. Let us start by giving the termination map, which is close to the termination map of rational terms compatible with $\equiv_{\mathsf{KA}}$.

▶ **Definition 4.10** (Termination). *We define $\epsilon : \mathcal{T}_{GR} \to 2$ inductively, as follows:*

$$\epsilon(0) = 0 \qquad \epsilon(a) = 0 \qquad \epsilon(e+f) = \max\{\epsilon(e), \epsilon(f)\} \qquad \epsilon(e^\star) = 1$$
$$\epsilon(1) = 1 \qquad \epsilon(p) = 0 \qquad \epsilon(e \cdot f) = \min\{\epsilon(e), \epsilon(f)\} \ .$$

To check that ϵ characterises guarded rational terms that can terminate, we record the following lemma, which says that $\epsilon(e) = 1$ precisely when $[\![e]\!]_{\mathsf{KAO}}$ includes the empty string.

▶ **Lemma 4.11.** *Let $e \in \mathcal{T}_{GR}$. Now $\epsilon(e) \leqq_{\mathsf{KAO}} e$, and $\epsilon(e) = 1$ if and only if $\epsilon \in [\![e]\!]_{\mathsf{KAO}}$.*

Next we define the continuation map, or more specifically, *partial continuation map* [2]: given $\mathfrak{a} \in \Gamma$, this function gives a *set* of terms representing possible continuations of the program after performing $\mathfrak{a}$. We start with the continuation map for actions.

▶ **Definition 4.12** (Σ-continuation). *We define $\delta : \mathcal{T}_{GR} \times \Sigma \to 2^{\mathcal{T}_{GR}}$ inductively*

$$\delta(0,a) = \delta(1,a) = \emptyset \qquad \delta(e+f,a) = \delta(e,a) \cup \delta(f,a)$$
$$\delta(a,a') = \{1 : a = a'\} \qquad \delta(e \cdot f, a) = \{e' \cdot f : e' \in \delta(e,a)\} \cup \Delta(e,f,a)$$
$$\delta(p,a) = \emptyset \qquad \delta(e^\star, a) = \{e' \cdot e^\star : e' \in \delta(e,a)\} \ ,$$

where $\Delta(e,f,a) = \delta(f,a)$ when $\epsilon(e) = 1$, and $\emptyset$ otherwise.

Finally, we give the continuation map for observations, which is similar to the one for actions, except that on sequential composition it is subtly different.

▶ **Definition 4.13** ($\mathcal{A}$-continuation). *We define $\zeta : \mathcal{T}_{GR} \times \mathcal{A} \to 2^{\mathcal{T}_{GR}}$ inductively*

$$\zeta(0,\alpha) = \delta(1,\alpha) = \emptyset \qquad \zeta(e+f,\alpha) = \zeta(e,\alpha) \cup \zeta(f,\alpha)$$
$$\zeta(a,\alpha) = \emptyset \qquad \zeta(e \cdot f, \alpha) = \{e' \cdot f : e' \in \zeta(e,\alpha)\} \cup Z(e,f,\alpha)$$
$$\zeta(p,\alpha) = \{1 : \pi_\alpha \leqq_{\mathsf{BA}} p\} \qquad \zeta(e^\star, \alpha) = \{e' \cdot e^\star : e' \in \zeta(e,\alpha)\} \ ,$$

where $Z(e,f,\alpha) = \zeta(f,\alpha)$ when $\epsilon(e) = 1$ or $\epsilon(e') = 1$ for some $e' \in \zeta(e,\alpha)$, and $\emptyset$ otherwise.

For instance, let $p, q \in \mathcal{T}_P$. If $\alpha \in [\![p]\!]_{\mathsf{BA}} \cap [\![q]\!]_{\mathsf{BA}}$, then we can calculate that

$$\zeta(p \cdot q, \alpha) = \{p' \cdot q : p' \in \zeta(p,\alpha)\} \cup Z(p,q,\alpha) = \{1 \cdot q\} \cup \{1\} \ ,$$

which is to say that the program can either continue in $1 \cdot q$, where it has to make an observation validating q, or it can choose to re-use the observation α to validate q, continuing in $Z(p,q,\alpha) = \zeta(q,\alpha) = \{1\}$, because $1 \in \zeta(p,\alpha)$. This notion that ζ can apply an observation more than once to validate multiple assertions in a row can be formalised.

▶ **Lemma 4.14.** *Let $e \in \mathcal{T}_{GR}$, $\alpha \in \mathcal{A}$, and $e' \in \zeta(e,\alpha)$. Now $\zeta(e',\alpha) \subseteq \zeta(e,\alpha)$.*

To check that δ and ζ indeed output continuations of the term after the action or assertion has been performed, we should check that they are compatible with $\equiv_{\mathsf{KAO}}$.

▶ **Lemma 4.15.** *Let $e \in \mathcal{T}_{GR}$. For all $a \in \Sigma$ and $e' \in \delta(e,a)$, we have $a \cdot e' \leqq_{\mathsf{KAO}} e$. Furthermore, for all $\alpha \in \mathcal{A}$ and $e' \in \zeta(e,\alpha)$, we have $\pi_\alpha \cdot e' \leqq_{\mathsf{KAO}} e$.*

We also need the *reach* of a guarded rational term, which is meant to describe the set of terms that can be obtained by repeatedly applying continuation maps.

▶ **Definition 4.16** (Reach of a term). *For $e \in \mathcal{T}_{GR}$, we define $\rho : \mathcal{T}_{GR} \to 2^{\mathcal{T}_{GR}}$ inductively:*

$$\begin{aligned} \rho(0) &= \emptyset & \rho(a) &= \{1, a\} & \rho(e+f) &= \rho(e) \cup \rho(f) \\ \rho(1) &= \{1\} & \rho(p) &= \{1, p\} & \rho(e \cdot f) &= \{e' \cdot f : e' \in \rho(e)\} \cup \rho(f) \\ & & & & \rho(e^\star) &= \{1\} \cup \{e' \cdot e^\star : e' \in \rho(e)\} \ . \end{aligned}$$

Indeed, $\rho(e)$ truly contains all terms reachable from e by means of δ and ζ:

▶ **Lemma 4.17.** *Let $e \in \mathcal{T}_{GR}$. If $a \in \Sigma$, then $\delta(e, a) \subseteq \rho(e)$; if $e' \in \rho(e)$, then $\delta(e', a) \subseteq \rho(e)$. Furthermore, if $\alpha \in \mathcal{A}$, then $\zeta(e, \alpha) \subseteq \rho(e)$; if $e' \in \rho(e)$, then $\zeta(e', \alpha) \subseteq \rho(e)$.*

It is not hard to see that for all $e \in \mathcal{T}_{GR}$, we have that $\rho(e)$ is finite. Note that $\rho(e)$ does not, in general, contain e itself. On the other hand, $\rho(e)$ does contain a set of terms that is sufficient to reconstruct e, up to $\equiv_{\mathsf{KAO}}$. We describe these as follows.

▶ **Definition 4.18** (Initial factors). *For $e \in \mathcal{T}_{GR}$, we define $\iota : \mathcal{T}_{GR} \to 2^{\mathcal{T}_{GR}}$ inductively:*

$$\begin{aligned} \iota(0) &= \emptyset & \iota(a) &= \{a\} & \iota(e+f) &= \iota(e) \cup \iota(f) \\ \iota(1) &= \{1\} & \iota(p) &= \{p\} & \iota(e \cdot f) &= \{e' \cdot f : e' \in \iota(e)\} \\ & & & & \iota(e^\star) &= \{1\} \cup \{e' \cdot e^\star : e' \in \iota(e)\} \ . \end{aligned}$$

Now, to reconstruct e from $\iota(e)$, all we have to do is sum its elements.

▶ **Lemma 4.19.** *Let $e \in \mathcal{T}_{GR}$. Then $e \equiv_{\mathsf{KAO}} \sum_{e' \in \iota(e)} e'$.*

5 Completeness

We now set out to show that $[\![-]\!]_{\mathsf{KAO}}$ characterises $\equiv_{\mathsf{KAO}}$. Since soundness of $[\![-]\!]_{\mathsf{KAO}}$ w.r.t. $\equiv_{\mathsf{KAO}}$ was already shown in Lemma 4.9, it remains to prove completeness, i.e., if e and f are interpreted equally by $[\![-]\!]_{\mathsf{KAO}}$, then they can be proven equivalent via $\equiv_{\mathsf{KAO}}$. To this end, we first identify a subset of guarded rational terms for which a completeness result can be shown by relying on the completeness result for KA. To describe these terms, we need the following.

▶ **Definition 5.1** (Atomic and guarded terms). *Let Π denote the set $\{\pi_\alpha : \alpha \in \mathcal{A}\}$. The set of* atomic guarded rational terms*, denoted $\mathcal{T}_{AGR}$, is the subset of $\mathcal{T}_{GR}$ generated by the grammar*

$$e, f ::= 0 \mid 1 \mid a \in \Sigma \mid \pi_\alpha \in \Pi \mid e + f \mid e \cdot f \mid e^\star \ .$$

Furthermore, if $e \in \mathcal{T}_{GR}$ such that $[\![e]\!]_{\mathsf{KAO}} = [\![e]\!]_{\mathsf{WKAO}}$, we say that e is closed*.*

Completeness of $\equiv_{\mathsf{KAO}}$ with respect to $[\![-]\!]_{\mathsf{KAO}}$ can then be established for atomic and closed guarded rational terms as follows.

▶ **Proposition 5.2.** *Let $e, f \in \mathcal{T}_{AGR}$ be closed, and $[\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}}$; then $e \equiv_{\mathsf{KAO}} f$.*

Sketch. By noting that the semantics of an atomic and closed term coincide with the KA-semantics (over an extended alphabet); the result then follows by appealing to completeness of KA, and the fact that the axioms of KA are contained in those for KAO. ◀

To show that $[\![-]\!]_{\mathsf{KAO}}$ characterises $\equiv_{\mathsf{KAO}}$ for *all* guarded rational terms, it suffices to find for every $e \in \mathcal{T}_{GR}$ a closed $\hat{e} \in \mathcal{T}_{AGR}$ with $e \equiv_{\mathsf{KAO}} \hat{e}$. After all, if we have such a transformation, then $[\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}}$ implies $[\![\hat{e}]\!]_{\mathsf{KAO}} = [\![\hat{f}]\!]_{\mathsf{KAO}}$, which implies $\hat{e} \equiv_{\mathsf{KAO}} \hat{f}$, and hence $e \equiv_{\mathsf{KAO}} f$.

In the remainder of this section, we describe how to obtain a closed and atomic guarded rational term from a guarded rational term e. This transformation works by first "disassembling" e using partial derivatives; on an intuitive level, this is akin to creating a (non-deterministic finite) automaton that accepts the observation language described by e. Next, we "reassemble" from this representation an atomic term $\hat{e}$, equivalent to e; this is analogous to constructing a rational expression from the automaton. To show that $\hat{e}$ is closed, we leverage Lemma 4.14, essentially arguing that the automaton obtained from e encodes $\preceq$.

We extend the notation for vectors and matrices over rational terms to guarded rational terms. We can then represent a guarded rational term by a matrix and a vector, as follows.

▶ **Definition 5.3.** *For $e \in \mathcal{T}_{GR}$, define the $\rho(e)$-vector x_e and $\rho(e)$-matrix M_e by*

$$x_e(e') = \epsilon(e') \qquad\qquad M_e(e', e'') = \sum_{e'' \in \delta(e', a)} a + \sum_{e'' \in \zeta(e', \alpha)} \pi_\alpha \ .$$

Note that, in the above, $\delta(e', a)$ and $\zeta(e', \alpha)$ are finite by Lemma 4.17.

Kozen's argument that matrices over rational terms satisfy the axioms of Kleene algebra [23] generalises to guarded rational terms. This leads to a straightforward generalisation of Lemma 2.4. For the sake of brevity, we use T to stand in for WKAO or KAO.

▶ **Lemma 5.4.** *Let M be a Q-matrix. Using the entries of M and applying the operators of Kleene algebra, we can construct a Q-matrix $M^\star$, which has the following property. Let y be any Q-vector; now $M^\star \cdot y$ is the least (w.r.t. $\leqq_{\mathsf{T}}$) Q-vector x such that $M \cdot x + y \leqq_{\mathsf{T}} x$.*

We can now use the above to reassemble a term from M_e and x_e as follows.

▶ **Definition 5.5** (Transformation of terms)**.** *Let $e \in \mathcal{T}_{GR}$. We write s_e for the $\rho(e)$-vector given by $M_e^\star \cdot x_e$, and $\hat{e}$ for the guarded rational term given by $\sum_{e' \in \iota(e)} s_e(e')$.*

Since $\hat{e}$ is constructed from M_e and s_e, and these are built using atomic guarded rational terms, $\hat{e}$ is atomic. We carry on to show that $\hat{e} \equiv_{\mathsf{KAO}} e$. To this end, the following is useful.

▶ **Proposition 5.6** (Least solutions)**.** *Let $e, f \in \mathcal{T}_{GR}$. Now $s_e \fatsemi f$ is the least (w.r.t. $\leqq_{\mathsf{T}}$) $\rho(e)$-vector s such that for each $e' \in \rho(e)$, it holds that*

$$\epsilon(e') \cdot f + \sum_{e'' \in \delta(e', a)} a \cdot s(e'') + \sum_{e'' \in \zeta(e', \alpha)} \pi_\alpha \cdot s(e'') \leqq_{\mathsf{T}} s(e') \ .$$

When $s = s_e \fatsemi f$, the above is an equivalence, i.e., we can substitute $\leqq_{\mathsf{T}}$ with $\equiv_{\mathsf{T}}$.

Sketch. The least such s coincides with $s_e = M_e^\star \cdot x_e$, by using the property of s_e that we obtain as a result of Lemma 5.4. The latter claim goes by standard argument akin to the argument showing that the least pre-fixpoint of a monotone operator is a fixpoint. ◀

Using the above, we show that the contents of $\rho(e)$ themselves qualify as a least $\rho(e)$-vector satisfying system obtained from e (and fixing $f = 1$). More concretely, we have the following.

▶ **Proposition 5.7.** *Let $e \in \mathcal{T}_{GR}$ and $e' \in \rho(e)$. Then $s_e(e') \equiv_{\mathsf{KAO}} e'$.*

Sketch. The key idea is to first relate the least $\rho(e)$-vectors satisfying the system obtained from e to those satisfying the systems arising from its subterms. The proof then follows by induction on e, and some straightforward derivations. ◀

This allows us to conclude (using Lemma 4.19) that $e \equiv_{\mathsf{KAO}} \hat{e}$.

▶ **Corollary 5.8** (Transformation preserves equivalence). *Let $e \in \mathcal{T}_{GR}$; then $e \equiv_{\mathsf{KAO}} \hat{e}$.*

It remains to show that $\hat{e}$ is closed. To this end, the following characterisation is helpful:

▶ **Lemma 5.9.** *Let $L \subseteq \Gamma^\star$, and define $\trianglelefteq$ as the smallest relation on $\Gamma^\star$ satisfying the rules*

$$\frac{}{\epsilon \trianglelefteq \epsilon} \qquad \frac{\mathfrak{a} \in \Gamma \qquad w \trianglelefteq x}{\mathfrak{a}w \trianglelefteq \mathfrak{a}x} \qquad \frac{\alpha \in \mathcal{A} \qquad w \trianglelefteq x}{\alpha w \trianglelefteq \alpha\alpha x} \ .$$

$L{\downarrow}$ is the smallest subset of $\Gamma^\star$ such that $L \subseteq L{\downarrow}$, and if $w \trianglelefteq x$ with $x \in L{\downarrow}$, then $w \in L{\downarrow}$.

We can then employ the characterisation of s_e in Proposition 5.6 to show, by induction on the length of the $\trianglelefteq$-chain, that the components of s_e are closed.

▶ **Proposition 5.10.** *Let $e \in \mathcal{T}$. For all $e' \in \rho(e)$, it holds that $s_e(e')$ is closed.*

Sketch. Note that s_e satisfies the system obtained from e as in Proposition 5.6, both w.r.t. $\equiv_{\mathsf{KAO}}$ and $\equiv_{\mathsf{WKAO}}$. Using Lemma 5.9, we can then show (by induction on the number of applications of $\trianglelefteq$) that $[\![s(e')]\!]_{\mathsf{WKAO}} \subseteq [\![s(e')]\!]_{\mathsf{KAO}}$; the other inclusion holds trivially. ◀

Hence, we can conclude (using Lemma 4.8) that $\hat{e}$ is closed as well, as follows.

▶ **Corollary 5.11** (Transformation yields closed term). *Let $e \in \mathcal{T}_{GR}$; now $\hat{e}$ is closed.*

We are now ready conclude with the main result of this section.

▶ **Theorem 5.12** (Soundness and completeness). *Let $e, f \in \mathcal{T}_{GR}$; then*

$$e \equiv_{\mathsf{KAO}} f \iff [\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}} \ .$$

Sketch. Since we have a way to obtain from $e \in \mathcal{T}_{GR}$ a closed term $\hat{e} \in \mathcal{T}_{AGR}$, for which it furthermore holds that $e \equiv_{\mathsf{KAO}} \hat{e}$, we can appeal to Proposition 5.2. ◀

6 Decision procedure

We now design a procedure that takes terms $e, f \in \mathcal{T}_{GR}$ and decides whether $[\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}}$; by Theorem 5.12, this gives us a decision procedure for $e \equiv_{\mathsf{KAO}} f$. To this end, we reduce to the problem of language equivalence between NFAs over guarded rational terms, defined using partial derivatives. This, in turn, can be efficiently decided using bisimulation techniques.

First, observe that $\mathcal{T}_{GR}$ carries a non-deterministic automaton structure.

▶ **Definition 6.1** (Syntactic automaton). *The set $\mathcal{T}_{GR}$ carries an NDA structure $(\mathcal{T}_{GR}, \epsilon, \theta)$ where $\epsilon : \mathcal{T}_{GR} \to 2$ is as in Definition 4.10, and $\theta : \mathcal{T}_{GR} \times \Gamma \to 2^{\mathcal{T}_{GR}}$ is given by*

$$\theta(e, \mathfrak{a}) = \begin{cases} \delta(e, \mathfrak{a}) & \text{if } \mathfrak{a} \in \Sigma \\ \zeta(e, \mathfrak{a}) & \text{if } \mathfrak{a} \in \mathcal{A} \end{cases} \ .$$

We call this the syntactic automaton *of guarded rational terms.*

Let $\ell \colon \mathcal{T}_{GR} \to 2^{\Gamma^\star}$ be the semantics of this automaton as given in Section 2. It is easy to see that ℓ is the unique function such that

$$\ell(e) = \{\epsilon : \epsilon(e) = 1\} \cup \bigcup_{e' \in \delta(e,a)} \{a\} \cdot \ell(e') \cup \bigcup_{e' \in \zeta(e,\alpha)} \{\alpha\} \cdot \ell(e') \ . \tag{1}$$

To use this NDA for KAO equivalence, we need to show that the language accepted by the state $e \in \mathcal{T}_{GR}$ is $[\![e]\!]_{\mathsf{KAO}}$. For this goal, it helps to algebraically characterise expressions in terms of their derivatives. We call this a *fundamental theorem* of KAO, after [39, 40].

▶ **Theorem 6.2** (Fundamental theorem). *For all $e \in \mathcal{T}_{GR}$, the following holds*

$$e \equiv_{\mathsf{KAO}} \epsilon(e) + \sum_{e' \in \delta(e,a)} a \cdot e' + \sum_{e' \in \zeta(e,\alpha)} \pi_\alpha \cdot e' \ .$$

Sketch. The right-to-left containment is a result of Lemmas 4.11 and 4.15. The converse is a straightforward calculation using Proposition 5.6 and Corollary 5.8. ◀

Next, we turn the fundamental theorem into a statement about the KAO semantics. Before we do, however, we need the following basic result about the KAO semantics.

▶ **Lemma 6.3.** *Let us fix $\alpha \in \mathcal{A}$. For all $e \in \mathcal{T}_{GR}$, we have:*

$$\left[\!\!\left[\sum_{e' \in \zeta(e,\alpha)} \pi_\alpha \cdot e' \right]\!\!\right]_{\mathsf{KAO}} = \bigcup_{e' \in \zeta(e,\alpha)} \{\alpha\} \cdot [\![e']\!]_{\mathsf{KAO}} \ .$$

We now arrive at a semantic analogue of the fundamental theorem.

▶ **Proposition 6.4.** *For all $e \in \mathcal{T}_{GR}$, we have:*

$$[\![e]\!]_{\mathsf{KAO}} = \{\epsilon : \epsilon(e) = 1\} \cup \bigcup_{e' \in \delta(e,a)} \{a\} \cdot [\![e']\!]_{\mathsf{KAO}} \cup \bigcup_{e' \in \zeta(e,\alpha)} \{\alpha\} \cdot [\![e']\!]_{\mathsf{KAO}} \ .$$

Sketch. By applying soundness of $\equiv_{\mathsf{KAO}}$ w.r.t. $[\![-]\!]_{\mathsf{KAO}}$ (Lemma 4.9) to the fundamental theorem, and using Lemma 6.3 for the term summing over the $\mathcal{A}$-continuations of e. ◀

The language of a state in the syntactic NFA is then characterised by the KAO semantics.

▶ **Theorem 6.5** (Soundness of translation). *Let $e \in \mathcal{T}_{GR}$; then $[\![e]\!]_{\mathsf{KAO}} = \ell(e)$.*

Sketch. A direct consequence of Proposition 6.4 and the uniqueness of ℓ in satisfying (1). ◀

To decide whether $[\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}}$ it suffices to show that e and f are language equivalent in the syntactic automaton. We express this in terms of ι as defined in Definition 4.18.

▶ **Corollary 6.6.** *For all $e, f \in \mathcal{T}_{GR}$, we have*

$$[\![e]\!]_{\mathsf{KAO}} = [\![f]\!]_{\mathsf{KAO}} \quad \Leftrightarrow \quad \bigcup_{e' \in \iota(e)} \ell(e') = \bigcup_{f' \in \iota(f)} \ell(f') \ .$$

By construction, $\iota(e) \subseteq \rho(e)$ for every $e \in \mathcal{T}_{GR}$. Since $\rho(e)$ and $\rho(f)$ are closed under partial derivatives, we can restrict the syntactic automaton to $\rho(e) \cup \rho(f)$, to obtain a *finite* NDA. To decide $e \equiv_{\mathsf{KAO}} f$, it suffices to decide $\bigcup_{e' \in \iota(e)} \ell(e') = \bigcup_{f' \in \iota(f)} \ell(f')$ on this NFA.

This leads us to the main result of this section: a decision procedure for KAO.

▶ **Theorem 6.7** (Decision procedure). *For all $e, f \in \mathcal{T}_{GR}$, we have $e \equiv_{\mathsf{KAO}} f$ if and only if there exists R such that $(\iota(e), \iota(f)) \in R$ and R is a bisimulation up to congruence for the syntactic automaton restricted to $\rho(e) \cup \rho(f)$.*

Sketch. By Theorem 2.5, such an R exists precisely when $\bigcup_{e' \in \iota(e)} [\![e']\!]_{\mathsf{KAO}} = \bigcup_{f' \in \iota(f)} [\![f']\!]_{\mathsf{KAO}}$; by Corollary 6.6 and Theorem 5.12 this is equivalent to $e \equiv_{\mathsf{KAO}} f$. ◀

▶ **Example 6.8.** We know that for all $a, b \in \Omega$ we have that $a \wedge b \not\equiv_{\mathsf{KAO}} a \cdot b$. This also follows from our decision procedure, which we argue by showing that any attempt to construct a bisimulation up to congruence R containing $(\{a \wedge b\}, \{a \cdot b\})$ fails.

We start with $R = \{(\{a \wedge b\}, \{a \cdot b\})\}$, and note that $\bar{\epsilon}(\{a \wedge b\}) = 0 = \bar{\epsilon}(\{a \cdot b\})$. We now take a derivative w.r.t. $\alpha \in \mathcal{A}$ such that $\pi_\alpha \leqq_{\mathsf{BA}} a \wedge b$. To grow R into a bisimulation up to congruence, all derivatives should be checked and possibly added to R; this specific choice, however, will lead to a counterexample. We have that $\bar{\theta}(\{a \wedge b\}, \alpha) = \{1\}$ and $\bar{\theta}(\{a \cdot b\}, \alpha) = \{1 \cdot b, 1\}$. We add $(\{1\}, \{1 \cdot b, 1\})$ to R, noting that $\bar{\epsilon}(\{1\}) = 1 = \bar{\epsilon}(\{1 \cdot b, 1\})$, and continue with the next derivative w.r.t. $\beta \in \mathcal{A}$ such that $\pi_\beta \leqq_{\mathsf{BA}} b$. We get $\bar{\theta}(\{1\}, \beta) = \emptyset$ and $\bar{\theta}(\{1 \cdot b, 1\}, \beta) = \{1\}$. We now have $(\emptyset, \{1\}) \in R$, but $\bar{\epsilon}(\emptyset) \neq \bar{\epsilon}(\{1\})$. Thus, we cannot construct a bisimulation up to congruence R such that $(\{a \wedge b\}, \{a \cdot b\}) \in R$.

▶ Remark 6.9. For KAO-expressions without observations, the derivatives with respect to an element of $\mathcal{A}$ result in the empty set. Hence, for these KAO-expressions, deciding equivalence comes down to standard derivative-based techniques for rational expressions [37].

7 Related work

This work fits in the larger tradition of Kleene algebra as a presentation of the "laws of programming", the latter having been studied by Hoare and collaborators [14, 13]. More precisely, our efforts can be grouped with recent efforts to extend Kleene algebra with concurrency [15, 13, 32, 21], and thence with Boolean guards [19].

We proved that $\equiv_{\mathsf{KAO}}$ is sound and complete w.r.t. $[\![-]\!]_{\mathsf{KAO}}$, based on the existing completeness proof for KA. By re-using completeness results for a simpler algebra, the proof shows a clear separation of concerns between the "old" algebra being extended and the new layer of axioms placed on top. This strategy pops up quite often in some form [29, 1, 32, 21]. We note that unlike [29, 1], our transformation does not proceed by induction on the term, but leverages the least fixpoints computable in every Kleene algebra. Further, the combination of KA with additional hypotheses presented in [27] might yield another route to completeness.

The use of linear systems to study automata was pioneered by Conway [9] and Backhouse [3]. Kozen's completeness proof expanded on this by generalising Kleene algebra to matrices [23]. The connection between linear systems, derivatives and completeness was studied by Kozen [26] and Jacobs [18]. To keep our presentation simple, we give our proof of completeness in elementary terms; a proof in terms of coalgebra [38] may yet be possible.

Using bisimulation to decide language equivalence in a deterministic finite automaton originates from Hopcroft and Karp [17]. This technique has many generalisations [36].

8 Conclusions and further work

Kleene algebra with observations (KAO) is an algebraic framework that adds Boolean guards to Kleene algebra in such a way that a sensible extension with concurrency is still possible, in contrast with Kleene algebra with tests (KAT). Indeed, the laws of KAO prevent the problem presented by Antinomy 3.1. The axiomatisation of KAO, as well as the decision procedure for equivalence, give an alternative foundation for combining KAO with concurrent Kleene algebra to arrive at a new equational calculus of concurrent programs.

The most obvious direction of further work is to define *concurrent Kleene algebra with observations* (*CKAO*) as the amalgamation of axioms of KAO and CKA, in pursuit of a characterisation of its equational theory and an analogous decidability result. We conjecture

that languages of *sp-pomsets* [12, 11, 33] over actions and atoms, closed under some suitable relation, form the free model; a proof would likely build on [31] and re-use techniques from [21]. While the equational theory of CKAO might be decidable, we are less optimistic about a feasible algorithm; deciding the equational theory of CKA is already EXPSPACE-complete [7].

Another avenue of future research would be to create a programming language using KAO (or, possibly, CKAO) by instantiating actions and observations, and adding axioms that encode the intention of those primitives. NetKAT [1, 10, 41], a language for describing and reasoning about network behaviour is an excellent example of such an endeavour based on KAT. Our long-term hope is that CKAO will function as a foundation for a "concurrent" version of NetKAT aimed at describing and reasoning about networks with concurrency.

Finally, we note that the decision procedure for KAO based on bisimulation up to congruence leaves room for optimisation. Besides adapting the work on symbolic algorithms for bisimulation-based algorithms in KAT [34], the transitivity property witnessed in Lemma 4.14 seems like it could sometimes allow a bisimulation-based algorithm to decide early.

References

1 Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: semantic foundations for networks. In *POPL*, pages 113–126, 2014. doi:10.1145/2535838.2535862.

2 Valentin M. Antimirov. Partial Derivatives of Regular Expressions and Finite Automaton Constructions. *Theor. Comput. Sci.*, 155(2):291–319, 1996. doi:10.1016/0304-3975(95)00182-4.

3 Roland Backhouse. *Closure algorithms and the star-height problem of regular languages*. PhD thesis, University of London, 1975.

4 Garrett Birkhoff and Thomas C. Bartee. *Modern applied algebra*. McGraw-Hill, 1970.

5 Maurice Boffa. Une remarque sur les systèmes complets d'identités rationnelles. *ITA*, 24:419–428, 1990.

6 Filippo Bonchi and Damien Pous. Checking NFA equivalence with bisimulations up to congruence. In *POPL*, pages 457–468, 2013. doi:10.1145/2429069.2429124.

7 Paul Brunet, Damien Pous, and Georg Struth. On Decidability of Concurrent Kleene Algebra. In *CONCUR*, pages 28:1–28:15, 2017. doi:10.4230/LIPIcs.CONCUR.2017.28.

8 Janusz A. Brzozowski. Derivatives of Regular Expressions. *J. ACM*, 11(4):481–494, 1964. doi:10.1145/321239.321249.

9 John Horton Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

10 Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A Coalgebraic Decision Procedure for NetKAT. In *POPL*, pages 343–355, 2015. doi:10.1145/2676726.2677011.

11 Jay L. Gischer. The Equational Theory of Pomsets. *Theor. Comput. Sci.*, 61:199–224, 1988. doi:10.1016/0304-3975(88)90124-7.

12 Jan Grabowski. On partial languages. *Fundam. Inform.*, 4(2):427, 1981.

13 Tony Hoare. Laws of Programming: The Algebraic Unification of Theories of Concurrency. In *CONCUR*, pages 1–6, 2014. doi:10.1007/978-3-662-44584-6_1.

14 Tony Hoare, Ian J. Hayes, Jifeng He, Carroll Morgan, A. W. Roscoe, Jeff W. Sanders, Ib Holm Sørensen, J. Michael Spivey, and Bernard Sufrin. Laws of Programming. *Commun. ACM*, 30(8):672–686, 1987. doi:10.1145/27651.27653.

15 Tony Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent Kleene Algebra. In *CONCUR*, pages 399–414, 2009. doi:10.1007/978-3-642-04081-8_27.

16 Tony Hoare, Stephan van Staden, Bernhard Möller, Georg Struth, and Huibiao Zhu. Developments in Concurrent Kleene Algebra. *J. Log. Algebr. Meth. Program.*, 85(4):617–636, 2016. doi:10.1016/j.jlamp.2015.09.012.

17 John E. Hopcroft and Richard M. Karp. A linear algorithm for testing equivalence of finite automata. Technical Report TR71-114, Cornell University, December 1971.

18 Bart Jacobs. A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages. In *Algebra, Meaning, and Computation (Joseph A. Goguen festschrift)*, pages 375–404, 2006. doi:10.1007/11780274_20.

19 Peter Jipsen and M. Andrew Moshier. Concurrent Kleene algebra with tests and branching automata. *J. Log. Algebr. Meth. Program.*, 85(4):637–652, 2016. doi:10.1016/j.jlamp.2015.12.005.

20 Tobias Kappé, Paul Brunet, Jurriaan Rot, Alexandra Silva, Jana Wagemaker, and Fabio Zanasi. Kleene Algebra with Observations. *CoRR*, abs/1811.10401, 2018. arXiv:1811.10401.

21 Tobias Kappé, Paul Brunet, Alexandra Silva, and Fabio Zanasi. Concurrent Kleene Algebra: Free Model and Completeness. In *ESOP*, pages 856–882, 2018. doi:10.1007/978-3-319-89884-1_30.

22 Stephen C. Kleene. Representation of Events in Nerve Nets and Finite Automata. *Automata Studies*, pages 3–41, 1956.

23 Dexter Kozen. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Inf. Comput.*, 110(2):366–390, 1994. doi:10.1006/inco.1994.1037.

24 Dexter Kozen. Kleene algebra with tests and commutativity conditions. In *TACAS*, pages 14–33, 1996. doi:10.1007/3-540-61042-1_35.

25 Dexter Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. Comput. Log.*, 1(1):60–76, 2000. doi:10.1145/343369.343378.

26 Dexter Kozen. Myhill-Nerode Relations on Automatic Systems and the Completeness of Kleene Algebra. In *STACS*, pages 27–38, 2001. doi:10.1007/3-540-44693-1_3.

27 Dexter Kozen and Konstantinos Mamouras. Kleene Algebra with Equations. In *ICALP*, pages 280–292, 2014. doi:10.1007/978-3-662-43951-7_24.

28 Dexter Kozen and Maria-Christina Patron. Certification of Compiler Optimizations Using Kleene Algebra with Tests. In *CL*, pages 568–582, 2000. doi:10.1007/3-540-44957-4_38.

29 Dexter Kozen and Frederick Smith. Kleene Algebra with Tests: Completeness and Decidability. In *CSL*, pages 244–259, 1996. doi:10.1007/3-540-63172-0_43.

30 Daniel Krob. Complete Systems of B-Rational Identities. *Theor. Comput. Sci.*, 89(2):207–343, 1991. doi:10.1016/0304-3975(91)90395-I.

31 Michael R. Laurence and Georg Struth. Completeness Theorems for Bi-Kleene Algebras and Series-Parallel Rational Pomset Languages. In *RAMiCS*, pages 65–82, 2014. doi:10.1007/978-3-319-06251-8_5.

32 Michael R. Laurence and Georg Struth. Completeness Theorems for Pomset Languages and Concurrent Kleene Algebras, 2017. arXiv:1705.05896.

33 Kamal Lodaya and Pascal Weil. Series-parallel languages and the bounded-width property. *Theor. Comput. Sci.*, 237(1):347–380, 2000. doi:10.1016/S0304-3975(00)00031-1.

34 Damien Pous. Symbolic Algorithms for Language Equivalence and Kleene Algebra with Tests. In *POPL*, pages 357–368, 2015. doi:10.1145/2676726.2677007.

35 Michael O. Rabin and Dana S. Scott. Finite Automata and Their Decision Problems. *IBM J. of Research and Dev.*, 3(2):114–125, 1959. doi:10.1147/rd.32.0114.

36 Jurriaan Rot, Filippo Bonchi, Marcello M. Bonsangue, Damien Pous, Jan Rutten, and Alexandra Silva. Enhanced coalgebraic bisimulation. *Mathematical Structures in Comput. Sci.*, 27(7):1236–1264, 2017. doi:10.1017/S0960129515000523.

37 Jan J. M. M. Rutten. Automata and Coinduction (An Exercise in Coalgebra). In *CONCUR*, pages 194–218, 1998. doi:10.1007/BFb0055624.

38 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

39 Jan J. M. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comput. Sci.*, 308(1-3):1–53, 2003. doi:10.1016/S0304-3975(02)00895-2.

40 Alexandra Silva. *Kleene Coalgebra*. PhD thesis, Radboud Universiteit Nijmegen, 2010.

41 Steffen Smolka, Spiridon Aristides Eliopoulos, Nate Foster, and Arjun Guha. A fast compiler for NetKAT. In *ICFP*, pages 328–341, 2015. doi:10.1145/2784731.2784761.