Completeness and Incompleteness of Synchronous Kleene Algebra

Jana Wagemaker[1(✉)], Marcello Bonsangue[2], Tobias Kappé[1], Jurriaan Rot[1,3], and Alexandra Silva[1]

[1] University College London, London, UK
j.wagemaker@ucl.ac.uk
[2] Leiden University, Leiden, The Netherlands
[3] Radboud University, Nijmegen, The Netherlands

Abstract. *Synchronous Kleene algebra* (*SKA*), an extension of Kleene algebra (KA), was proposed by Prisacariu as a tool for reasoning about programs that may execute synchronously, i.e., in lock-step. We provide a countermodel witnessing that the axioms of SKA are incomplete w.r.t. its language semantics, by exploiting a lack of interaction between the *synchronous product* operator and the Kleene star. We then propose an alternative set of axioms for SKA, based on Salomaa's axiomatisation of regular languages, and show that these provide a sound and complete characterisation w.r.t. the original language semantics.

1 Introduction

Kleene algebra (*KA*) is applied in various contexts, such as relational algebra and automata theory. An important use of KA is as a logic of programs. This is because the axioms of KA correspond well to properties expected of sequential program composition, and hence they provide a logic for reasoning about control flow of sequential programs presented as Kleene algebra expressions. Regular languages then provide a canonical semantics for programs expressed in Kleene algebra, due to a tight connection between regular languages and the axioms of KA: an equation is provable using the Kleene algebra axioms if and only if the corresponding regular languages coincide [5,18,21].

In [24], Prisacariu proposes an extension of Kleene algebra, called *synchronous Kleene algebra* (*SKA*). The aim was to introduce an algebra useful for studying not only sequential programs but also *synchronous* concurrent programs. Here, synchrony is understood as in Milner's SCCS [23], i.e., each program executes a single action instantaneously at each discrete time step. Hence, the synchrony paradigm assumes that basic actions execute in one unit of time and that at each time step, all components capable of acting will do so. This model permits a *synchronous product* operator, which yields a program that, at each

This work was partially supported by ERC Starting Grant ProFoundNet (679127), a Leverhulme Prize (PLP–2016–129) and a Marie Curie Fellowship (795119). The first author conducted part of this work at Centrum Wiskunde & Informatica, Amsterdam.

G. Hutton (Ed.): MPC 2019, LNCS 11825, pp. 385–413, 2019.
https://doi.org/10.1007/978-3-030-33636-3_14

time step, executes some combination of the actions put forth by the operand programs.

This new operator is governed by various expected axioms such as associativity and commutativity. Another axiom describes the interaction between the synchronous product and the sequential product, capturing the intended lockstep behaviour. Crucially, the axioms do not entail certain equations that relate the Kleene star (used to describe loops) and the synchronous product.

The contributions of this paper are twofold. First, we show that the lack of connection between the Kleene star and the synchronous product is problematic. In particular, we exploit this fact to devise a countermodel that violates a semantically valid equation, thus showing that the SKA axioms are incomplete w.r.t. the language semantics. This invalidates the completeness result in [24].

The second and main contribution of this paper is a sound and complete characterisation of the equational theory of SKA in terms of a generalisation of regular languages. The key difference with [24] is the shift from *least* fixpoint axioms in the style of Kozen [18] to a *unique* fixpoint axiom in the style of Salomaa [26]. In the completeness proof, we give a reduction to the completeness result of Salomaa via a normal form for SKA expressions. As a by-product, we get a proof of the correctness of the partial derivatives for SKA provided in [7].

This paper is organised as follows. In Sect. 2 we discuss the necessary preliminaries. In Sect. 3 we discuss SKA as presented in [24]. Next, in Sect. 4, we demonstrate why SKA is incomplete, and in Sect. 5 go on to provide a new set of axioms, which we call SF_1. The latter section also includes basic results about the partial derivatives for SKA from [7]. In Sect. 6 we provide an algebraic characterisation of SF_1-terms; this characterisation is used in Sect. 7, where we prove completeness of SF_1 w.r.t. to its language model. In Sect. 8 we consider related work and conclude by discussing directions for future work in Sect. 9. For the sake of readability, some of the proofs appear in the appendix.

2 Preliminaries

Throughout this paper, we write 2 for the two-element set $\{0, 1\}$.

Languages. Throughout the paper we fix a finite alphabet Σ. A *word* formed over Σ is a finite sequence of symbols from Σ. The *empty word* is denoted by ε. We write Σ^* for the set of all words over Σ. *Concatenation* of words $u, v \in \Sigma^*$ is denoted by $uv \in \Sigma^*$. A *language* is a set of words. For $K, L \subseteq \Sigma^*$, we define

$$K \cdot L = \{uv : u \in K, v \in L\} \qquad K + L = K \cup L \qquad K^* = \bigcup\nolimits_{n \in \mathbb{N}} K^n,$$

where $K^0 = \{\varepsilon\}$ and $K^{n+1} = K \cdot K^n$.

Kleene Algebra. We define a *Kleene algebra* [18] as a tuple $(A, +, \cdot, ^*, 0, 1)$ where A is a set, * is a unary operator, $+$ and $\cdot$ are binary operators and 0 and 1 are constants. Moreover, for all $e, f, g \in A$ the following axioms are satisfied:

$$e + (f + g) = (e + f) + g \qquad e + f = f + e \qquad e + 0 = e \qquad e + e = e$$
$$e \cdot 1 = e = 1 \cdot e \qquad e \cdot 0 = 0 = 0 \cdot e \qquad e \cdot (f \cdot g) = (e \cdot f) \cdot g$$
$$e^* = 1 + e \cdot e^* = 1 + e^* \cdot e \qquad (e + f) \cdot g = e \cdot g + f \cdot g \qquad e \cdot (f + g) = e \cdot f + e \cdot g$$

Additionally, we write $e \leq f$ as a shorthand for $e + f = f$, and require that the *least fixpoint axioms* [18] hold, which stipulate that for $e, f, g \in A$ we have

$$e + f \cdot g \leq g \implies f^* \cdot e \leq g \qquad e + f \cdot g \leq f \implies e \cdot g^* \leq f$$

The set of *regular expressions*, denoted $\mathcal{T}_{\mathsf{KA}}$, is described by the grammar:

$$\mathcal{T}_{\mathsf{KA}} \ni e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^*$$

Regular expressions can be interpreted in terms of languages. This is done by defining $[\![-]\!]_{\mathsf{KA}} : \mathcal{T}_{\mathsf{KA}} \to \mathcal{P}(\Sigma^*)$ inductively, as follows.

$$[\![0]\!]_{\mathsf{KA}} = \emptyset \qquad [\![a]\!]_{\mathsf{KA}} = \{a\} \qquad [\![e \cdot f]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}} \cdot [\![f]\!]_{\mathsf{KA}}$$
$$[\![1]\!]_{\mathsf{KA}} = \{\varepsilon\} \qquad [\![e + f]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}} + [\![f]\!]_{\mathsf{KA}} \qquad [\![e^*]\!]_{\mathsf{KA}} = [\![e]\!]^*_{\mathsf{KA}}$$

A language L is called *regular* if and only if $L = [\![e]\!]_{\mathsf{KA}}$ for some $e \in \mathcal{T}_{\mathsf{KA}}$.

We write $\equiv_{\mathsf{KA}}$ for the smallest congruence on $\mathcal{T}_{\mathsf{KA}}$ induced by the Kleene algebra axioms—e.g., for all $e \in \mathcal{T}_{\mathsf{KA}}$, we have $1 + e \cdot e^* \equiv_{\mathsf{KA}} e^*$. Intuitively, $e \equiv_{\mathsf{KA}} f$ means that the regular expressions e and f can be proved equivalent according to the axioms of Kleene algebra. A pivotal result in the study of Kleene algebras tells us that $[\![-]\!]_{\mathsf{KA}}$ characterises $\equiv_{\mathsf{KA}}$, in the following sense:

Theorem 2.1 (Soundness and Completeness of KA [18]). *For all $e, f \in \mathcal{T}_{\mathsf{KA}}$, we have that $e \equiv_{\mathsf{KA}} f$ if and only if $[\![e]\!]_{\mathsf{KA}} = [\![f]\!]_{\mathsf{KA}}$.*

Remark 2.2. The above can be generalised, as follows. Let $\mathcal{K} = (A, +, \cdot, ^*, 0, 1)$ be a KA, and let $\sigma : \Sigma \to A$. Then for all $e, f \in \mathcal{T}_{\mathsf{KA}}$ such that $e \equiv_{\mathsf{KA}} f$, interpreting e and f according to σ in $\mathcal{K}$ yields the same result. For instance, since $(a^*)^* \equiv_{\mathsf{KA}} a^*$, we know that for *any* element e of *any* KA $\mathcal{K}$, we have that $(e^*)^* = e$.

Linear Systems. Let Q be a finite set. A *Q-vector* is a function $x : Q \to \mathcal{T}_{\mathsf{KA}}$. A *$Q$-matrix* is a function $M : Q \times Q \to \mathcal{T}_{\mathsf{KA}}$. Let x and y be Q-vectors. Addition is defined pointwise, setting $(x + y)(q) = x(q) + y(q)$. Multiplication by a Q-matrix M is given by

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

When $x(q) \equiv_{\mathsf{KA}} y(q)$ for all $q \in Q$, we write $x \equiv_{\mathsf{KA}} y$.

Definition 2.3. *A Q-*linear system *is a pair (M, x) with M a Q-matrix and x a Q-vector. A solution to (M, x) in KA is a Q-vector y such that $M \cdot y + x \equiv_{\mathsf{KA}} y$.*

Non-deterministic Finite Automata. A *non-deterministic automaton (NDA)* over an alphabet Σ is a triple (X, o, d) where $o\colon X \to 2$ is called the *termination function* and $d\colon X \times \Sigma \to X$ called the *continuation function*. If X is finite, (X, o, d) is referred to as a *non-deterministic finite automaton (NFA)*.

The semantics of an NDA (X, o, d) can be characterised recursively as the unique map $\ell : X \to \mathcal{P}(\Sigma^*)$ such that

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

This coincides with the standard definition of language acceptance.

3 Synchronous Kleene Algebra

Synchronous Kleene algebra extends Kleene algebra with an additional operator denoted $\times$, which we refer to as the synchronous product [24].

Definition 3.1 (Synchronous Kleene Algebra). *A* synchronous KA (SKA) *is a tuple* $(A, S, +, \cdot, ^*, \times, 0, 1)$ *such that* $(A, +, \cdot, ^*, 0, 1)$ *is a Kleene algebra and* $\times$ *is a binary operator on* A*, with* $S \subseteq A$ *closed under* $\times$ *and* $(S, \times)$ *a semilattice. Furthermore, the following hold for all* $e, f, g \in A$ *and* $\alpha, \beta \in S$*:*

$$e \times (f + g) = e \times f + e \times g \qquad e \times (f \times g) = (e \times f) \times g \qquad e \times 0 = 0$$
$$(\alpha \cdot e) \times (\beta \cdot f) = (\alpha \times \beta) \cdot (e \times f) \qquad e \times f = f \times e \qquad e \times 1 = e$$

Note that 0 and 1 need not be elements of S. The *semilattice terms*, denoted $\mathcal{T}_{\mathsf{SL}}$, are given by the following grammar.

$$\mathcal{T}_{\mathsf{SL}} \ni e, f ::= a \in \Sigma \mid e \times f$$

The *synchronous regular terms*, denoted $\mathcal{T}_{\mathsf{SKA}}$, are given by the grammar:

$$\mathcal{T}_{\mathsf{SKA}} \ni e, f ::= 0 \mid 1 \mid a \in \mathcal{T}_{\mathsf{SL}} \mid e + f \mid e \cdot f \mid e \times f \mid e^*$$

Thus we have $\mathcal{T}_{\mathsf{SL}} \subseteq \mathcal{T}_{\mathsf{SKA}}$. We then define $\equiv_{\mathsf{SKA}}$ as the smallest congruence on $\mathcal{T}_{\mathsf{SKA}}$ satisfying the axioms of SKA Here, $\mathcal{T}_{\mathsf{SL}}$ plays the role of the semilattice; for instance, for $a \in \mathcal{T}_{\mathsf{SL}}$ we have that $a \times a \equiv_{\mathsf{SKA}} a$.

Remark 3.2. In [24], $\times$ is declared to be idempotent on the *generators* of the semilattice, whereas in our definition it holds for semilattice elements in general. This does not change anything, as the axiom $a \times a = a$ for generators together with commutativity and associativity results in idempotence on the semilattice. We present SKA as in Definition 3.1 to prevent a meta-definition of a third sort (namely the semilattice generated by Σ) present in the signature of the algebra. We have also left out the second distributivity and unit axioms that follow immediately from the ones presented and commutativity.

3.1 A Language Model for SKA

Similar to Kleene algebra, there is a language model for SKA [24].

Words over $\mathcal{P}(\Sigma) \setminus \{\emptyset\} = \mathcal{P}_n(\Sigma)$ are called *synchronous strings*, and sets of synchronous strings are called *synchronous languages*. The standard language operations (sum, concatenation, Kleene closure) are also defined on synchronous languages. The synchronous product of synchronous languages K, L is given by:

$$K \times L = \{u \times v : u \in K, v \in L\}$$

where we define $\times$ inductively for $u, v \in (\mathcal{P}_n(\Sigma))^*$ and $x, y \in \mathcal{P}_n(\Sigma)$, as follows:

$$u \times \varepsilon = u = \varepsilon \times u \qquad \text{and} \qquad (x \cdot u) \times (y \cdot v) = (x \cup y) \cdot (u \times v)$$

To define the language semantics for all elements in $\mathcal{T}_{\mathsf{SKA}}$, we first give an interpretation of elements in $\mathcal{T}_{\mathsf{SL}}$ in terms of non-empty finite subsets of Σ.

Definition 3.3. *For $a \in \Sigma$ and $e, f \in \mathcal{T}_{\mathsf{SL}}$, define $[\![-]\!]_{\mathsf{SL}} : \mathcal{T}_{\mathsf{SL}} \to \mathcal{P}_n(\Sigma)$ by*

$$[\![a]\!]_{\mathsf{SL}} = \{a\} \qquad [\![e \times f]\!]_{\mathsf{SL}} = [\![e]\!]_{\mathsf{SL}} \cup [\![f]\!]_{\mathsf{SL}}$$

Denote the smallest congruence on $\mathcal{T}_{\mathsf{SL}}$ with respect to idempotence, associativity and commutativity of $\times$ with $\equiv_{\mathsf{SL}}$. It is not hard to show that $[\![-]\!]_{\mathsf{SL}}$ characterises $\equiv_{\mathsf{SL}}$, in the following sense.

Lemma 3.4 (Soundness and Completeness of SL). *For all $e, f \in \mathcal{T}_{\mathsf{SL}}$, we have $[\![e]\!]_{\mathsf{SL}} = [\![f]\!]_{\mathsf{SL}}$ if and only if $e \equiv_{\mathsf{SL}} f$.*

The semantics of synchronous regular terms is given in terms of a mapping to synchronous languages: $[\![-]\!]_{\mathsf{SKA}} : \mathcal{T}_{\mathsf{SKA}} \to \mathcal{P}((\mathcal{P}_n(\Sigma))^*)$. We have:

$$\begin{aligned} &[\![0]\!]_{\mathsf{SKA}} = \emptyset \quad [\![1]\!]_{\mathsf{SKA}} = \{\varepsilon\} \quad [\![a]\!]_{\mathsf{SKA}} = \{[\![a]\!]_{\mathsf{SL}}\} \quad \forall a \in \mathcal{T}_{\mathsf{SL}} \quad [\![e^*]\!]_{\mathsf{SKA}} = [\![e]\!]^*_{\mathsf{SKA}} \\ &[\![e \cdot f]\!]_{\mathsf{SKA}} = [\![e]\!]_{\mathsf{SKA}} \cdot [\![f]\!]_{\mathsf{SKA}} \quad [\![e + f]\!]_{\mathsf{SKA}} = [\![e]\!]_{\mathsf{SKA}} + [\![f]\!]_{\mathsf{SKA}} \quad [\![e \times f]\!]_{\mathsf{SKA}} = [\![e]\!]_{\mathsf{SKA}} \times [\![f]\!]_{\mathsf{SKA}} \end{aligned} \tag{2}$$

A synchronous language L is called *regular* when $L = [\![e]\!]_{\mathsf{SKA}}$ for some $e \in \mathcal{T}_{\mathsf{SKA}}$.

Let $S = \{\{x\} : x \in \mathcal{P}_n(\Sigma)\}$, that is to say, S is the set of synchronous languages consisting of a single word, whose single letter is in turn a subset of Σ. Furthermore, let $\mathcal{L}_\Sigma$ denote the set of synchronous languages over Σ. It is straightforward to prove that $\mathcal{L}_\Sigma$ together with S is closed under the SKA operations and satisfies the SKA axioms [24]; more precisely, we have:

Lemma 3.5. *The structure $(\mathcal{L}_\Sigma, S, +, \cdot, ^*, \times, \emptyset, \{\varepsilon\})$ is an SKA, that is, synchronous languages over Σ form an SKA.*

As a consequence of Lemma 3.5, we obtain soundness of the SKA axioms with respect to the language model based on synchronous regular languages:

Lemma 3.6 (Soundness of SKA). *For all $e, f \in \mathcal{T}_{\mathsf{SKA}}$, we have that $e \equiv_{\mathsf{SKA}} f$ implies $[\![e]\!]_{\mathsf{SKA}} = [\![f]\!]_{\mathsf{SKA}}$.*

Remark 3.7. The above generalises almost analogously to Remark 2.2. Let $\mathcal{M}$ be an SKA with semilattice S, and let $\sigma : \Sigma \to S$ be a function. Then for all $e, f \in \mathcal{T}_{\mathsf{SKA}}$ such that $e \equiv_{\mathsf{SKA}} f$, if we interpret e in $\mathcal{M}$ according to σ, then we should get the same result as when we interpret f in $\mathcal{M}$ according to σ.

In other words, when $e \equiv_{\mathsf{SKA}} f$ holds, it follows that $e = f$ is a valid equation in *every* SKA, provided that the symbols from Σ are interpreted as elements of the semilattice. It is not hard to show that this claim does not hold when symbols from Σ can be interpreted as elements of the carrier at large.

4 Incompleteness of SKA

We now prove incompleteness of the SKA axioms as presented in [24]. Fix alphabet $\mathcal{A} = \{a\}$. First, note that the language model of SKA has the following property.

Lemma 4.1. *For $\alpha \in \mathcal{T}_{\mathsf{SL}}$, we have $[\![\alpha^* \times \alpha^*]\!]_{\mathsf{SKA}} = [\![\alpha^*]\!]_{\mathsf{SKA}}$.*

If $\equiv_{\mathsf{SKA}}$ were complete w.r.t. $[\![-]\!]_{\mathsf{SKA}}$, then the above implies that $a^* \times a^* \equiv_{\mathsf{SKA}} a^*$ holds. In this section, we present a countermodel where all the axioms of SKA are true, but $\alpha^* \times \alpha^* = \alpha^*$ does not hold for any $\alpha \in S$. This shows that $a^* \times a^* \not\equiv_{\mathsf{SKA}} a^*$; consequently, $\equiv_{\mathsf{SKA}}$ cannot be complete w.r.t. $[\![-]\!]_{\mathsf{SKA}}$.

Countermodel for SKA
We define our countermodel as follows. For the semilattice, let $S = \{\{\{s\}\}\}$, the set containing the synchronous language $\{\{s\}\}$. We denote the set of all synchronous languages over alphabet $\{s\}$ with $\mathcal{L}_s$; the carrier of our model is formed by $\mathcal{L}_s \cup \{\dagger\}$, where $\dagger$ is a symbol not found in $\mathcal{L}_s$. The symbol $\dagger$ exists only in the model, and not in the algebraic theory. It remains to define the SKA operators on this carrier, which we do as follows.

Definition 4.2. *An element of $\mathcal{L}_s \cup \{\dagger\}$ is said to be infinite when it is an infinite language. For $K, L \in \mathcal{L}_s \cup \{\dagger\}$, define the SKA operators as follows:*

$$K + L = \begin{cases} \dagger K \cup L & K = \dagger \vee L = \dagger \\ & \textit{otherwise} \end{cases}$$

$$K \cdot L = \begin{cases} \emptyset & K = \emptyset \vee L = \emptyset \\ \dagger & K = \dagger \vee L = \dagger \\ \{u \cdot v : u \in K, v \in L\} & \textit{otherwise} \end{cases}$$

$$K \times L = \begin{cases} \emptyset & K = \emptyset \vee L = \emptyset \\ \dagger & K = \dagger \vee L = \dagger \vee K, L \textit{ infinite} \\ \{u \times v : u \in K, v \in L\} & \textit{otherwise} \end{cases}$$

$$K^* = \begin{cases} \dagger & K = \dagger \\ \bigcup_{n \in \mathbb{N}} K^n & \textit{otherwise} \end{cases}$$

where $u \times v$ for $u \in K$ and $v \in L$ and K^n is as defined in Sect. 3. Here, the cases are given in order of priority—e.g., if $K = \emptyset$ and $L = \dagger$, then $K \cdot L = \emptyset$.

The intuition behind this model is that SKA has no axioms that relate to the synchronous execution of starred expressions, such as in $\alpha^* \times \alpha^*$, nor can such a relation be derived from the axioms, meaning that a model has some leeway in defining the outcome in such cases. Since the language of a starred expression is generally infinite, we choose $\times$ such that it diverges to the extra element $\dagger$ when given infinite languages as input; for the rest of the operators, the behaviour on $\dagger$ is chosen to comply with the axioms.

First, we verify that our operators satisfy the SKA axioms.

Lemma 4.3. *$\mathcal{M} = (\mathcal{L}_s \cup \{\dagger\}, \{\{\{s\}\}\}, +, \cdot, ^*, \times, \emptyset, \{\varepsilon\})$ with the operators as defined in Definition 4.2 forms an SKA.*

Proof. For the sake of brevity, we validate one of the least fixpoint axioms and the synchrony axiom; the other axioms are treated in the appendix.

Let $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. We verify that $K + L \cdot J \leq J \implies L^* \cdot K \leq J$. Assume that $K + L \cdot J \leq J$. If $J = \dagger$, then the result follows by definition of $\leq$ and our choice of $+$. Otherwise, if $J \in \mathcal{L}_s$, we distinguish two cases. If $L = \dagger$, then J must be $\emptyset$ (otherwise $J = \dagger$); hence $K = \emptyset$, and the claim holds. Lastly, if $L \in \mathcal{L}_s$, then $K \in \mathcal{L}_s$. In this case, all of the operands are languages, and thus the proof goes through as it does for KA.

For the synchrony axiom, we need only check

$$(A \cdot K) \times (A \cdot L) = (A \times A) \cdot (K \times L)$$

for $A = \{\{s\}\}$ as that is the only element in S. Let $K, L \in \mathcal{L}_s \cup \{\dagger\}$. If either K or L is $\emptyset$, both sides of the equation reduce to $\emptyset$. Otherwise, if K or L is $\dagger$, then both sides of the equation reduce to $\dagger$. If K and L are both infinite then $A \cdot K$ and $A \cdot L$ are infinite and the claim follows. In all the remaining cases where K and L are elements of $\mathcal{L}_s$ and at most one of them is infinite, the proof goes through as it does for synchronous regular languages (Lemma 3.6). □

This leads us to the following theorem:

Theorem 4.4. *The axioms of SKA presented in Definition 3.1 are incomplete. That is, there exist $e, f \in \mathcal{T}_{\mathsf{SKA}}$ such that $[\![e]\!]_{\mathsf{SKA}} = [\![f]\!]_{\mathsf{SKA}}$ but $e \not\equiv_{\mathsf{SKA}} f$.*

Proof. Take $a \in \mathcal{A}$. We know from Lemma 4.1 that $[\![a^* \times a^*]\!]_{\mathsf{SKA}} = [\![a^*]\!]_{\mathsf{SKA}}$. Now suppose $a^* \times a^* \equiv_{\mathsf{SKA}} a^*$. As our countermodel is an SKA that means in particular that $\{\{s\}\}^* \times \{\{s\}\}^* = \{\{s\}\}^*$ should hold (c.f. Remark 3.7). However, in this model we can calculate that $\{\{s\}\}^* \times \{\{s\}\}^* = \dagger \neq \{\{s\}\}^*$. Hence, we have a contradiction. Thus $a^* \times a^* \not\equiv_{\mathsf{SKA}} a^*$, rendering SKA incomplete. □

5 A New Axiomatisation

We now create an alternative algebraic formalism, which we call SF_1, and prove that its axioms are sound and complete w.r.t the model of synchronous regular languages. Whereas the definition of SKA relies on Kleene algebras (with *least fixpoint axioms*) as presented by Kozen [18], the definition of SF_1 builds on F_1-algebras (with a *unique fixpoint axiom*) as presented by Salomaa [26]. The axioms of Salomaa are strictly stronger than Kozen's [10], and we will see that the unique fixpoint axiom allows us to derive a connection between the synchronous product and the Kleene star, even though this connection is not represented in an axiom directly (see Remark 5.8).

Definition 5.1. *An* F_1-algebra *[26] is a tuple* $(A, +, \cdot, ^*, 0, 1, H)$ *where* A *is a set,* * *is a unary operator,* $+$ *and* $\cdot$ *are binary operators and* 0 *and* 1 *are constants, and such that for all* $e, f, g \in A$ *the following axioms are satisfied:*

$$
\begin{array}{llll}
e + (f + g) = (e + f) + g & e + f = f + e & e + 0 = e & e + e = e \\
e \cdot 1 = e = 1 \cdot e & e \cdot 0 = 0 = 0 \cdot e & e \cdot (f \cdot g) = (e \cdot f) \cdot g & \\
e^* = 1 + e \cdot e^* = 1 + e^* \cdot e & (e + f) \cdot g = e \cdot g + f \cdot g & e \cdot (f + g) = e \cdot f + e \cdot g &
\end{array}
$$

Additionally, the loop tightening *and* unique fixpoint axiom *hold:*

$$
(e + 1)^* = e^* \qquad H(f) = 0 \wedge e + f \cdot g = g \implies f^* \cdot e = g
$$

Lastly, we have the following axioms for H*:*

$$
\begin{array}{lll}
H(0) = 0 & H(e + f) = H(e) + H(f) & H(e^*) = (H(e))^* \\
H(1) = 1 & H(e \cdot f) = H(e) \cdot H(f) &
\end{array}
$$

In [26], an $e \in A$ with $H(e) = 1$ is said to have the *empty word property*, which will be reflected in the semantic interpretation of $H(e)$ stated below.

The set of F_1-*expressions*, denoted $\mathcal{T}_{\mathsf{F}_1}$, is described by:

$$
\mathcal{T}_{\mathsf{F}_1} \ni e, f ::= 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid H(e)
$$

We can interpret F_1-expressions in terms of languages through $[\![-]\!]_{\mathsf{F}_1} : \mathcal{T}_{\mathsf{F}_1} \to \mathcal{P}(\Sigma^*)$, defined analogously to $[\![-]\!]_{\mathsf{KA}}$, where furthermore for $e \in \mathcal{T}_{\mathsf{F}_1}$ we have

$$
[\![H(e)]\!]_{\mathsf{F}_1} = [\![e]\!]_{\mathsf{F}_1} \cap \{\varepsilon\}
$$

We write $\equiv_{\mathsf{F}_1}$ for the smallest congruence on $\mathcal{T}_{\mathsf{F}_1}$ induced by the F_1-axioms. Additionally, we require that for $a \in \Sigma$, we have $H(a) \equiv_{\mathsf{F}_1} 0$. A characterisation similar to Theorem 2.1 can then be established as follows[1]:

[1] Unlike [26], we include H in the syntax; one can prove that for any $e \in \mathcal{T}_{\mathsf{F}_1}$ it holds that $H(e) \equiv 0$ or $H(e) \equiv 1$, and hence any occurence of H can be removed from e. This is what allows us to apply the completeness result from op. cit. here.

Theorem 5.2 (Soundness and Completeness of F_1 [26]). *For all $e, f \in \mathcal{T}_{\mathsf{F}_1}$, we have that $e \equiv_{\mathsf{F}_1} f$ if and only if $[\![e]\!]_{\mathsf{F}_1} = [\![f]\!]_{\mathsf{F}_1}$.*

Remark 5.3. Kozen [18] noted that the above does not generalise along the same lines as in Remark 2.2. In particular, the axiom $H(a) \equiv_{\mathsf{SKA}} 0$ is not stable under substitution; for instance, if we interpret $H(a)$ according to the valuation $a \mapsto \{\epsilon\}$ in the F_1-algebra of languages, then we obtain $\{\epsilon\}$, whereas 0 is interpreted as $\emptyset$.

Definition 5.4. *A* synchronous F_1-algebra *(*SF_1-algebra *for short) is a tuple $(A, S, +, \cdot, ^*, 0, 1, H)$, such that $(A, +, \cdot, ^*, 0, 1, H)$ is an F_1-algebra and $\times$ is a binary operator on A, with $S \subseteq A$ closed under $\times$ and $(S, \times)$ a semilattice. Furthermore, the following hold for all $e, f, g \in A$ and $\alpha, \beta \in S$:*

$$e \times (f + g) = e \times f + e \times g \qquad e \times (f \times g) = (e \times f) \times g \qquad e \times 0 = 0$$
$$(\alpha \cdot e) \times (\beta \cdot f) = (\alpha \times \beta) \cdot (e \times f) \qquad e \times f = f \times e \qquad e \times 1 = e$$

Moreover, H is compatible with $\times$ as well, i.e., for $e, f \in A$ we have that $H(e \times f) = H(e) \times H(f)$. Lastly, for $\alpha \in S$ we require that $H(\alpha) = 0$.

Remark 5.5. The countermodel from Sect. 4 cannot be extended to a model of SF_1. To see this, note that we have $H(\{\{s\}\}) = 0$ and $\emptyset + \{\{s\}\} \cdot \dagger = \dagger$, but $\{\{s\}\}^* \cdot \emptyset \neq \dagger$—contradicting the unique fixpoint axiom.

The set of SF_1*-expressions* over Σ, denoted $\mathcal{T}_{\mathsf{SF}_1}$, is described by:

$$\mathcal{T}_{\mathsf{SF}_1} \ni e, f ::= 0 \mid 1 \mid a \in \mathcal{T}_{\mathsf{SL}} \mid e + f \mid e \cdot f \mid e \times f \mid e^* \mid H(e)$$

We interpret $\mathcal{T}_{\mathsf{SF}_1}$ in terms of languages via $[\![-]\!]_{\mathsf{SF}_1} : \mathcal{T}_{\mathsf{SF}_1} \to \mathcal{L}_\Sigma$, defined analogously to $[\![-]\!]_{\mathsf{SKA}}$, where furthermore for $e \in \mathcal{T}_{\mathsf{SF}_1}$ we have

$$[\![H(e)]\!]_{\mathsf{SF}_1} = [\![e]\!]_{\mathsf{SF}_1} \cap \{\varepsilon\}$$

Note that when $e \in \mathcal{T}_{\mathsf{SKA}}$, then $e \in \mathcal{T}_{\mathsf{SF}_1}$ and $[\![e]\!]_{\mathsf{SKA}} = [\![e]\!]_{\mathsf{SF}_1}$.

Define $\equiv_{\mathsf{SF}_1}$ as the smallest congruence on $\mathcal{T}_{\mathsf{SF}_1}$ induced by the axioms of SF_1, where $\mathcal{T}_{\mathsf{SL}}$ fulfills the role of the semilattice—e.g., if $a \in \mathcal{T}_{\mathsf{SL}}$, then $a \times a \equiv_{\mathsf{SF}_1} a$. This axiomatisation is sound with respect to the language model.[2]

Lemma 5.6. *Let $e, f \in \mathcal{T}_{\mathsf{SF}_1}$. If $e \equiv_{\mathsf{SF}_1} f$ then $[\![e]\!]_{\mathsf{SF}_1} = [\![f]\!]_{\mathsf{SF}_1}$.*

Remark 5.7. The caveat from Remark 5.3 can be transposed to this setting. However, the condition that for $\alpha \in S$ we have that $H(\alpha) = 0$ allows one to strengthen the above along the same lines as Remark 3.7, that is, if $e \equiv_{\mathsf{SF}_1} f$, then interpreting e and f in some SKA according to some valuation of Σ in terms of semilattice elements will produce the same outcome.

[2] Note that for the synchronous language model we know the least fixpoint axioms are sound as well (Lemma 3.6). However, there might be other SF_1-models where the least fixpoint axioms are not valid.

Remark 5.8. To demonstrate the use of the new axioms, we give an algebraic proof of $\alpha^* \times \alpha^* \equiv_{\mathsf{SF}_1} \alpha^*$ for $\alpha \in \mathcal{T}_{\mathsf{SL}}$:

$$\begin{aligned}\alpha^* \times \alpha^* &\equiv_{\mathsf{SF}_1} (1 + \alpha \cdot \alpha^*) \times (1 + \alpha \cdot \alpha^*) \equiv_{\mathsf{SF}_1} 1 + \alpha \cdot \alpha^* + (\alpha \cdot \alpha^*) \times (\alpha \cdot \alpha^*) \\ &\equiv_{\mathsf{SF}_1} 1 + \alpha \cdot \alpha^* + (\alpha \times \alpha) \cdot (\alpha^* \times \alpha^*) \equiv_{\mathsf{SF}_1} \alpha^* + \alpha \cdot (\alpha^* \times \alpha^*)\end{aligned}$$

Since $H(\alpha) = 0$, we can apply the unique fixpoint axiom to find $\alpha^* \cdot \alpha^* \equiv_{\mathsf{SF}_1} \alpha^* \times \alpha^*$. In SF_1, it is not hard to show that $\alpha^* \cdot \alpha^* \equiv_{\mathsf{F}_1} \alpha^*$; hence, we find $\alpha^* \times \alpha^* \equiv_{\mathsf{SF}_1} \alpha^*$.

Remark 5.9. Adding $\alpha^* \times \alpha^* = \alpha^*$ for $\alpha \in \mathcal{T}_{\mathsf{SL}}$ as an axiom to the old axiomatisation of SKA would not have been sufficient; one can easily find another semantical truth that does not hold in our countermodel, such as $[\![(\alpha \cdot \beta)^* \times (\alpha \cdot \beta)^*]\!]_{\mathsf{SKA}} = [\![(\alpha \cdot \beta)^*]\!]_{\mathsf{SKA}}$. Adding $e^* \times e^* = e^*$ as an axiom is also not an option, as this is not sound; for instance, take $e = a + b$ for $a, b \in \Sigma$. In order to keep the axiomatisation finitary, a unique fixpoint axiom provided an outcome.

5.1 Partial Derivatives

In this section we develop the theory of SKA and set up the necessary machinery for Sect. 6 and the completeness proof in Sect. 7. We start by presenting partial derivatives, which provide a termination and continuation map on $\mathcal{T}_{\mathsf{SF}_1}$. These derivatives allow us to turn the set of synchronous regular terms into a non-deterministic automaton structure, such that the language accepted by $e \in \mathcal{T}_{\mathsf{SF}_1}$ as a state in this automaton is the same as the semantics of e. Furthermore, partial derivatives turn out to provide a way to algebraically characterise a term by means of acceptance and reachable terms, which is useful in the completeness proof of SF_1.

The termination and continuation map for SF_1-expressions presented below are a trivial extension of the ones from [7]. Intuitively, the termination map is 1 if an expression can immediately terminate, and 0 otherwise; the continuation map of a term w.r.t. A gives us the set of terms reachable with an A-step.

Definition 5.10 (Termination map). *For $a \in \Sigma$, we define $o : \mathcal{T}_{\mathsf{SF}_1} \to 2$ inductively, as follows:*

$$\begin{array}{llll} o(0) = 0 & o(e^*) = 1 & o(e + f) = \max(o(e), o(f)) & o(e \times f) = \min(o(e), o(f)) \\ o(1) = 1 & o(a) = 0 & o(e \cdot f) = \min(o(e), o(f)) & o(H(e)) = o(e) \end{array}$$

Definition 5.11 (Continuation map). *For $a \in \Sigma$, we inductively define $\delta : \mathcal{T}_{\mathsf{SF}_1} \times \mathcal{P}_n(\Sigma) \to \mathcal{P}(\mathcal{T}_{\mathsf{SF}_1})$ as follows:*

$$\begin{aligned} \delta(0, A) &= \delta(1, A) = \emptyset & \delta(e \times f, A) &= \Delta(e, f, A) \cup \Delta(f, e, A) \\ \delta(H(e), A) &= \emptyset & &\quad \cup \{e' \times f' : e' \in \delta(e, B_1), \\ \delta(a, A) &= \{1 : A = \{a\}\} & &\quad f' \in \delta(f, B_2), B_1 \cup B_2 = A\} \\ \delta(e^*, A) &= \{e' \cdot e^* : e' \in \delta(e, A)\} & \delta(e \cdot f, A) &= \{e' \cdot f : e' \in \delta(e, A)\} \\ \delta(e + f, A) &= \delta(e, A) \cup \delta(f, A) & &\quad \cup \Delta(f, e, A) \end{aligned}$$

where $\Delta(e, f, A)$ is defined to be $\delta(e, A)$ when $o(f) = 1$, and $\emptyset$ otherwise.

Definition 5.12 (Syntactic Automaton). *We call the NDA* $(\mathcal{T}_{\mathsf{SF}_1}, o, \delta)$ *the* syntactic automaton *of* SF_1*-expressions.*

In Sect. 6 we give a proof of correctness of partial derivatives: for $e \in \mathcal{T}_{\mathsf{SF}_1}$ the semantics of e is equivalent to the language accepted by e as a state in the syntactic automaton. An analogous property holds for (partial) derivatives in Kleene algebras [1,8], which makes derivatives a powerful tool for reasoning about language models and deciding equivalences of terms [6].

In the next two sections, we want to use terms reachable from e, that is to say, terms that are a result of repeatedly applying the continuation map on e. To this end, we define the following function:

Definition 5.13. *For* $e, f \in \mathcal{T}_{\mathsf{SF}_1}$ *and* $a \in \Sigma$*, we inductively define the* reach *function* $\rho : \mathcal{T}_{\mathsf{SF}_1} \to \mathcal{P}(\mathcal{T}_{\mathsf{SF}_1})$ *as follows:*

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

Using a straightforward inductive argument, one can prove that for all $e \in \mathcal{T}_{\mathsf{SF}_1}$, $\rho(e)$ is finite. Note that e is not always a member of $\rho(e)$. To see that $\rho(e)$ indeed contains all terms reachable from e, we record the following.

Lemma 5.14. *For all* $e \in \mathcal{T}_{\mathsf{SF}_1}$ *and* $A \in \mathcal{P}_n(\Sigma)$*, we have* $\delta(e, A) \subseteq \rho(e)$*. Also, if* $e' \in \rho(e)$*, then* $\delta(e', A) \subseteq \rho(e)$*.*

5.2 Normal Form

In this section we develop a *normal form* for expressions in $\mathcal{T}_{\mathsf{SL}}$, which we will use in the completeness proof for SF_1. As $[\![-]\!]_{\mathsf{SL}}$ is a surjective function it has at least one right inverse. Let us pick one and denote it by $(-)^{\Pi}$. We thus have $(-)^{\Pi} : \mathcal{P}_n(\Sigma) \to \mathcal{T}_{\mathsf{SL}}$ such that $[\![-]\!]_{\mathsf{SL}} \circ (-)^{\Pi}$ is the identity on $\mathcal{P}_n(\Sigma)$.

The normal form for expressions in $\mathcal{T}_{\mathsf{SL}}$ is defined as follows:

Definition 5.15 (Normal form). *For all* $e \in \mathcal{T}_{\mathsf{SL}}$ *the* normal form *of* e*, denoted as* $\overline{e}$*, is defined as* $([\![e]\!]_{\mathsf{SL}})^{\Pi}$*. Let* $\overline{\mathcal{T}_{\mathsf{SL}}} = \{\overline{e} : e \in \mathcal{T}_{\mathsf{SL}}\}$*.*

Intuitively, an expression in normal form is standardised with respect to idempotence, associativity and commutativity. For instance, for a term $(a \times a) \times (c \times b)$ with $a, b, c \in \Sigma$, the chosen normal form, dictated by the chosen right inverse, could be $(a \times b) \times c$, and all terms provably equivalent to $(a \times a) \times (c \times b)$ will have this same normal form. Using Lemma 3.4, we can formalise this in the following two results:

Lemma 5.16. *For all* $e \in \mathcal{T}_{\mathsf{SL}}$*, we have that* e *is provably equivalent to its normal form:* $e \equiv_{\mathsf{SL}} \overline{e}$*. Moreover, if two expressions* $e, f \in \mathcal{T}_{\mathsf{SL}}$ *are provably equivalent, they have the same normal form: if* $e \equiv_{\mathsf{SL}} f$*, then* $\overline{e} = \overline{f}$*.*

Proof. As $(-)^{\Pi}$ is a right inverse of $[\![-]\!]_{\mathsf{SL}}$, we can derive the following:

$$[\![\overline{e}]\!]_{\mathsf{SL}} = [\![([\![e]\!]_{\mathsf{SL}})^{\Pi}]\!]_{\mathsf{SL}} = [\![e]\!]_{\mathsf{SL}}$$

From completeness we get $e \equiv_{\mathsf{SL}} \overline{e}$. For the second part of the statement we obtain via soundness that $[\![e]\!]_{\mathsf{SL}} = [\![f]\!]_{\mathsf{SL}}$ and subsequently that $\overline{e} = \overline{f}$. □

Normalising normalised terms does not change anything.

Lemma 5.17. *For all $e \in \overline{\mathcal{T}_{\mathsf{SL}}}$ we have that $\overline{e} = e$.*

We extend $(-)^{\Pi}$ from synchronous strings of length one to words and synchronous languages in the obvious way. For a synchronous string aw with $a \in \mathcal{P}_n(\Sigma)$ and $w \in (\mathcal{P}_n(\Sigma))^*$, and synchronous language $L \in \mathcal{L}_{\Sigma}$ we define:

$$\varepsilon^{\Pi} = \varepsilon \qquad (aw)^{\Pi} = a^{\Pi} \cdot (w^{\Pi}) \qquad L^{\Pi} = \{w^{\Pi} : w \in L\}$$

We abuse notation and assume the type of $(-)^{\Pi}$ is clear from the argument.

Since $(-)^{\Pi}$ is a homomorphism of languages, we have the following.

Lemma 5.18. *For synchronous languages L and K, all of the following hold: (i) $(L \cup K)^{\Pi} = L^{\Pi} \cup K^{\Pi}$, (i) $(L \cdot K)^{\Pi} = L^{\Pi} \cdot K^{\Pi}$, and (iii) $(L^*)^{\Pi} = (L^{\Pi})^*$.*

6 A Fundamental Theorem for SF_1

In this section we shall algebraically capture an expression in terms of its partial derivatives. This characterisation of an SF_1-term will be useful later on in proving completeness but also provides us with a straightforward method to prove correctness of the partial derivatives. Following [25,27], we call this characterisation a *fundamental theorem* for SF_1. Before we state and prove the fundamental theorem, we prove an intermediary lemma:

Lemma 6.1. *For all $e, f \in \mathcal{T}_{\mathsf{SF}_1}$, we have*

$$\sum_{e' \in \delta(e,A)} (A^{\Pi} \cdot e') \times \sum_{e' \in \delta(f,A)} (A^{\Pi} \cdot e') \equiv_{\mathsf{SF}_1} \sum_{\substack{e' \in \delta(e,A) \\ e'' \in \delta(f,B)}} (A \cup B)^{\Pi} \cdot (e' \times e'')$$

Proof. First note the following derivation for $A, B \in \mathcal{P}_n(\Sigma)$, using Lemma 5.16, the fact that all axioms of $\equiv_{\mathsf{SL}}$ are included in $\equiv_{\mathsf{SF}_1}$, and that $(-)^{\Pi}$ is a right inverse of $[\![-]\!]_{\mathsf{SL}}$:

$$\begin{aligned} A^{\Pi} \times B^{\Pi} &\equiv_{\mathsf{SF}_1} \overline{A^{\Pi} \times B^{\Pi}} = ([\![A^{\Pi} \times B^{\Pi}]\!]_{\mathsf{SL}})^{\Pi} \\ &= ([\![A^{\Pi}]\!]_{\mathsf{SL}} \cup [\![B^{\Pi}]\!]_{\mathsf{SL}})^{\Pi} = (A \cup B)^{\Pi} \end{aligned}$$

Using distributivity, the synchrony axiom and the equation above, we can derive:

$$\sum_{e' \in \delta(e,A)} (A^{\Pi} \cdot e') \times \sum_{e' \in \delta(f,A)} (A^{\Pi} \cdot e') \quad \equiv_{\mathsf{SF}_1} \sum_{\substack{e' \in \delta(e,A) \\ e'' \in \delta(f,B)}} (A^{\Pi} \cdot e') \times (B^{\Pi} \cdot e'')$$

$$\equiv_{\mathsf{SF}_1} \sum_{\substack{e' \in \delta(e,A) \\ e'' \in \delta(f,B)}} (A^{\Pi} \times B^{\Pi}) \cdot (e' \times e'') \quad \equiv_{\mathsf{SF}_1} \sum_{\substack{e' \in \delta(e,A) \\ e'' \in \delta(f,B)}} (A \cup B)^{\Pi} \cdot (e' \times e'')$$

The synchrony axiom can be applied because $A^{\Pi}, B^{\Pi} \in \mathcal{T}_{\mathsf{SL}}$. □

Theorem 6.2 (Fundamental Theorem). *For all $e \in \mathcal{T}_{\mathsf{SF}_1}$, we have*

$$e \equiv_{\mathsf{SF}_1} o(e) + \sum_{e' \in \delta(e,A)} A^{\Pi} \cdot e'.$$

Proof. This proof is mostly analogous to the proof of the fundamental theorem for regular expressions, such as the one that can be found in [27].

We proceed by induction on e. In the base, we have three cases to consider: $e \in \{0, 1\}$ or $e = a$ for $a \in \Sigma$. For $e \in \{0, 1\}$, the result follows immediately. For $e = a$, the only non-empty derivative is $\delta(a, \{a\})$ and the result follows:

$$o(a) + \sum_{e' \in \delta(a,A)} A^{\Pi} \cdot e' \equiv_{\mathsf{SF}_1} o(a) + \overline{a} \cdot 1 \equiv_{\mathsf{SF}_1} \overline{a} \equiv_{\mathsf{SF}_1} a$$

In the inductive step, we treat only the case for synchronous composition; the others can be found in the appendix. If $e = e_0 \times e_1$, derive as follows:

$$\begin{aligned}
&e_0 \times e_1 \\
&\equiv_{\mathsf{SF}_1} \Big(o(e_0) + \sum_{e' \in \delta(e_0,A)} A^{\Pi} \cdot e'\Big) \times \Big(o(e_1) + \sum_{e' \in \delta(e_1,A)} A^{\Pi} \cdot e'\Big) && \text{(Ind. hyp.)} \\
&\equiv_{\mathsf{SF}_1} o(e_0) \times o(e_1) + \sum_{e' \in \delta(e_0,A)} (A^{\Pi} \cdot e') \times o(e_1) + o(e_0) \times \sum_{e' \in \delta(e_1,A)} A^{\Pi} \cdot e' \\
&\qquad + \sum_{e' \in \delta(e_0,A)} (A^{\Pi} \cdot e') \times \sum_{e' \in \delta(e_1,A)} (A^{\Pi} \cdot e') && \text{(Distributivity)} \\
&\equiv_{\mathsf{SF}_1} o(e_0 \times e_1) + \sum_{e' \in \delta(e_0,A)} (A^{\Pi} \cdot e') \times o(e_1) + o(e_0) \times \sum_{e' \in \delta(e_1,A)} A^{\Pi} \cdot e' \\
&\qquad + \sum_{\substack{e' \in \delta(e_0,A) \\ e'' \in \delta(e_1,B)}} (A \cup B)^{\Pi} \cdot (e' \times e'') && \text{(Def. } o\text{, Lemma 6.1)} \\
&\equiv_{\mathsf{SF}_1} o(e_0 \times e_1) + \sum_{e' \in \Delta(e_0,e_1,A)} A^{\Pi} \cdot e' + \sum_{e' \in \Delta(e_1,e_0,A)} A^{\Pi} \cdot e' + \sum_{\substack{e' \in \{e'_0 \times e'_1 : e'_0 \in \delta(e_0,A), \\ e'_1 \in \delta(e_1,B), C = A \cup B\}}} C^{\Pi} \cdot e' \\
&\equiv_{\mathsf{SF}_1} o(e_0 \times e_1) + \sum_{e' \in \delta(e_0 \times e_1,A)} A^{\Pi} \cdot e' && \text{(Def. } \delta\text{)} \quad \square
\end{aligned}$$

Correctness of Partial Derivatives for SF_1
We now relate the partial derivatives for SF_1 to their semantics. Let $\ell : \mathcal{T}_{\mathsf{SF}_1} \to \mathcal{L}_\Sigma$ be the semantics of the syntactic automaton $(\mathcal{T}_{\mathsf{SF}_1}, o, \delta)$ (Definition 5.12), uniquely defined by Eq. 1:

$$\ell(e) = \{\varepsilon : o(e) = 1\} \cup \bigcup_{e' \in \delta(e,A)} \{A\} \cdot \ell(e') \tag{2}$$

To prove correctness of derivatives for SF_1, we prove that the language semantics of the syntactic automaton and the SF_1-expression coincide:

Theorem 6.3 (Soundness of derivatives). *For all $e \in \mathcal{T}_{\mathsf{SF}_1}$ we have:*

$$\ell(e) = [\![e]\!]_{\mathsf{SF}_1}$$

Proof The claim follows almost immediately from the fundamental theorem. From Lemma 3.6 and Theorem 6.2, we obtain

$$[\![e]\!]_{\mathsf{SF}_1} = \{\varepsilon : o(e) = 1\} \cup \bigcup_{e' \in \delta(e,A)} \{A\} \cdot [\![e']\!]_{\mathsf{SF}_1}$$

Note that $[\![A^\Pi]\!]_{\mathsf{SF}_1} = \{[\![A^\Pi]\!]_{\mathsf{SL}}\} = \{A\}$ by definition of the SF_1 semantics of a term in $\mathcal{T}_{\mathsf{SL}}$ and the fact that $(-)^\Pi$ is a right inverse. Because ℓ is the only function satisfying Eq. 2, we obtain the desired equality between $[\![e]\!]_{\mathsf{SF}_1}$ and the language $\ell(e)$ accepted by e as a state of the automaton $(\mathcal{T}_{\mathsf{SF}_1}, o, \delta)$. □

7 Completeness of SF_1

In this section we prove completeness of the SF_1-axioms with respect to the synchronous language model: we prove that for $e, f \in \mathcal{T}_{\mathsf{SF}_1}$, if $[\![e]\!]_{\mathsf{SF}_1} = [\![f]\!]_{\mathsf{SF}_1}$, then $e \equiv_{\mathsf{SF}_1} f$. We first prove completeness of SF_1 for a subset of SF_1-expressions, relying on the completeness result of F_1 (Lemma 7.3). Then we demonstrate that for every SF_1-expression we can find an equivalent SF_1-expression in this specific subset (Theorem 7.6). This subset is formed as follows.

Definition 7.1. *The set of SF_1-expressions in* normal form, $\mathcal{T}_{\mathsf{NSF}}$, *is described by the grammar*

$$\mathcal{T}_{\mathsf{NSF}} \ni e, f ::= 0 \mid 1 \mid a \in \overline{\mathcal{T}_{\mathsf{SL}}} \mid e + f \mid e \cdot f \mid e^*$$

where $\overline{\mathcal{T}_{\mathsf{SL}}}$ is as defined in Definition 5.15.

From this description it is immediate that an SF_1-term $e \in \mathcal{T}_{\mathsf{NSF}}$ is formed from terms of $\overline{\mathcal{T}_{\mathsf{SL}}}$ connected via the regular F_1-algebra operators. Hence, F_1-expressions formed over the alphabet $\overline{\mathcal{T}_{\mathsf{SL}}}$ are the same set of terms as $\mathcal{T}_{\mathsf{NSF}}$. We shall use this observation to prove completeness for $\mathcal{T}_{\mathsf{NSF}}$ with respect to the language model by leveraging completeness of F_1.

We use the function $(-)^\Pi$ to give a translation between the SF_1 semantics of a term in $\mathcal{T}_{\mathsf{NSF}}$ and the F_1 semantics of that same term:

Lemma 7.2. *For all $e \in \mathcal{T}_{\mathsf{NSF}}$, we have $(\llbracket e \rrbracket_{\mathsf{SF}_1})^{\Pi} = \llbracket e \rrbracket_{\mathsf{F}_1}$.*

Proof. We proceed by induction on the construction of e. In the base, there are three cases to consider. If $e = 0$, then $\llbracket e \rrbracket_{\mathsf{SF}_1} = \emptyset = \llbracket e \rrbracket_{\mathsf{F}_1}$, and we are done. If $e = 1$, then $(\llbracket e \rrbracket_{\mathsf{SF}_1})^{\Pi} = (\{\varepsilon\})^{\Pi} = \{\varepsilon\} = \llbracket 1 \rrbracket_{\mathsf{F}_1}$, and the claim follows. If $e = a$ for $a \in \overline{\mathcal{T}_{\mathsf{SL}}}$, we use Lemma 5.17 to obtain $\overline{a} = a$. As $a \in \overline{\mathcal{T}_{\mathsf{SL}}} \subseteq \mathcal{T}_{\mathsf{SL}}$, we know that $(\llbracket a \rrbracket_{\mathsf{SF}_1})^{\Pi} = (\{\llbracket a \rrbracket_{\mathsf{SL}}\})^{\Pi} = \{(\llbracket a \rrbracket_{\mathsf{SL}})^{\Pi}\} = \{\overline{a}\} = \{a\} = \llbracket a \rrbracket_{\mathsf{F}_1}$, and the claim follows.

For the inductive step, first consider $e = H(e_0)$. $(\llbracket H(e_0) \rrbracket_{\mathsf{SF}_1})^{\Pi} = \{\varepsilon\}$ if $\varepsilon \in \llbracket e_0 \rrbracket_{\mathsf{SF}_1}$ and $\emptyset$ otherwise. We also have $\llbracket H(e_0) \rrbracket_{\mathsf{F}_1} = \{\varepsilon\}$ if $\varepsilon \in \llbracket e_0 \rrbracket_{\mathsf{F}_1}$ and $\emptyset$ otherwise. The induction hypothesis states that $(\llbracket e_0 \rrbracket_{\mathsf{SF}_1})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{F}_1}$, from which we obtain that $\varepsilon \in \llbracket e_0 \rrbracket_{\mathsf{SF}_1} \Leftrightarrow \varepsilon \in \llbracket e_0 \rrbracket_{\mathsf{F}_1}$. Hence we can conclude that $(\llbracket H(e_0) \rrbracket_{\mathsf{SF}_1})^{\Pi} = \llbracket H(e_0) \rrbracket_{\mathsf{F}_1}$. All other inductive cases follow immediately from Lemma 5.18. The details can be found in the appendix. □

We are now ready to prove completeness of SF_1 for terms in normal form.

Lemma 7.3. *Let $e, f \in \mathcal{T}_{\mathsf{NSF}}$. If $\llbracket e \rrbracket_{\mathsf{SF}_1} = \llbracket f \rrbracket_{\mathsf{SF}_1}$, then $e \equiv_{\mathsf{SF}_1} f$.*

Proof. By the premise, we have that $(\llbracket e \rrbracket_{\mathsf{SF}_1})^{\Pi} = (\llbracket f \rrbracket_{\mathsf{SF}_1})^{\Pi}$. From Lemma 7.2 we get $(\llbracket e \rrbracket_{\mathsf{SF}_1})^{\Pi} = \llbracket e \rrbracket_{\mathsf{F}_1}$ and $(\llbracket f \rrbracket_{\mathsf{SF}_1})^{\Pi} = \llbracket f \rrbracket_{\mathsf{F}_1}$, which results in $\llbracket e \rrbracket_{\mathsf{F}_1} = \llbracket f \rrbracket_{\mathsf{F}_1}$. From Theorem 5.2 we know that this entails that $e \equiv_{\mathsf{F}_1} f$. As SF_1 contains all the axioms of F_1, we may then conclude that $e \equiv_{\mathsf{SF}_1} f$ and the claim follows. □

In order to prove completeness with respect to the language model for all $e \in \mathcal{T}_{\mathsf{SF}_1}$, we prove that for every $e \in \mathcal{T}_{\mathsf{SF}_1}$ there exists a term $\hat{e} \in \mathcal{T}_{\mathsf{NSF}}$ in normal form such that $e \equiv_{\mathsf{SF}_1} \hat{e}$. To see this is indeed enough to establish completeness of SF_1, imagine we have such a procedure to transform e into $\hat{e}$ in normal form. We can then conclude that $\llbracket e \rrbracket_{\mathsf{SF}_1} = \llbracket f \rrbracket_{\mathsf{SF}_1}$ implies $\llbracket \hat{e} \rrbracket_{\mathsf{SF}_1} = \llbracket \hat{f} \rrbracket_{\mathsf{SF}_1}$, which by Lemma 7.3 implies $\hat{e} \equiv_{\mathsf{SF}_1} \hat{f}$, and consequently that $e \equiv_{\mathsf{SF}_1} f$.

To obtain $\hat{e}$, we will make use of the "unfolding" of an SF_1-expression e in terms of partial derivatives, given by the fundamental theorem, which will give rise to a linear system. We will then show that this linear system has a unique solution that has the properties we require from $\hat{e}$. Since e is also a solution to this linear system, we can conclude that they are provably equivalent.

Let us start with the following property of linear systems over SF_1. A Q-vector is a function $x : Q \to \mathcal{T}_{\mathsf{SF}_1}$ and a Q-matrix is a function $M : Q \times Q \to \mathcal{T}_{\mathsf{SF}_1}$. We call a matrix M *guarded* if $H(M(i,j)) = 0$ for all $i, j \in Q$. We say a vector p and matrix M are in normal form if $p(i) \in \mathcal{T}_{\mathsf{NSF}}$ for all $i \in Q$ and $M(i,j) \in \mathcal{T}_{\mathsf{NSF}}$ for all $i, j \in Q$. The following lemma is a variation of [26, Lemma 2] and the proof is a direct adaptation of the proof found in [17, Lemma 3.12].

Lemma 7.4. *Let (M, p) be a Q-linear system such that M and p are guarded. We can construct Q-vector x that is the unique (up to SF_1-equivalence) solution to (M, p) in SF_1. Moreover, if M and p are in normal form, then so is x.*

We now define the linear system associated to an SF_1-expression e. This linear system makes use of the partial derivatives for SF_1, and essentially represents an NFA that accepts the language described by e.

Definition 7.5. *Let $e \in \mathcal{T}_{\mathsf{SF}_1}$, and choose $Q_e = \rho(e) \cup \{e\}$, where ρ is the reach function from Definition 5.13. Define the Q_e-vector x_e and the Q_e-matrix M_e by*

$$x_e(e') = o(e') \qquad\qquad M_e(e', e'') = \sum_{e'' \in \delta(e', A)} A^{\Pi}$$

We can now use Lemma 7.4 to obtain the desired normal form $\hat{e}$:

Theorem 7.6. *For all $e \in \mathcal{T}_{\mathsf{SF}_1}$, there exists an $\hat{e} \in \mathcal{T}_{\mathsf{NSF}}$ such that $\hat{e} \equiv_{\mathsf{SF}_1} e$.*

Proof. It is clear from their definition that x_e and M_e are both in normal form and that M_e is guarded. From Lemma 7.4 we then get that there exists a unique solution s_e to (M_e, x_e), and s_e is a Q_e-vector in normal form. Now consider the Q_e-vector y such that $y(q) = q$ for all $q \in Q_e$. Using Lemma 5.14 and Theorem 6.2, we can derive the following:

$$\begin{aligned} x_e(q) + M_e \cdot y(q) &\equiv_{\mathsf{SF}_1} x_e(q) + \sum_{q' \in Q_e} M_e(q, q') \cdot y(q') \\ &\equiv_{\mathsf{SF}_1} o(q) + \sum_{q' \in Q_e} \sum_{q' \in \delta(q, A)} A^{\Pi} \cdot q' \\ &\equiv_{\mathsf{SF}_1} o(q) + \sum_{q' \in \delta(q, A)} A^{\Pi} \cdot q' \equiv_{\mathsf{SF}_1} q = y(q) \end{aligned}$$

This demonstrates that y is also a solution to (M_e, x_e). As we know from Lemma 7.4 that s_e is unique, we get that $y \equiv_{\mathsf{SF}_1} s_e$. This means that $e = y(e) \equiv_{\mathsf{SF}_1} s_e(e)$. As s_e is in normal form we get that $s_e(e) \in \mathcal{T}_{\mathsf{NSF}}$. Thus, if we take $s_e(e) = \hat{e}$, then we have obtained the desired result. □

Combining Theorem 7.6 and Lemma 7.3 gives the main result of this section:

Theorem 7.7 (Soundness and Completeness). *For all $e, f \in \mathcal{T}_{\mathsf{SF}_1}$, we have*

$$e \equiv_{\mathsf{SF}_1} f \Leftrightarrow [\![e]\!]_{\mathsf{SF}_1} = [\![f]\!]_{\mathsf{SF}_1}$$

As a corollary of Theorem 6.3 and Theorem 7.7 we know that SF_1 is decidable by deciding language equivalence in the syntactic automaton.

8 Related Work

Synchonous cooperation among processes has been extensively studied in the context of process calculi such as ASP [4] and SCCS [23]. SKA bears a strong resemblance to SCCS, with the most notable differences being the equivalence axiomatised (bisimulation vs. language equivalence), and the use of Kleene star (unbounded finite recursion) instead of fixpoint (possibly infinite recursion). Contrary to ASP, but similar to SCCS, SKA cannot express incompatibility of action synchronisation.

In the context of Kleene algebra based frameworks for concurrent reasoning, a synchronous product is just one possible interpretation of concurrency. An interleaving-based approach with a concurrent operator (a parallel operator denoted with $\|$) is explored in Concurrent Kleene Algebra [14,15,17,22].

We have proved that $\equiv_{\mathsf{SF}_1}$ is sound and complete with respect to the synchronous language model by making use of the completeness of F_1 [26]. The strategy of transforming an expression e to an equivalent expression $\hat{e}$ with a particular property is often used in literature [16,17,20,22]. In particular, we adopted the use of linear systems as a representation of automata, which was first done by Conway [9] and Backhouse [2]. The machinery that we used to solve linear systems in F_1 is based on Salomaa [26] and can also be found in [17] and [19]. The idea of the syntactic automaton originally comes from Brzozowski, who did this for regular expressions [8]. He worked with derivatives which turn a Kleene algebra expression into a deterministic automaton. We worked with partial derivatives instead, resulting in a non-deterministic finite automaton for each SF_1-expression. Partial derivatives were first proposed by Antimirov [1].

Other related work is that of Hayes et al. [12]. They explore an algebra of synchronous atomic steps that interprets the synchrony model SKA is based on (Milner's SCCS calculus). However, their algebra is not based on a Kleene algebra—they use concurrent refinement algebra [11] instead. Later, Hayes et al. presented an abstract algebra for reasoning about concurrent programs with an abstract synchronisation operator [13], of which their earlier algebra of atomic steps is an instance. A key difference seems to be that Hayes et al. use different units for synchronous and sequential composition. It would be interesting to compare expressive powers of the two algebras more extensively.

A decision procedure for equivalence between SKA terms is given by Broda et al. [7]. They defined partial derivatives for SKA that we also used in the proof of completeness, and used those to construct an NFA that accepts the semantics of a given SKA expression. Deciding language equivalence of two automata then leads to a decision procedure for semantic equivalence of SKA expressions.

9 Conclusions and Further Work

We have presented a complete axiomatisation with respect to the model of synchronous regular languages. We have first proved incompleteness of SKA via a countermodel, exploiting the fact that SKA did not have any axioms relating the synchronous product to the Kleene star. We then provided a set of axioms based on the F_1-axioms from Salomaa [26] and the axioms governing the synchronous product familiar from SKA. This was shown to be a sound and complete axiomatisation with respect to the synchronous language model.

In the original SKA paper there is a presentation of *synchronous Kleene algebra with tests* including a wrongful claim of completeness. An obvious next step would be to see if we can prove completeness of SF_1 with tests. We conjecture SF_1 with tests is indeed complete and that this is straightforward to prove via a reduction to SF_1 in a style similar to the completeness proof of KAT [20].

Another generalisation is to add a unit to the semilattice, making it a bounded semilattice. This will probably lead to a type of delay operation [23].

Our original motivation to study SKA was to use it as an axiomatisation of Reo, a modular language of connectors combining synchronous data flow with an asynchronous one [3]. The semantics of Reo is based on an automata model very similar to that of SKAT, in which transitions are labelled by sets of ports (representing a synchronous data flow) and constraints (the tests of SKAT). Interestingly, automata are combined using an operation analogous to the synchronous product of SKAT expressions. We aim to study the application of SKA or SKAT to Reo in future work.

Acknowledgements. The first author is grateful for discussions with Hans-Dieter Hiep and Benjamin Lion.

A Appendix

Lemma 3.5. *The structure* $(\mathcal{L}_\Sigma, S, +, \cdot, ^*, \times, \emptyset, \{\varepsilon\})$ *is an SKA, that is, synchronous languages over* Σ *form an SKA.*

Proof. The carrier $\mathcal{L}_\Sigma$ is obviously closed under the operations of synchronous Kleene algebra. We need only argue that each of the SKA axioms is valid on synchronous languages.

The proof for the Kleene algebra axioms follows from the observation that synchronous languages over the alphabet Σ are simply languages over the alphabet $\mathcal{P}_n(\Sigma)$. Thus we know that the Kleene algebra axioms are satisfied, as languages over alphabet $\mathcal{P}_n(\Sigma)$ with $1 = \{\varepsilon\}$ and $0 = \emptyset$ form a Kleene algebra.

For the semilattice axioms, note that S is isomorphic to $\mathcal{P}_n(\Sigma)$ (by sending a singleton set in S to its sole element), and that the latter forms a semilattice when equipped with $\cup$. Since the isomorphism between S and $\mathcal{P}_n(\Sigma)$ respects these operators, it follows that $(S, \times)$ is also a semilattice.

The first SKA axiom that we check is commutativity. We prove that $\times$ on synchronous strings is commutative by induction on the paired length of the strings. Consider synchronous strings u and v. For the base, where u and v equal ε, the result is immediate. In the induction step, we take $u = xu'$ with $x \in \mathcal{P}_n(\Sigma)$. If $v = \varepsilon$ we are done immediately. Now for the case $v = yv'$ with $y \in \mathcal{P}_n(\Sigma)$. We have $u \times v = (xu') \times (yv') = (x \cup y) \cdot (u' \times v')$. From the induction hypothesis we know that $u' \times v' = v' \times u'$. Combining this with commutativity of union we have $u \times v = (x \cup y) \cdot (v' \times u') = v \times u$. Take synchronous languages K and L. Now consider $w \in K \times L$. This means that $w = u \times v$ for $u \in K$ and $v \in L$. From commutativity of synchronous strings we know that $w = u \times v = v \times u$. And thus we have $w \in L \times K$. The other inclusion is analogous.

It is obvious that the axioms $K \times \emptyset = \emptyset$ and $K \times \{\varepsilon\} = K$ are satisfied.

For associativity we again first argue that $\times$ on synchronous strings is associative. Take synchronous strings u, v and w. We will show by induction on the paired length of u, v and w that $u \times (v \times w) = (u \times v) \times w$. If $u, v, w = \varepsilon$ the result is immediate. Now consider $u = xu'$ for $x \in \mathcal{P}_n(\Sigma)$. If v or w equals

ε the result is again immediate. Hence we consider the case where $v = yv'$ and $w = zw'$ for $y, z \in \mathcal{P}_n(\Sigma)$. From the induction hypothesis we know that $u' \times (v' \times w') = (u' \times v') \times w'$. We can therefore derive

$$\begin{aligned} u \times (v \times w) &= (xu') \times (yv' \times zw') = (xu') \times ((y \cup z) \cdot (v' \times w')) \\ &= (x \cup (y \cup z)) \cdot (u' \times (v' \times w')) = (x \cup (y \cup z)) \cdot ((u' \times v') \times w') \end{aligned}$$

From associativity of union, we then know that $(x \cup (y \cup z)) \cdot ((u' \times v') \times w') = (u \times v) \times w$. Now consider $t \in K \times (L \times J)$ for K, L and J synchronous languages. Thus $t = u \times (v \times w)$ for $u \in K$, $v \in L$ and $w \in J$. From associativity of synchronous strings we know that $t = u \times (v \times w) = (u \times v) \times w$, and thus we have $t \in (K \times L) \times J$. The other inclusion is analogous.

For distributivity consider $w \in K \times (L + J)$ for K, L, J synchronous languages. This means that $w = u \times v$ for $u \in K$ and $v \in L + J$. Thus we know $v \in L$ or $v \in J$. We immediately get that $u \times v \in K \times L$ or $u \times v \in K \times J$ and therefore that $w \in K \times L + K \times J$. The other direction is analogous.

For the synchrony axiom we take synchronous languages K, L and $A, B \in S$. Suppose $A = \{x\}$ and $B = \{y\}$ for $x, y \in \mathcal{P}_n(\Sigma)$. Take $w \in (A \cdot K) \times (B \cdot L)$. This means that $w = u \times v$ for $u \in A \cdot K$ and $v \in B \cdot L$. Thus we know that $u = xu'$ with $u' \in K$ and $v = yv'$ with $v' \in L$. From this we conclude $w = u \times v = (xu') \times (yv') = (x \cup y) \cdot (u' \times v')$. As $u' \in K$ and $v' \in L$ and $x \cup y = x \times y$ with $x \in A$ and $y \in B$, we have that $w \in (A \times B) \cdot (K \times L)$. For the other direction, consider $w \in (A \times B) \cdot (K \times L)$. This entails $w = t \cdot v$ for $t \in A \times B$ and $v \in K \times L$. As $A \times B = \{x \cup y\}$ we have $t = x \cup y$. And $v = u \times s$ for $u \in K$ and $s \in L$. Thus $t \cdot v = (x \cup y) \cdot (u \times s) = (xu) \times (ys)$ for $u \in K$, $s \in L$, $x \in A$ and $y \in B$. Hence $w \in (A \cdot K) \times (B \cdot L)$. □

Lemma 3.6 (Soundness of SKA). *For all $e, f \in \mathcal{T}_{\mathsf{SKA}}$, we have that $e \equiv_{\mathsf{SKA}} f$ implies $[\![e]\!]_{\mathsf{SKA}} = [\![f]\!]_{\mathsf{SKA}}$.*

Proof. This is proved by induction on the construction of $\equiv_{\mathsf{SKA}}$. In the base case we need to check all the axioms generating $\equiv_{\mathsf{SKA}}$, which we have already done for Lemma 3.5. For the inductive step, we need to check the closure rules for congruence preserve soundness. This is all immediate from the definition of the semantics of SKA and the induction hypothesis. For instance, if $e = e_0 + e_1$, $f = f_0 + f_1$, $e_0 \equiv_{\mathsf{SKA}} f_0$ and $e_1 \equiv_{\mathsf{SKA}} f_1$, then $[\![e]\!]_{\mathsf{SKA}} = [\![e_0]\!]_{\mathsf{SKA}} + [\![e_1]\!]_{\mathsf{SKA}} = [\![f_0]\!]_{\mathsf{SKA}} + [\![f_1]\!]_{\mathsf{SKA}} = [\![f]\!]_{\mathsf{SKA}}$, where use that $[\![e_0]\!]_{\mathsf{SKA}} = [\![f_0]\!]_{\mathsf{SKA}}$ and $[\![e_1]\!]_{\mathsf{SKA}} = [\![f_1]\!]_{\mathsf{SKA}}$ as a consequence of the induction hypothesis.

Lemma 4.1. *For $\alpha \in \mathcal{T}_{\mathsf{SL}}$, we have $[\![\alpha^* \times \alpha^*]\!]_{\mathsf{SKA}} = [\![\alpha^*]\!]_{\mathsf{SKA}}$.*

Proof. For the first inclusion, take $w \in [\![a^* \times a^*]\!]_{\mathsf{SKA}} = [\![a^*]\!]_{\mathsf{SKA}} \times [\![a^*]\!]_{\mathsf{SKA}}$. Thus we have $w = u \times v$ for $u, v \in [\![a^*]\!]_{\mathsf{SKA}}$. Hence $u = x_1 \cdots x_n$ for $x_i \in [\![a]\!]_{\mathsf{SKA}}$ and $v = y_1 \cdots y_m$ for $y_i \in [\![a]\!]_{\mathsf{SKA}}$. As $[\![a]\!]_{\mathsf{SKA}} = \{[\![a]\!]_{\mathsf{SL}}\}$ with $[\![a]\!]_{\mathsf{SL}} \in \mathcal{P}_n(\Sigma)$, we know that $x_i = [\![a]\!]_{\mathsf{SL}}$ and $y_i = [\![a]\!]_{\mathsf{SL}}$. Assume that $n \leq m$ without loss of generality. We then know that $v = u \cdot [\![a]\!]_{\mathsf{SL}}^{m-n}$, where synchronous string e^n indicates n copies of string e concatenated. Unrolling the definition of $\times$ on

words, we find $u \times v = u \times (u \cdot [\![a]\!]^k_{\mathsf{SL}}) = (u \times u) \cdot [\![a]\!]^k_{\mathsf{SL}} = u \cdot [\![a]\!]^k_{\mathsf{SL}} = v$, and hence $w = u \times v = v \in [\![a^*]\!]_{\mathsf{SKA}}$. For the other inclusion, take $w \in [\![a^*]\!]_{\mathsf{SKA}}$. As $\varepsilon \in [\![a^*]\!]_{\mathsf{SKA}}$ and $w \times \varepsilon = w$, we immediately have $w \in [\![a^*]\!]_{\mathsf{SKA}} \times [\![a^*]\!]_{\mathsf{SKA}}$. □

Lemma A.1. *For $K, L \in \mathcal{L}_s$, K a non-empty finite language and L an infinite language, $K \times L$ is an infinite language.*

Proof. Suppose that $K \times L$ is a finite language. Hence we have an upper bound on the length of words in $K \times L$. Since the length of the synchronous product of two words is obviously the maximum of the length of the operands, this means we also have an upper bound on the length of words in L, and as we have finite words over a finite alphabet in L this means that L is finite. Hence we get a contradiction, thus $K \times L$ is infinite.

Lemma 4.3. *$\mathcal{M} = (\mathcal{L}_s \cup \{\dagger\}, \{\{\{s\}\}\}, +, \cdot, ^*, \times, \emptyset, \{\varepsilon\})$ with the operators as defined in Definition 4.2 forms an SKA.*

Proof. In the main text we treated one of the least fixpoint axioms and the synchrony axiom, and here we will treat all the remaining cases. For the sake of brevity, for each axiom we omit the cases where we can appeal to the proof for (synchronous) regular languages.

The proof that $(S, \times)$ is a semilattice is the same as in Lemma 3.5. Next, we take a look at the Kleene algebra axioms. If $K \in \mathcal{L}_s$, then $K + \emptyset = \emptyset$ holds by definition of union of sets. If $K = \dagger$, we get $\dagger + \emptyset = \dagger$, and the axiom also holds.

For $K \in \mathcal{L}_s \cup \{\dagger\}$, the axiom $K + K = K$ also easily holds by definition of the plus operator. Same for $K \cdot \{\varepsilon\} = K = K \cdot \{\varepsilon\}$ and $K \cdot \emptyset = \emptyset = \emptyset \cdot K$ by definition of the operator for sequential composition.

It is easy to see the axioms $1 + e \cdot e^* \equiv_{\mathsf{SKA}} e^*$ and $1 + e^* \cdot e \equiv_{\mathsf{SKA}} e^*$ hold for $K \in \mathcal{L}_s$. In case $K = \dagger$, for $1 + e \cdot e^* \equiv_{\mathsf{SKA}} e^*$ we have

$$1 + \dagger \cdot \dagger^* = 1 + \dagger \cdot \dagger = 1 + \dagger = \dagger = \dagger^*$$

and a similar derivation for $1 + e^* \cdot e \equiv_{\mathsf{SKA}} e^*$.

For the commutativity of $+$ we take $K, L \in \mathcal{L}_s \cup \{\dagger\}$. If $K = \dagger$ or $L = \dagger$, we have $K + L = \dagger = L + K$.

For associativity of the plus operator we take $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. If any of K, L or J is $\dagger$, it is easy to see the axiom holds.

For associativity of the sequential composition operator, consider $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. We first can observe that if one of K, L or J is empty, then the equality holds trivially. Otherwise, if one of K, L and J is $\dagger$, then $(K \cdot L) \cdot J = \dagger = K \cdot (L \cdot J)$.

Next, we verify distributivity of concatenation over $+$. We will show a detailed proof for left-distributivity only; right-distributivity can be proved similarly. Let $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. If one of K, L, or J is empty, then the claim holds immediately (the derivation is slightly different for K versus L or J). Otherwise, if one of K, L or J is $\dagger$, then $K \cdot (L + J) = \dagger = K \cdot L + K \cdot J$.

For the remaining least fixpoint axiom, let $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. Assume that $K + L \cdot J \leq L$. We need to prove that $K \cdot J^* \leq L$. If $L = \dagger$, then the claim holds immediately. If $L \in \mathcal{L}_s$ and $J = \dagger$, then L must be empty, hence K is empty, and the claim holds. If $L, J \in \mathcal{L}_s$, then also $K \in \mathcal{L}_s$ and the proof goes through as it does for KA.

We now get to the axioms for the $\times$-operator. The commutativity axiom is obvious from the commutative definition of $\times$ (as we already know that $\times$ is commutative on synchronous strings). The axiom $K \times \emptyset = \emptyset$ is also satisfied by definition. The same holds for the axiom $K \times \{\varepsilon\} = K$ as $\{\varepsilon\}$ is finite.

For associativity of the synchronous product, consider $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. If one of K, L or J is empty, then both sides of the equation evaluate to $\emptyset$. Otherwise, if one of K, J, or L is $\dagger$, then both sides of the equation evaluate to $\dagger$. If K, J and L are all languages, and at most one of them is finite, then either $K \times L = \dagger$, in which case the left-hand side evaluates to $\dagger$, or $K \times L$ is infinite (by Lemma A.1) and $J = \dagger$, in which case the right-hand side evaluates to $\dagger$ again. The right-hand side can be shown to evaluate to $\dagger$ by a similar argument. In the remaining cases (at least two out of K, J and L are finite languages and none of them is $\dagger$ or $\emptyset$), the proof of associativity for the language model applies.

For distributivity of synchronous product over $+$, let $K, L, J \in \mathcal{L}_s \cup \{\dagger\}$. If one of K, L or J is $\emptyset$, then the proof is straightforward. Otherwise, if one of K, L or J is $\dagger$, then both sides evaluate to $\dagger$. If K and $L + J$ are infinite, then the outcome is again $\dagger$ on both sides (note that $L + J$ being infinite implies that either L or J is infinite). In the remaining cases, K, L and J are languages and either K or $L + J$ (hence L and J) is finite. In either case the proof for synchronous regular languages goes through. □

Lemma 5.6. *Let $e, f \in \mathcal{T}_{\mathsf{SF}_1}$. If $e \equiv_{\mathsf{SF}_1} f$ then $[\![e]\!]_{\mathsf{SF}_1} = [\![f]\!]_{\mathsf{SF}_1}$.*

Proof. We need to verify each of the axioms of SF_1. The proof for the axioms of F_1 is immediate via the observation that synchronous languages over the alphabet Σ are simply languages over the alphabet $\mathcal{P}_n(\Sigma)$. Thus we know that the F_1-axioms are satisfied, as languages over alphabet $\mathcal{P}_n(\Sigma)$ with $1 = \{\varepsilon\}$ and $0 = \emptyset$ form an F_1-algebra. The additional axioms are the same as the ones that were added to KA for SKA, and we know they are sound from Lemma 3.6. □

Lemma 5.14. *For all $e \in \mathcal{T}_{\mathsf{SF}_1}$ and $A \in \mathcal{P}_n(\Sigma)$, we have $\delta(e, A) \subseteq \rho(e)$. Also, if $e' \in \rho(e)$, then $\delta(e', A) \subseteq \rho(e)$.*

Proof. We prove the first statement by induction on the structure of e. In the base, if we have $e \in \{0, 1\}$, the claim holds vacuously. If we have $a \in \Sigma$, then $\rho(a) = \{1, a\}$ and $\delta(a, A) = \{1 : A = \{a\}\}$, so the claim follows. For the inductive step, there are five cases to consider.

- If $e = H(e_0)$, then immediately $\delta(H(e_0), A) = \emptyset$ so the claim holds vacuously.
- If $e = e_0 + e_1$, then by induction we have $\delta(e_0, A) \subseteq \rho(e_0)$ and $\delta(e_1, A) \subseteq \rho(e_1)$. Hence, we find that $\delta(e, A) = \delta(e_0, A) \cup \delta(e_1, A) \subseteq \rho(e_0) \cup \rho(e_1) = \rho(e)$.

– If $e = e_0 \cdot e_1$, then by induction we have $\delta(e_0, A) \subseteq \rho(e_0)$ and $\delta(e_1, A) \subseteq \rho(e_1)$. Hence, we can calculate that

$$\begin{aligned}\delta(e, A) &= \{e_0' \cdot e_1 : e_0' \in \delta(e_0, A)\} \cup \Delta(e_1, e_0, A) \\ &\subseteq \{e_0' \cdot e_1 : e_0' \in \rho(e_0)\} \cup \rho(e_1) = \rho(e)\end{aligned}$$

– If $e = e_0 \times e_1$, then by induction we have $\delta(e_0, A) \subseteq \rho(e_0)$ and $\delta(e_1, A) \subseteq \rho(e_1)$ for all $A \in \mathcal{P}_n(\Sigma)$. Hence, we can calculate that

$$\begin{aligned}\delta(e, A) &= \{e_0' \times e_1' : e_0' \in \delta(e_0, B_1), e_1' \in \delta(e_1, B_2), B_1 \cup B_2 = A\} \\ &\quad \cup \Delta(e_0, e_1, A) \cup \Delta(e_1, e_0, A) \\ &\subseteq \{e_0' \times e_1' : e_0' \in \rho(e_0), e_1' \in \rho(e_1)\} \cup \rho(e_0) \cup \rho(e_1) = \rho(e)\end{aligned}$$

– If $e = e_0^*$, then by induction we have $\delta(e_0, A) \subseteq \rho(e_0)$. Hence, we find that

$$\delta(e, A) = \{e_0' \cdot e_0^* : e_0' \in \delta(e_0, A)\} \subseteq \{e_0' \cdot e_0^* : e_0' \in \rho(e_0)\} \subseteq \rho(e)$$

For the second statement, we prove that if $e' \in \rho(e)$, then $\rho(e') \subseteq \rho(e)$. The result of the first part tells us that $\delta(e', A) \subseteq \rho(e')$, which together with $\rho(e') \subseteq \rho(e)$ proves the claim. We proceed by induction on e. In the base, there are two cases to consider. First, if $e = 0$, then the claim holds vacuously. If $e = 1$, then the only $e' \in \rho(e)$ is $e' = 1$, so the claim holds. If $e = a$ for $a \in \Sigma$, we have $\rho(e) = \{1, a\}$. It trivially holds that $\rho(e') \subseteq \rho(e)$ for $e' \in \rho(e)$.

For the inductive step, there are four cases to consider.

– If $e = H(e_0)$, then $\rho(e) = \{1\}$, and the proof is as in the case where $e = 1$.
– If $e = e_0 + e_1$, assume w.l.o.g. that $e' \in \rho(e_0)$. By induction, we derive that

$$\rho(e') \subseteq \rho(e_0) \subseteq \rho(e)$$

– If $e = e_0 \cdot e_1$ then there are two cases to consider.
  • If $e' = e_0' \cdot e_1$ where $e_0' \in \rho(e_0)$, then we calculate

$$\begin{aligned}\rho(e') &= \{e_0'' \cdot e_1 : e_0'' \in \rho(e_0')\} \cup \rho(e_1) \\ &\subseteq \{e_0'' \cdot e_1 : e_0'' \in \rho(e_0)\} \cup \rho(e_1) = \rho(e)\end{aligned}$$

  • If $e' \in \rho(e_1)$, then by induction we have $\rho(e') \subseteq \rho(e_1) \subseteq \rho(e)$.
– If $e = e_0 \times e_1$ then there are three cases to consider.
  • The first case is $e' = e_0' \times e_1'$ where $e_0' \in \rho(e_0)$ and $e_1' \in \rho(e_1)$, we get $\rho(e_0') \subseteq \rho(e_0)$ and $\rho(e_1') \subseteq \rho(e_1)$ by induction. We calculate

$$\begin{aligned}\rho(e') &= \{e_0'' \times e_1'' : e_0'' \in \rho(e_0'), e_1'' \in \rho(e_1')\} \cup \rho(e_0') \cup \rho(e_1') \\ &\subseteq \{e_0'' \cdot e_1'' : e_0'' \in \rho(e_0), e_1'' \in \rho(e_1)\} \cup \rho(e_0) \cup \rho(e_1) \\ &= \rho(e)\end{aligned}$$

  • For $e' \in \rho(e_0)$, then by induction we have $\rho(e') \subseteq \rho(e_0) \subseteq \rho(e)$.
  • For $e' \in \rho(e_1)$, the argument is similar to the previous case.

– If $e = e_0^*$, then either $e' = 1$ or $e' = e_0' \cdot e_0^*$ for some $e_0' \in \rho(e_0)$. In the former case, $\rho(e') = \{1\} \subseteq \rho(e)$. In the latter case, we find by induction that

$$\begin{aligned}\rho(e') &= \{e_0'' \cdot e_0^* : e_0'' \in \rho(e_0')\} \cup \rho(e_0^*) \\ &\subseteq \{e_0'' \cdot e_0^* : e_0'' \in \rho(e_0)\} \cup \rho(e_0^*) \subseteq \rho(e_0^*)\end{aligned}$$

□

Lemma 5.17. *For all $e \in \overline{\mathcal{T}_{\mathsf{SL}}}$ we have that $\overline{e} = e$.*

Proof. As $e \in \overline{\mathcal{T}_{\mathsf{SL}}}$ we have that $e = \overline{e_0}$ for some $e_0 \in \mathcal{T}_{\mathsf{SL}}$. From Lemma 5.16 we know that $\overline{e_0} \equiv_{\mathsf{SF}_1} e_0$. So we get $e \equiv_{\mathsf{SF}_1} e_0$. Again from Lemma 5.16 we then know that $\overline{e} = \overline{e_0} = e$. □

Lemma A.2. *For $x, y \in (\mathcal{P}_n(\Sigma))^*$, we have $(x \cdot y)^{\Pi} = x^{\Pi} \cdot y^{\Pi}$.*

Proof. We proceed by induction on the length of xy. In the base, we have $xy = \varepsilon$. Thus $x = \varepsilon$ and $y = \varepsilon$. We have $\varepsilon^{\Pi} = \varepsilon$ so the result follows immediately. In the inductive step we consider $xy = aw$ for $a \in \mathcal{P}_n(\Sigma)$. We have to consider two cases. In the first case we have $x = ax'$. The induction hypothesis gives us that $(x' \cdot y)^{\Pi} = x'^{\Pi} \cdot y^{\Pi}$. We then have $(x \cdot y)^{\Pi} = (ax' \cdot y)^{\Pi} = a^{\Pi} \cdot (x' \cdot y)^{\Pi} = a^{\Pi} \cdot x'^{\Pi} \cdot y^{\Pi} = x^{\Pi} \cdot y^{\Pi}$. In the second case we have $x = \varepsilon$ and $y = aw$. We then conclude that $(x \cdot y)^{\Pi} = y^{\Pi} = x^{\Pi} \cdot y^{\Pi}$. □

Lemma 5.18. *For synchronous languages L and K, all of the following hold: (i) $(L \cup K)^{\Pi} = L^{\Pi} \cup K^{\Pi}$, (ii) $(L \cdot K)^{\Pi} = L^{\Pi} \cdot K^{\Pi}$, and (iii) $(L^*)^{\Pi} = (L^{\Pi})^*$.*

Proof. (i) First, suppose $w \in (L \cup K)^{\Pi}$. Thus we have $w = v^{\Pi}$ for $v \in L \cup K$. This gives us $v \in L$ or $v \in K$. We assume the former without loss of generality. Thus we know $w = v^{\Pi} \in L^{\Pi}$. Hence we know $w \in L^{\Pi} \cup K^{\Pi}$. The other direction can be proved analogously.

(ii) First, suppose $w \in (L \cdot K)^{\Pi}$. Thus we have $w = v^{\Pi}$ for some $v \in L \cdot K$. This gives us $v = v_1 \cdot v_2$ for some $v_1 \in L$ and some $v_2 \in K$. By definition of $(-)^{\Pi}$ we know that $v_1^{\Pi} \in L^{\Pi}$ and $v_2^{\Pi} \in K^{\Pi}$. Thus we have $v_1^{\Pi} \cdot v_2^{\Pi} \in L^{\Pi} \cdot K^{\Pi}$. From Lemma A.2 we know that $w = v^{\Pi} = (v_1 \cdot v_2)^{\Pi} = v_1^{\Pi} \cdot v_2^{\Pi}$, which gives us the desired result of $w \in L^{\Pi} \cdot K^{\Pi}$. The other direction can be proved analogously.

(iii) Take $w \in (L^*)^{\Pi}$. Thus we have $w = v^{\Pi}$ for some $v \in L^*$. By definition of the star of a synchronous language we know that $v = u_1 \cdots u_n$ for $u_i \in L$. As $u_i \in L$, we have $u_i^{\Pi} \in L^{\Pi}$ and $u_1^{\Pi} \cdots u_n^{\Pi} \in (L^{\Pi})^*$. By Lemma A.2, we know that $w = v^{\Pi} = (u_1 \cdots u_n)^{\Pi} = u_1^{\Pi} \cdots u_n^{\Pi}$. Thus we have $w \in (L^{\Pi})^*$, which is the desired result. The other direction can be proved analogously.

□

Theorem 6.2 (Fundamental Theorem). *For all $e \in \mathcal{T}_{\mathsf{SF}_1}$, we have*

$$e \equiv_{\mathsf{SF}_1} o(e) + \sum_{e' \in \delta(e,A)} A^{\Pi} \cdot e'.$$

Proof. Here we treat the inductive cases not displayed in the main proof, where we treated only the synchronous case.

– If $e = H(e_0)$, derive:

$$\begin{aligned}
H(e_0) &\equiv_{\mathsf{SF}_1} H(o(e_0)) + \sum_{e' \in \delta(e_0, A)} H(A^{\Pi}) \cdot H(e') && \text{(IH, compatibility of } H\text{)} \\
&\equiv_{\mathsf{SF}_1} H(o(e_0)) && (H(A^{\Pi}) = 0) \\
&\equiv_{\mathsf{SF}_1} o(H(e_0)) && (o(H(e_0)) \in 2) \\
&\equiv_{\mathsf{SF}_1} o(H(e_0)) + \sum_{e' \in \delta(H(e_0), A)} A^{\Pi} \cdot e' && \text{(Def. } \delta\text{)}
\end{aligned}$$

– If $e = e_0 + e_1$, derive:

$$\begin{aligned}
e_0 + e_1 &\equiv_{\mathsf{SF}_1} o(e_0) + \sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e' + o(e_1) + \sum_{e' \in \delta(e_1, A)} A^{\Pi} \cdot e' && \text{(IH)} \\
&\equiv_{\mathsf{SF}_1} o(e_0 + e_1) + \sum_{e' \in \delta(e_0, A) \cup \delta(e_1, A)} A^{\Pi} \cdot e' && \text{(Def. } o\text{, merge sums)} \\
&\equiv_{\mathsf{SF}_1} o(e_0 + e_1) + \sum_{e' \in \delta(e_0 + e_1, A)} A^{\Pi} \cdot e' && \text{(Def. } \delta\text{)}
\end{aligned}$$

– If $e = e_0 \cdot e_1$, derive:

$$\begin{aligned}
e_0 \cdot e_1 &\equiv_{\mathsf{SF}_1} \big(o(e_0) + \sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e'\big) \cdot e_1 && \text{(IH)} \\
&\equiv_{\mathsf{SF}_1} o(e_0) \cdot e_1 + \sum_{e' \in \delta(e_0, A)} (A^{\Pi} \cdot e' \cdot e_1) && \text{(Distributivity)} \\
&\equiv_{\mathsf{SF}_1} o(e_0) \cdot \big(o(e_1) + \sum_{e' \in \delta(e_1, A)} A^{\Pi} \cdot e'\big) + \sum_{e' \in \delta(e_0, A)} (A^{\Pi} \cdot e' \cdot e_1) && \text{(IH)} \\
&\equiv_{\mathsf{SF}_1} o(e_0 \cdot e_1) + o(e_0) \cdot \sum_{e' \in \delta(e_1, A)} A^{\Pi} \cdot e' + \sum_{e' \in \delta(e_0, A)} (A^{\Pi} \cdot e' \cdot e_1) \\
& && \text{(Def. } o\text{, distributivity)} \\
&\equiv_{\mathsf{SF}_1} o(e_0 \cdot e_1) + \sum_{e' \in \Delta(e_1, e_0, A)} A^{\Pi} \cdot e' + \sum_{e' \in \{e'_0 \cdot e_1 : e'_0 \in \delta(e_0, A)\}} A^{\Pi} \cdot e' \\
&\equiv_{\mathsf{SF}_1} o(e_0 \cdot e_1) + \sum_{e' \in \delta(e_0 \cdot e_1, A)} A^{\Pi} \cdot e' && \text{(Def. } \delta\text{)}
\end{aligned}$$

– If $e = e_0^*$, we derive:

$$
\begin{aligned}
e_0^* &\equiv_{\mathsf{SF}_1} \Big(o(e_0) + \sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e'\Big)^* && \text{(Induction hypothesis)} \\
&\equiv_{\mathsf{SF}_1} \Big(\sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e'\Big)^* && (o(e_0) \in 2 \text{ and loop tightening}) \\
&\equiv_{\mathsf{SF}_1} 1 + \Big(\sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e'\Big) \cdot \Big(\sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e'\Big)^* && \text{(star axiom of } \mathsf{SF}_1) \\
&\equiv_{\mathsf{SF}_1} 1 + \Big(\sum_{e' \in \delta(e_0, A)} A^{\Pi} \cdot e'\Big) \cdot e_0^* && \text{(first two steps)} \\
&\equiv_{\mathsf{SF}_1} 1 + \sum_{e' \in \delta(e_0, A)} (A^{\Pi} \cdot e' \cdot e_0^*) && \text{(Distributivity)} \\
&\equiv_{\mathsf{SF}_1} o(e_0^*) + \sum_{e' \in \delta(e_0^*, A)} A^{\Pi} \cdot e' && \text{(Def. } o\text{, def. } \delta) \ \square
\end{aligned}
$$

Lemma 7.2. *For all $e \in \mathcal{T}_{\mathsf{NSF}}$, we have $(\llbracket e \rrbracket_{\mathsf{SF}_1})^{\Pi} = \llbracket e \rrbracket_{\mathsf{F}_1}$.*

Proof. In the main text we have treated the base cases. The inductive cases work as follows. There are three cases to consider. If $e = e_0 + e_1$, then $(\llbracket e \rrbracket_{\mathsf{SKA}})^{\Pi} = (\llbracket e_0 \rrbracket_{\mathsf{SKA}} \cup \llbracket e_1 \rrbracket_{\mathsf{SKA}})^{\Pi} = (\llbracket e_0 \rrbracket_{\mathsf{SKA}})^{\Pi} \cup (\llbracket e_1 \rrbracket_{\mathsf{SKA}})^{\Pi}$ (Lemma 5.18). From the induction hypothesis we obtain $(\llbracket e_0 \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{KA}}$ and $(\llbracket e_1 \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_1 \rrbracket_{\mathsf{KA}}$. Combining these results we get $(\llbracket e \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{KA}} \cup \llbracket e_1 \rrbracket_{\mathsf{KA}} = \llbracket e_0 \rrbracket_{\mathsf{KA}} + \llbracket e_1 \rrbracket_{\mathsf{KA}} = \llbracket e_0 + e_1 \rrbracket_{\mathsf{KA}}$, so the claim follows. Secondly, if $e = e_0 \cdot e_1$, then $(\llbracket e \rrbracket_{\mathsf{SKA}})^{\Pi} = (\llbracket e_0 \rrbracket_{\mathsf{SKA}} \cdot \llbracket e_1 \rrbracket_{\mathsf{SKA}})^{\Pi} = (\llbracket e_0 \rrbracket_{\mathsf{SKA}})^{\Pi} \cdot (\llbracket e_1 \rrbracket_{\mathsf{SKA}})^{\Pi}$ (Lemma 5.18). From the induction hypothesis we obtain $(\llbracket e_0 \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{KA}}$ and $(\llbracket e_1 \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_1 \rrbracket_{\mathsf{KA}}$. We can then conclude that $(\llbracket e \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{KA}} \cdot \llbracket e_1 \rrbracket_{\mathsf{KA}} = \llbracket e_0 \cdot e_1 \rrbracket_{\mathsf{KA}}$. Lastly, if $e = e_0^*$, we get $(\llbracket e_0^* \rrbracket_{\mathsf{SKA}})^{\Pi} = ((\llbracket e_0 \rrbracket_{\mathsf{SKA}})^*)^{\Pi} = ((\llbracket e_0 \rrbracket_{\mathsf{SKA}})^{\Pi})^*$ (Lemma 5.18). From the induction hypothesis we obtain $(\llbracket e_0 \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{KA}}$. Thus we have $(\llbracket e \rrbracket_{\mathsf{SKA}})^{\Pi} = \llbracket e_0 \rrbracket_{\mathsf{KA}}^* = \llbracket e_0^* \rrbracket_{\mathsf{KA}}$ and the claim follows. $\square$

Lemma 7.4. *Let (M, p) be a Q-linear system such that M and p are guarded. We can construct Q-vector x that is the unique (up to SF_1-equivalence) solution to (M, p) in SF_1. Moreover, if M and p are in normal form, then so is x.*

Proof. We will construct x by induction on the size of Q. In the base, let $Q = \emptyset$. In this case the unique Q-vector is a solution. In the inductive step, take $k \in Q$ and let $Q' = Q \setminus \{k\}$. Then construct the Q'-linear system (M', p') as follows:

$$
\begin{aligned}
M'(i,j) &= M(i,k) \cdot M(k,k)^* \cdot M(k,j) + M(i,j) \\
p'(i) &= p(i) + M(i,k) \cdot M(k,k)^* \cdot p(k)
\end{aligned}
$$

As Q' is a strictly smaller set than Q and M' is guarded, we can apply our induction hypothesis to (M', p'). So we know by induction that (M', p') has a

unique solution x'. Moreover, if M' and p' are in normal form, so is x'; note that if M and p are in normal form, then so are M' and p'.

We use x' to construct the Q-vector x:

$$x(i) = \begin{cases} x'(i) & i \neq k \\ M(k,k)^* \cdot \big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j)\big) & i = k \end{cases}$$

The first thing to show now is that x is indeed a solution of (M, p). To this end, we need to show that $M \cdot x + p \equiv_{\mathsf{SF}_1} x$. We have two cases. For $i \in Q'$ we derive:

$$\begin{aligned}
x(i) &= x'(i) && \text{(Def. } x\text{)} \\
&\equiv_{\mathsf{SF}_1} p'(i) + \sum_{j \in Q'} M'(i,j) \cdot x'(j) && (x' \text{ solution of } (M', p')) \\
&\equiv_{\mathsf{SF}_1} p(i) + M(i,k) \cdot M(k,k)^* \cdot p(k) \\
&\quad + \sum_{j \in Q'} (M(i,k) \cdot M(k,k)^* \cdot M(k,j) + M(i,j)) \cdot x'(j) && \text{(Def. } (M', p')\text{)} \\
&\equiv_{\mathsf{SF}_1} p(i) + \sum_{j \in Q'} M(i,j) \cdot x'(j) \\
&\quad + M(i,k) \cdot M(k,k)^* \cdot \big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j)\big) && \text{(Distributivity)} \\
&\equiv_{\mathsf{SF}_1} p(i) + \sum_{j \in Q'} M(i,j) \cdot x(j) + M(i,k) \cdot x(k) && \text{(Def. } x\text{)} \\
&\equiv_{\mathsf{SF}_1} p(i) + \sum_{j \in Q} M(i,j) \cdot x(j) && \text{(Merge sum)}
\end{aligned}$$

For $i = k$, we derive:

$$\begin{aligned}
x(k) &= M(k,k)^* \cdot \big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j)\big) && \text{(Def. } x\text{)} \\
&\equiv_{\mathsf{SF}_1} (1 + M(k,k) \cdot M(k,k)^*) \cdot \big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j)\big) && \text{(star axiom)} \\
&\equiv_{\mathsf{SF}_1} p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j) \\
&\quad + M(k,k) \cdot M(k,k)^* \cdot \big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j)\big) && \text{(Distributivity)} \\
&\equiv_{\mathsf{SF}_1} p(k) + \sum_{j \in Q'} M(k,j) \cdot x(j) + M(k,k) \cdot x(k) && \text{(Def. } x\text{)} \\
&\equiv_{\mathsf{SF}_1} p(k) + \sum_{j \in Q} M(k,j) \cdot x(j) && \text{(Merge sum)}
\end{aligned}$$

We now know that x is a solution to (M, p) because $M \cdot x + p \equiv_{\mathsf{SF}_1} x$. Furthermore, if M and p are in normal form, then so is x', and thus x is in normal form by construction.

Next we claim that x is unique. Let y be any solution of (M, p). We choose the Q'-vector y' by taking $y'(i) = y(i)$. To see that y' is a solution to (M', p'), we first claim that the following holds:

$$y(k) \equiv_{\mathsf{SF}_1} M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot y(j)\Big) \tag{3}$$

To see that this is true, derive

$$\begin{aligned}
y(k) &\equiv_{\mathsf{SF}_1} p(k) + \sum_{j \in Q} M(k,j) \cdot y(j) && (y \text{ solution of } (M,p)) \\
&\equiv_{\mathsf{SF}_1} p(k) + M(k,k) \cdot y(k) + \sum_{j \in Q'} M(k,j) \cdot y(j) && (\text{Split sum}) \\
&\equiv_{\mathsf{SF}_1} M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot y(j)\Big) && (\text{Unique fixpoint axiom})
\end{aligned}$$

Note that we can apply the unique fixpoint axiom because we know that M is guarded and thus that $H(M(k,k)) = 0$.

Now we can derive the following:

$$\begin{aligned}
y'(i) &= y(i) && (\text{Def. } y) \\
&\equiv_{\mathsf{SF}_1} p(i) + \sum_{j \in Q} M(i,j) \cdot y(j) && (y \text{ solution of } (M,p)) \\
&\equiv_{\mathsf{SF}_1} p(i) + M(i,k) \cdot y(k) + \sum_{j \in Q'} M(i,j) \cdot y(j) && (\text{Split sum}) \\
&\equiv_{\mathsf{SF}_1} p(i) + \sum_{j \in Q'} M(i,j) \cdot y(j) \\
&\quad + M(i,k) \cdot M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot y(j)\Big) && (\text{Equation 3}) \\
&\equiv_{\mathsf{SF}_1} p(i) + M(i,k) \cdot M(k,k)^* \cdot p(k) \\
&\quad + \sum_{j \in Q'} \big(M(i,k) \cdot M(k,k)^* \cdot M(k,j) + M(i,j)\big) \cdot y(j) && (\text{Distributivity}) \\
&\equiv_{\mathsf{SF}_1} p'(i) + \sum_{j \in Q'} M'(i,j) \cdot y(j) && (\text{Def. } (M',p'))
\end{aligned}$$

Thus y' is a solution to (M', p'). As x' is the unique solution to (M', p'), we know that $y' \equiv_{\mathsf{SF}_1} x'$.

For $i \neq k$ we know that $x(i) = x'(i) \equiv_{\mathsf{SF}_1} y'(i) = y(i)$. For $i = k$ we can derive:

$$
\begin{aligned}
y(k) &\equiv_{\mathsf{SF}_1} M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot y(j)\Big) && \text{(Equation 3)} \\
&\equiv_{\mathsf{SF}_1} M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot y'(j)\Big) && \text{(Def. } y'\text{)} \\
&\equiv_{\mathsf{SF}_1} M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x'(j)\Big) && (x' \equiv_{\mathsf{SF}_1} y') \\
&\equiv_{\mathsf{SF}_1} M(k,k)^* \cdot \Big(p(k) + \sum_{j \in Q'} M(k,j) \cdot x(j)\Big) && \text{(Def. } x'\text{)} \\
&\equiv_{\mathsf{SF}_1} x(k) && \text{(Def. } x\text{)}
\end{aligned}
$$

Thus, $y \equiv_{\mathsf{SF}_1} x$, thereby proving that x is the unique solution to (M, p). □

References

1. Antimirov, V.M.: Partial derivatives of regular expressions and finite automaton constructions. Theor. Comput. Sci. **155**(2), 291–319 (1996). https://doi.org/10.1016/0304-3975(95)00182-4
2. Backhouse, R.: Closure algorithms and the star-height problem of regular languages. PhD thesis, University of London (1975)
3. Baier, C., Sirjani, M., Arbab, F., Rutten, J.: Modeling component connectors in reo by constraint automata. Sci. Comput. Program. **61**(2), 75–113 (2006). https://doi.org/10.1016/j.scico.2005.10.008
4. Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. Inf. Control **60**(1–3), 109–137 (1984). https://doi.org/10.1016/S0019-9958(84)80025-X
5. Boffa, M.: Une remarque sur les systèmes complets d'identités rationnelles. ITA **24**, 419–428 (1990)
6. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Proceedings of Principles of Programming Languages (POPL), pp. 457–468 (2013). https://doi.org/10.1145/2429069.2429124
7. Broda, S., Cavadas, S., Ferreira, M., Moreira, N.: Deciding synchronous Kleene algebra with derivatives. In: Drewes, F. (ed.) CIAA 2015. LNCS, vol. 9223, pp. 49–62. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22360-5_5
8. Brzozowski, J.A.: Derivatives of regular expressions. J. ACM **11**(4), 481–494 (1964). https://doi.org/10.1145/321239.321249
9. John Horton Conway: Regular Algebra and Finite Machines. Chapman and Hall Ltd., London (1971)
10. Foster, S., Struth, G.: On the fine-structure of regular algebra. J. Autom. Reason. **54**(2), 165–197 (2015). https://doi.org/10.1007/s10817-014-9318-9
11. Hayes, I.J.: Generalised rely-guarantee concurrency: an algebraic foundation. Formal Asp. Comput. **28**(6), 1057–1078 (2016). https://doi.org/10.1007/s00165-016-0384-0

12. Hayes, I.J., Colvin, R.J., Meinicke, L.A., Winter, K., Velykis, A.: An algebra of synchronous atomic steps. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016. LNCS, vol. 9995, pp. 352–369. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48989-6_22
13. Hayes, I.J., Meinicke, L.A., Winter, K., Colvin, R.J.: A synchronous program algebra: a basis for reasoning about shared-memory and event-based concurrency. Formal Asp. Comput. **31**(2), 133–163 (2019). https://doi.org/10.1007/s00165-018-0464-4
14. Kozen, D.: Myhill-Nerode relations on automatic systems and the completeness of Kleene algebra. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 27–38. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44693-1_3
15. Hoare, T., van Staden, S., Möller, B., Struth, G., Zhu, H.: Developments in concurrent Kleene algebra. J. Log. Algebr. Meth. Program. **85**(4), 617–636 (2016). https://doi.org/10.1016/j.jlamp.2015.09.012
16. Kappé, T., Brunet, P., Rot, J., Silva, A., Wagemaker, J., Zanasi, F.: Kleene algebra with observations. arXiv:1811.10401
17. Kappé, T., Brunet, P., Silva, A., Zanasi, F.: Concurrent Kleene algebra: free model and completeness. In: Proceedings of European Symposium on Programming (ESOP), pp. 856–882 (2018). https://doi.org/10.1007/978-3-319-89884-1_30
18. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Inf. Comput. **110**(2), 366–390 (1994). https://doi.org/10.1006/inco.1994.1037
19. Kozen, D.: Myhill-Nerode relations on automatic systems and the completeness of Kleene algebra. In: Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS), pp. 27–38 (2001). https://doi.org/10.1007/3-540-44693-1_3
20. Kozen, D., Smith, F.: Kleene algebra with tests: completeness and decidability. In: van Dalen, D., Bezem, M. (eds.) CSL 1996. LNCS, vol. 1258, pp. 244–259. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63172-0_43
21. Krob, D.: Complete systems of B-rational identities. Theor. Comput. Sci. **89**(2), 207–343 (1991). https://doi.org/10.1016/0304-3975(91)90395-I
22. Laurence, M.R., Struth, G.: Completeness theorems for pomset languages and concurrent Kleene algebras. arXiv:1705.05896
23. Milner, R.: Calculi for synchrony and asynchrony. Theor. Comput. Sci. **25**, 267–310 (1983). https://doi.org/10.1016/0304-3975(83)90114-7
24. Prisacariu, C.: Synchronous Kleene algebra. J. Log. Algebr. Program. **79**(7), 608–635 (2010). https://doi.org/10.1016/j.jlap.2010.07.009
25. Rutten, J.J.M.M.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theor. Comput. Sci. **308**(1–3), 1–53 (2003). https://doi.org/10.1016/S0304-3975(02)00895-2
26. Salomaa, A.: Two complete axiom systems for the algebra of regular events. J. ACM **13**(1), 158–169 (1966). https://doi.org/10.1145/321312.321326
27. Silva, A.: Kleene Coalgebra. PhD thesis, Radboud Universiteit Nijmegen (2010)