Concurrent Kleene Algebra: Free Model and Completeness

Tobias Kappé(✉), Paul Brunet, Alexandra Silva, and Fabio Zanasi

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

Abstract. Concurrent Kleene Algebra (CKA) was introduced by Hoare, Moeller, Struth and Wehrman in 2009 as a framework to reason about concurrent programs. We prove that the axioms for CKA with bounded parallelism are complete for the semantics proposed in the original paper; consequently, these semantics are the free model for this fragment. This result settles a conjecture of Hoare and collaborators. Moreover, the technique developed to this end allows us to establish a Kleene Theorem for CKA, extending an earlier Kleene Theorem for a fragment of CKA.

1 Introduction

Concurrent Kleene Algebra (CKA) [8] is a mathematical formalism which extends Kleene Algebra (KA) with a parallel composition operator, in order to express concurrent program behaviour.[1] In spite of such a seemingly simple addition, extending the existing KA toolkit (notably, completeness) to the setting of CKA turned out to be a challenging task. A lot of research happened since the original paper, both foundational [13,20] and on how CKA could be used to reason about important verification tasks in concurrent systems [9,11]. However, and despite several conjectures [9,13], the question of the characterisation of the free CKA and the completeness of the axioms remained open, making it impractical to use CKA in verification tasks. This paper settles these two open questions. We answer positively the conjecture that the free model of CKA is formed by series parallel pomset languages, downward-closed under Gischer's subsumption order [6]—a generalisation of regular languages to sets of partially ordered words. To this end, we prove that the original axioms proposed in [8] are indeed complete.

Our proof of completeness is based on extending an existing completeness result that establishes series-parallel rational pomset languages as the free Bi-Kleene Algebra (BKA) [20]. The extension to the existing result for BKA provides a clear understanding of the difficulties introduced by the presence of the exchange axiom and shows how to separate concerns between CKA and BKA, a technique which is also useful elsewhere. For one, our construction also provides

[1] In its original formulation, CKA also features an operator (*parallel star*) for unbounded parallelism: in harmony with several recent works [13,14], we study the variant of CKA without parallel star, sometimes called "weak" CKA.

A. Ahmed (Ed.): ESOP 2018, LNCS 10801, pp. 856–882, 2018.
https://doi.org/10.1007/978-3-319-89884-1_30

an extension of (half of) Kleene's theorem for BKA [14] to CKA, establishing pomset automata as an operational model for CKA and opening the door to decidability procedures similar to those previously studied for KA. Furthermore, it reduces deciding the equational theory of CKA to deciding the equational theory of BKA.

BKA is defined as CKA with the only (but significant) omission of the *exchange law*, $(e \parallel f) \cdot (g \parallel h) \leqq_{\mathsf{CKA}} (e \cdot g) \parallel (f \cdot h)$. The exchange law is the core element of CKA as it softens true concurrency: it states that when two sequentially composed programs (i.e., $e \cdot g$ and $f \cdot h$) are composed in parallel, they can be implemented by running their heads in parallel, followed by running their tails in parallel (i.e., $e \parallel f$, then $g \parallel h$). The exchange law allows the implementer of a CKA expression to interleave threads at will, without violating the specification.

To illustrate the use of the exchange law, consider a protocol with three actions: query a channel c, collect an answer from the same channel, and print an unrelated message m on screen. The specification for this protocol requires the query to happen before reception of the message, but the printing action being independent, it may be executed concurrently. We will write this specification as $(q(c) \cdot r(c)) \parallel p(m)$, with the operator $\cdot$ denoting sequential composition. However, if one wants to implement this protocol in a sequential programming language, a total ordering of these events has to be introduced. Suppose we choose to implement this protocol by printing m while we wait to receive an answer. This implementation can be written $q(c) \cdot p(m) \cdot r(c)$. Using the laws of CKA, we can prove that $q(c) \cdot p(m) \cdot r(c) \leqq_{\mathsf{CKA}} (q(c) \cdot r(c)) \parallel p(m)$, which we interpret as the fact that this implementation respects the specification. Intuitively, this means that the specification lists the necessary dependencies, but the implementation can introduce more.

Having a complete axiomatisation of CKA has two main benefits. First, it allows one to get certificates of correctness. Indeed, if one wants to use CKA for program verification, the decision procedure presented in [3] may be used to test program equivalence. If the test gives a negative answer, this algorithm provides a counter-example. However if the answer is positive, no meaningful witness is produced. With the completeness result presented here, that is constructive in nature, one could generate an axiomatic proof of equivalence in these cases. Second, it gives one a simple way of checking when the aforementioned procedure applies. By construction, we know that two terms are semantically equivalent whenever they are equal in every concurrent Kleene algebra, that is any model of the axioms of CKA. This means that if we consider a specific semantic domain, one simply needs to check that the axioms of CKA hold in there to know that the decision procedure of [3] is sound in this model.

While this paper was in writing, a manuscript with the same result appeared [19]. Among other things, the proof presented here is different in that it explicitly shows how to syntactically construct terms that express certain pomset languages, as opposed to showing that such terms must exist by reasoning on a semantic level. We refer to Sect. 5 for a more extensive comparison.

The remainder of this paper is organised as follows. In Sect. 2, we give an informal overview of the completeness proof. In Sect. 3, we introduce the necessary concepts, notation and lemmas. In Sect. 4, we work out the proof. We discuss the result in a broader perspective and outline further work in Sect. 5.

2 Overview of the Completeness Proof

We start with an overview of the steps necessary to arrive at the main result. As mentioned, our strategy in tackling CKA-completeness is to build on the existing BKA-completeness result. Following an observation by Laurence and Struth, we identify *downward-closure* (under Gischer's subsumption order [6]) as the feature that distinguishes the pomsets giving semantics to BKA-expressions from those associated with CKA-expressions. In a slogan,

$$\mathsf{CKA}\text{-semantics} = \mathsf{BKA}\text{-semantics} + \text{downward-closure}.$$

This situation is depicted in the upper part of the commuting diagram in Fig. 1. Intuitively, downward-closure can be thought of as the semantic outcome of adding the exchange axiom, which distinguishes CKA from BKA. Thus, if a and b are events that can happen in parallel according to the BKA-semantics of a term, then a and b may also be ordered in the CKA-semantics of that same term.

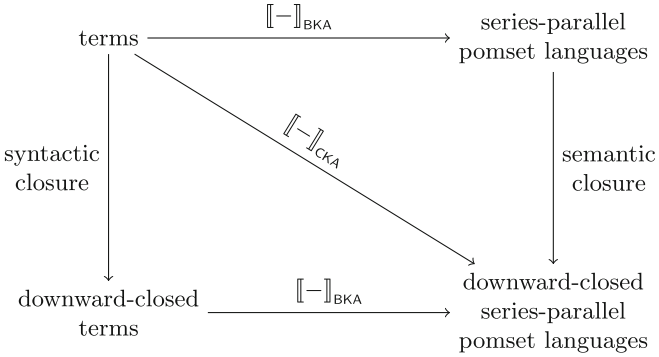

Fig. 1. The connection between BKA and CKA semantics mediated by closure.

The core of our CKA-completeness proof will be to construct a syntactic counterpart to the semantic closure. Concretely, we shall build a function that maps a CKA term e to an equivalent term $e{\downarrow}$, called the (syntactic) *closure* of e. The lower part of the commuting diagram in Fig. 1 shows the property that $e{\downarrow}$ must satisfy in order to deserve the name of closure: its BKA semantics has to be the same as the CKA semantics of e.

Example 2.1. Consider $e = a \parallel b$, whose CKA-semantics prescribe that a and b are events that may happen in parallel. One closure of this term would be $e{\downarrow} = a \parallel b + a \cdot b + b \cdot a$, whose BKA-semantics stipulate that either a and b execute purely in parallel, or a precedes b, or b precedes a—thus matching the optional parallelism of a and b. For a more non-trivial example, take $e = a^\star \parallel b^\star$, which represents that finitely many repetitions of a and b occur, possibly in parallel. A closure of this term would be $e{\downarrow} = (a^\star \parallel b^\star)^\star$: finitely many repetitions of a and b occur truly in parallel, which is repeated indefinitely.

In order to find $e{\downarrow}$ systematically, we are going to construct it in stages, through a completely syntactic procedure where each transformation has to be valid according to the axioms. There are three main stages.

(i) We note that, not unexpectedly, the hardest case for computing the closure of a term is when e is a parallel composition, i.e., when $e = e_0 \parallel e_1$ for some CKA terms e_0 and e_1. For the other operators, the closure of the result can be obtained by applying the same operator to the closures of its arguments. For instance, $(e + f){\downarrow} = e{\downarrow} + f{\downarrow}$. This means that we can focus on calculating the closure for the particular case of parallel composition.

(ii) We construct a *preclosure* of such terms e, whose BKA semantics contains all but possibly the sequentially composed pomsets of the CKA semantics of e. Since every sequentially composed pomset decomposes (uniquely) into non-sequential pomsets, we can use the preclosure as a basis for induction.

(iii) We extend this preclosure of e to a proper closure, by leveraging the fixpoint axioms of KA to solve a system of linear inequations. This system encodes "stringing together" non-sequential pomsets to build all pomsets in e.

As a straightforward consequence of the closure construction, we obtain a completeness theorem for CKA, which establishes the set of closed series-rational pomset languages as the free CKA.

3 Preliminaries

We fix a finite set of symbols Σ, the *alphabet*. We use the symbols a, b and c to denote elements of Σ. The two-element set $\{0, 1\}$ is denoted by 2. Given a set S, the set of subsets (*powerset*) of S is denoted by 2^S.

In the interest of readability, the proofs for technical lemmas in this section can be found in the full version [15].

3.1 Pomsets

A trace of a sequential program can be modelled as a word, where each letter represents an atomic event, and the order of the letters in the word represents the order in which the events took place. Analogously, a trace of a concurrent program can be thought of as word where letters are partially ordered, i.e., there need not be a causal link between events. In literature, such a partially ordered

word is commonly called a *partial word* [7], or *partially ordered multiset* (*pomset*, for short) [6]; we use the latter term.

A formal definition of pomsets requires some work, because the partial order should order *occurrences* of events rather than the events themselves. For this reason, we first define a labelled poset.

Definition 3.1. *A* labelled poset *is a tuple* $\langle S, \leq, \lambda \rangle$*, where* $\langle S, \leq \rangle$ *is a partially ordered set (i.e.,* S *is a set and* $\leq$ *is a partial order on* S*), in which* S *is called the* carrier *and* $\leq$ *is the* order*;* $\lambda : S \to \Sigma$ *is a function called the* labelling*.*

We denote labelled posets with lower-case bold symbols $\mathbf{u}$, $\mathbf{v}$, et cetera. Given a labelled poset $\mathbf{u}$, we write $S_{\mathbf{u}}$ for its carrier, $\leq_{\mathbf{u}}$ for its order and $\lambda_{\mathbf{u}}$ for its labelling. We write $\mathbf{1}$ for the empty labelled poset. We say that two labelled posets are *disjoint* if their carriers are disjoint.

Disjoint labelled posets can be composed parallelly and sequentially; parallel composition simply juxtaposes the events, while sequential composition imposes an ordering between occurrences of events originating from the left operand and those originating from the right operand.

Definition 3.2. *Let* $\mathbf{u}$ *and* $\mathbf{v}$ *be disjoint. We write* $\mathbf{u} \parallel \mathbf{v}$ *for the* parallel composition *of* $\mathbf{u}$ *and* $\mathbf{v}$*, which is the labelled poset with the carrier* $S_{\mathbf{u}\cup\mathbf{v}} = S_{\mathbf{u}} \cup S_{\mathbf{v}}$*, the order* $\leq_{\mathbf{u}\parallel\mathbf{v}} = \leq_{\mathbf{u}} \cup \leq_{\mathbf{v}}$ *and the labeling* $\lambda_{\mathbf{u}\parallel\mathbf{v}}$ *defined by*

$$\lambda_{\mathbf{u}\parallel\mathbf{v}}(x) = \begin{cases} \lambda_{\mathbf{u}}(x) & x \in S_{\mathbf{u}}; \\ \lambda_{\mathbf{v}}(x) & x \in S_{\mathbf{v}}. \end{cases}$$

Similarly, we write $\mathbf{u} \cdot \mathbf{v}$ *for the* sequential composition *of* $\mathbf{u}$ *and* $\mathbf{v}$*, that is, labelled poset with the carrier* $S_{\mathbf{u}\cup\mathbf{v}}$ *and the partial order*

$$\leq_{\mathbf{u}\cdot\mathbf{v}} = \leq_{\mathbf{u}} \cup \leq_{\mathbf{v}} \cup (S_{\mathbf{u}} \times S_{\mathbf{v}}),$$

as well as the labelling $\lambda_{\mathbf{u}\cdot\mathbf{v}} = \lambda_{\mathbf{u}\parallel\mathbf{v}}$*.*

Note that $\mathbf{1}$ is neutral for sequential and parallel composition, in the sense that we have $\mathbf{1} \parallel \mathbf{u} = \mathbf{1} \cdot \mathbf{u} = \mathbf{u} = \mathbf{u} \cdot \mathbf{1} = \mathbf{u} \parallel \mathbf{1}$.

There is a natural ordering between labelled posets with regard to concurrency.

Definition 3.3. *Let* $\mathbf{u}, \mathbf{v}$ *be labelled posets. A* subsumption *from* $\mathbf{u}$ *to* $\mathbf{v}$ *is a bijection* $h : S_{\mathbf{u}} \to S_{\mathbf{v}}$ *that preserves order and labels, i.e.,* $u \leq_{\mathbf{u}} u'$ *implies that* $h(u) \leq_{\mathbf{v}} h(u')$*, and* $\lambda_{\mathbf{v}} \circ h = \lambda_{\mathbf{u}}$*. We simplify and write* $h : \mathbf{u} \to \mathbf{v}$ *for a subsumption from* $\mathbf{u}$ *to* $\mathbf{v}$*. If such a subsumption exists, we write* $\mathbf{v} \sqsubseteq \mathbf{u}$*. Furthermore,* h *is an* isomorphism *if both* h *and its inverse* h^{-1} *are subsumptions. If there exists an isomorphism from* $\mathbf{u}$ *to* $\mathbf{v}$ *we write* $\mathbf{u} \cong \mathbf{v}$*.*

Intuitively, if $\mathbf{u} \sqsubseteq \mathbf{v}$, then $\mathbf{u}$ and $\mathbf{v}$ both order the same set of (occurrences of) events, but $\mathbf{u}$ has more causal links, or "is more sequential" than $\mathbf{v}$. One easily sees that $\sqsubseteq$ is a preorder on labelled posets of finite carrier.

Since the actual contents of the carrier of a labelled poset do not matter, we can abstract from them using isomorphism. This gives rise to pomsets.

Definition 3.4. *A* pomset *is an isomorphism class of labelled posets, i.e., the class* $[\mathbf{v}] \triangleq \{\mathbf{u} : \mathbf{u} \cong \mathbf{v}\}$ *for some labelled poset* $\mathbf{v}$*. Composition lifts to pomsets: we write* $[\mathbf{u}] \parallel [\mathbf{v}]$ *for* $[\mathbf{u} \parallel \mathbf{v}]$ *and* $[\mathbf{u}] \cdot [\mathbf{v}]$ *for* $[\mathbf{u} \cdot \mathbf{v}]$*. Similarly, subsumption also lifts to pomsets: we write* $[\mathbf{u}] \sqsubseteq [\mathbf{v}]$*, precisely when* $\mathbf{u} \sqsubseteq \mathbf{v}$*.*

We denote pomsets with upper-case symbols U, V, et cetera. The *empty pomset*, i.e., $[\mathbf{1}] = \{\mathbf{1}\}$, is denoted by 1; this pomset is neutral for sequential and parallel composition. To ensure that $[\mathbf{v}]$ is a set, we limit the discussion to labelled posets whose carrier is a subset of some set $\mathbb{S}$. The labelled posets in this paper have finite carrier; it thus suffices to choose $\mathbb{S} = \mathbb{N}$ to represent all pomsets with finite (or even countably infinite) carrier.

Composition of pomsets is well-defined: if $\mathbf{u}$ and $\mathbf{v}$ are not disjoint, we can find $\mathbf{u}', \mathbf{v}'$ disjoint from $\mathbf{u}, \mathbf{v}$ respectively such that $\mathbf{u} \cong \mathbf{u}'$ and $\mathbf{v} \cong \mathbf{v}'$. The choice of representative does not matter, for if $\mathbf{u} \cong \mathbf{u}'$ and $\mathbf{v} \cong \mathbf{v}'$, then $\mathbf{u} \cdot \mathbf{v} \cong \mathbf{u}' \cdot \mathbf{v}'$. Subsumption of pomsets is also well-defined: if $\mathbf{u}' \cong \mathbf{u} \sqsubseteq \mathbf{v} \cong \mathbf{v}'$, then $\mathbf{u}' \sqsubseteq \mathbf{v}'$. One easily sees that $\sqsubseteq$ is a partial order on finite pomsets, and that sequential and parallel composition are monotone with respect to $\sqsubseteq$, i.e., if $U \sqsubseteq W$ and $V \sqsubseteq X$, then $U \cdot V \sqsubseteq W \cdot X$ and $U \parallel V \sqsubseteq W \parallel X$. Lastly, we note that both types of composition are associative, both on the level of pomsets and labelled posets; we therefore omit parentheses when no ambiguity is likely.

Series-Parallel Pomsets. If $a \in \Sigma$, we can construct a labelled poset with a single element labelled by a; indeed, since any labelled poset thus constructed is isomorphic, we also use a to denote this isomorphism class; such a pomset is called a *primitive pomset*. A pomset built from primitive pomsets and sequential and parallel composition is called *series-parallel*; more formally:

Definition 3.5. *The set of* series-parallel *pomsets, denoted* $\mathsf{SP}(\Sigma)$*, is the smallest set such that* $1 \in \mathsf{SP}(\Sigma)$ *as well as* $a \in \mathsf{SP}(\Sigma)$ *for every* $a \in \Sigma$*, and is closed under parallel and sequential composition.*

We elide the sequential composition operator when we explicitly construct a pomset from primitive pomsets, i.e., we write ab instead of $a \cdot b$ for the pomset obtained by sequentially composing the (primitive) pomsets a and b. In this notation, sequential composition takes precedence over parallel composition.

All pomsets encountered in this paper are series-parallel. A useful feature of series-parallel pomsets is that we can deconstruct them in a standard fashion [6].

Lemma 3.1. *Let* $U \in \mathsf{SP}(\Sigma)$*. Then* exactly one *of the following is true: either (i)* $U = 1$*, or (ii)* $U = a$ *for some* $a \in \Sigma$*, or (iii)* $U = U_0 \cdot U_1$ *for* $U_0, U_1 \in \mathsf{SP}(\Sigma) \setminus \{1\}$*, or (iv)* $U = U_0 \parallel U_1$ *for* $U_0, U_1 \in \mathsf{SP}(\Sigma) \setminus \{1\}$*.*

In the sequel, it will be useful to refer to pomsets that are *not* of the third kind above, i.e., cannot be written as $U_0 \cdot U_1$ for $U_0, U_1 \in \mathsf{SP}(\Sigma) \setminus \{1\}$, as *non-sequential* pomsets. Lemma 3.1 gives a normal form for series-parallel pomsets, as follows.

Corollary 3.1. *A pomset* $U \in \mathsf{SP}(\Sigma)$ *can be uniquely decomposed as* $U = U_0 \cdot U_1 \cdots U_{n-1}$*, where for all* $0 \leq i < n$*,* U_i *is series parallel and non-sequential.*

Factorisation. We now go over some lemmas on pomsets that will allow us to factorise pomsets later on. First of all, one easily shows that subsumption is irrelevant on empty and primitive pomsets, as witnessed by the following lemma.

Lemma 3.2. *Let U and V be pomsets such that $U \sqsubseteq V$ or $V \sqsubseteq U$. If U is empty or primitive, then $U = V$.*

We can also consider how pomset composition and subsumption relate. It is not hard to see that if a pomset is subsumed by a sequentially composed pomset, then this sequential composition also appears in the subsumed pomset. A similar statement holds for pomsets that subsume a parallel composition.

Lemma 3.3 (Factorisation). *Let U, V_0, and V_1 be pomsets such that U is subsumed by $V_0 \cdot V_1$. Then there exist pomsets U_0 and U_1 such that:*

$$U = U_0 \cdot U_1,\ U_0 \sqsubseteq V_0,\ \text{and}\ U_1 \sqsubseteq V_1.$$

Also, if U_0, U_1 and V are pomsets such that $U_0 \parallel U_1 \sqsubseteq V$, then there exist pomsets V_0 and V_1 such that:

$$V = V_0 \parallel V_1,\ U_0 \sqsubseteq V_0,\ \text{and}\ U_1 \sqsubseteq V_1.$$

The next lemma can be thought of as a generalisation of Levi's lemma [21], a well-known statement about words, to pomsets. It says that if a sequential composition is subsumed by another (possibly longer) sequential composition, then there must be a pomset "in the middle", describing the overlap between the two; this pomset gives rise to a factorisation.

Lemma 3.4. *Let U and V be pomsets, and let $W_0, W_1, \ldots, W_{n-1}$ with $n > 0$ be non-empty pomsets such that $U \cdot V \sqsubseteq W_0 \cdot W_1 \cdots W_{n-1}$. There exists an $m < n$ and pomsets Y, Z such that:*

$$Y \cdot Z \sqsubseteq W_m,\ U \sqsubseteq W_0 \cdot W_1 \cdots W_{m-1} \cdot Y,\ \text{and}\ V \sqsubseteq Z \cdot W_{m+1} \cdot W_{m+2} \cdots W_n.$$

Moreover, if U and V are series-parallel, then so are Y and Z.

Levi's lemma also has an analogue for parallel composition.

Lemma 3.5. *Let U, V, W, X be pomsets such that $U \parallel V = W \parallel X$. There exist pomsets Y_0, Y_1, Z_0, Z_1 such that*

$$U = Y_0 \parallel Y_1,\ V = Z_0 \parallel Z_1,\ W = Y_0 \parallel Z_0,\ \text{and}\ X = Y_1 \parallel Z_1.$$

The final lemma is useful when we have a sequentially composed pomset subsumed by a parallelly composed pomset. It tells us that we can factor the involved pomsets to find subsumptions between smaller pomsets. This lemma first appeared in [6], where it is called the interpolation lemma.

Lemma 3.6 (Interpolation). *Let U, V, W, X be pomsets such that $U \cdot V$ is subsumed by $W \parallel X$. Then there exist pomsets W_0, W_1, X_0, X_1 such that*

$$W_0 \cdot W_1 \sqsubseteq W,\ X_0 \cdot X_1 \sqsubseteq X,\ U \sqsubseteq W_0 \parallel X_0,\ \text{and } V \sqsubseteq W_1 \parallel X_1.$$

Moreover, if W and X are series-parallel, then so are W_0, W_1, X_0 and X_1.

On a semi-formal level, the interpolation lemma can be understood as follows. If $U \cdot V \sqsubseteq W \parallel X$, then the events in W are partitioned between those that end up in U, and those that end up in V; these give rise to the "sub-pomsets" W_0 and W_1 of W, respectively. Similarly, X partitions into "sub-pomsets" X_0 and X_1. We refer to Fig. 2 for a graphical depiction of this situation.

Now, if y precedes z in $W_0 \parallel X_0$, then y must precede z in $W \parallel X$, and therefore also in $U \cdot V$. Since y and z are both events in U, it then follows that y precedes z in U, establishing that $U \sqsubseteq W_0 \parallel X_0$. Furthermore, if y precedes z in W, then we can exclude the case where y is in W_1 and z in W_0, for then z precedes y in $U \cdot V$, contradicting that y precedes z in $U \cdot V$. Accordingly, either y and z both belong to W_0 or W_1, or y is in W_0 while z is in W_1; in all of these cases, y must precede z in $W_0 \cdot W_1$. The other subsumptions hold analogously.

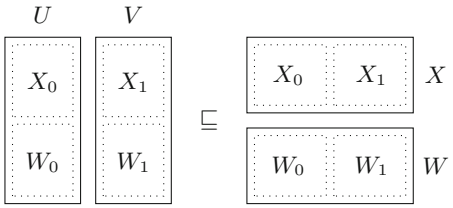

Fig. 2. Splitting pomsets in the interpolation lemma

Pomset Languages. The semantics of BKA and CKA are given in terms of sets of series-parallel pomsets.

Definition 3.6. *A subset of $\mathsf{SP}(\Sigma)$ is referred to as a* pomset language.

As a convention, we denote pomset languages by the symbols $\mathcal{U}$, $\mathcal{V}$, et cetera. Sequential and parallel composition of pomsets extends to pomset languages in a pointwise manner, i.e.,

$$\mathcal{U} \cdot \mathcal{V} \triangleq \{U \cdot V : U \in \mathcal{U}, V \in \mathcal{V}\}$$

and similarly for parallel composition. Like languages of words, pomset languages have a Kleene star operator, which is similarly defined, i.e., $\mathcal{U}^\star \triangleq \bigcup_{n \in \mathbb{N}} \mathcal{U}^n$, where the n^{th} power of $\mathcal{U}$ is inductively defined as $\mathcal{U}^0 \triangleq \{1\}$ and $\mathcal{U}^{n+1} \triangleq \mathcal{U}^n \cdot \mathcal{U}$.

A pomset language $\mathcal{U}$ is *closed under subsumption* (or simply *closed*) if whenever $U \in \mathcal{U}$ with $U' \sqsubseteq U$ and $U' \in \mathsf{SP}(\Sigma)$, it holds that $U' \in \mathcal{U}$. The *closure under subsumption* (or simply *closure*) of a pomset language $\mathcal{U}$, denoted $\mathcal{U}{\downarrow}$, is defined as the smallest pomset language that contains $\mathcal{U}$ and is closed, i.e.,

$$\mathcal{U}{\downarrow} \triangleq \{U' \in \mathsf{SP}(\Sigma) : \exists U \in \mathcal{U}.\ U' \sqsubseteq U\}$$

Closure relates to union, sequential composition and iteration as follows.

Lemma 3.7. *Let $\mathcal{U}, \mathcal{V}$ be pomset languages; then:*

$$(\mathcal{U} \cup \mathcal{V})\downarrow = \mathcal{U}\downarrow \cup \mathcal{V}\downarrow,\ (\mathcal{U} \cdot \mathcal{V})\downarrow = \mathcal{U}\downarrow \cdot \mathcal{V}\downarrow,\ \text{and } \mathcal{U}^\star\downarrow = \mathcal{U}\downarrow^\star.$$

Proof. The first claim holds for infinite unions, too, and follows immediately from the definition of closure.

For the second claim, suppose that $U \in \mathcal{U}$ and $V \in \mathcal{V}$, and that $W \sqsubseteq U \cdot V$. By Lemma 3.3, we find pomsets W_0 and W_1 such that $W = W_0 \cdot W_1$, with $W_0 \sqsubseteq U$ and $W_1 \sqsubseteq V$. It then holds that $W_0 \in \mathcal{U}\downarrow$ and $W_1 \in \mathcal{V}\downarrow$, meaning that $W = W_0 \cdot W_1 \in \mathcal{U}\downarrow \cdot \mathcal{V}\downarrow$. This shows that $(\mathcal{U} \cdot \mathcal{V})\downarrow \sqsubseteq \mathcal{U}\downarrow \cdot \mathcal{V}\downarrow$. Proving the reverse inclusion is a simple matter of unfolding the definitions.

For the third claim, we can calculate directly using the first and second parts of this lemma:

$$\mathcal{U}^\star\downarrow = \Big(\bigcup_{n\in\mathbb{N}} \underbrace{\mathcal{U}\cdot\mathcal{U}\cdots\mathcal{U}}_{n \text{ times}}\Big)\downarrow = \bigcup_{n\in\mathbb{N}} \Big(\underbrace{\mathcal{U}\cdot\mathcal{U}\cdots\mathcal{U}}_{n \text{ times}}\Big)\downarrow = \bigcup_{n\in\mathbb{N}} \underbrace{\mathcal{U}\downarrow\cdot\mathcal{U}\downarrow\cdots\mathcal{U}\downarrow}_{n \text{ times}} = \mathcal{U}\downarrow^\star \qquad \square$$

3.2 Concurrent Kleene Algebra

We now consider two extensions of Kleene Algebra (KA), known as *Bi-Kleene Algebra* (BKA) and *Concurrent Kleene Algebra* (CKA). Both extend KA with an operator for parallel composition and thus share a common syntax.

Definition 3.7. *The set $\mathcal{T}$ is the smallest set generated by the grammar*

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

The BKA-semantics of a term is a straightforward inductive application of the operators on the level of pomset languages. The CKA-semantics of a term is the BKA-semantics, downward-closed under the subsumption order; the CKA-semantics thus includes all possible sequentialisations.

Definition 3.8. *The function $[\![-]\!]_{\mathsf{BKA}} : \mathcal{T} \to 2^{\mathsf{SP}(\Sigma)}$ is defined as follows:*

$$[\![0]\!]_{\mathsf{BKA}} \triangleq \emptyset \qquad [\![e + f]\!]_{\mathsf{BKA}} \triangleq [\![e]\!]_{\mathsf{BKA}} \cup [\![f]\!]_{\mathsf{BKA}} \qquad [\![e^\star]\!]_{\mathsf{BKA}} \triangleq [\![e]\!]^\star_{\mathsf{BKA}}$$
$$[\![1]\!]_{\mathsf{BKA}} \triangleq \{1\} \qquad [\![e \cdot f]\!]_{\mathsf{BKA}} \triangleq [\![e]\!]_{\mathsf{BKA}} \cdot [\![f]\!]_{\mathsf{BKA}}$$
$$[\![a]\!]_{\mathsf{BKA}} \triangleq \{a\} \qquad [\![e \parallel f]\!]_{\mathsf{BKA}} \triangleq [\![e]\!]_{\mathsf{BKA}} \parallel [\![f]\!]_{\mathsf{BKA}}$$

Finally, $[\![-]\!]_{\mathsf{CKA}} : \mathcal{T} \to 2^{\mathsf{SP}(\Sigma)}$ is defined as $[\![e]\!]_{\mathsf{CKA}} \triangleq [\![e]\!]_{\mathsf{BKA}}\downarrow$.

Following Lodaya and Weil [22], if $\mathcal{U}$ is a pomset language such that $\mathcal{U} = [\![e]\!]_{\mathsf{BKA}}$ for some $e \in \mathcal{T}$, we say that the language $\mathcal{U}$ is *series-rational*. Note that if $\mathcal{U}$ is such that $\mathcal{U} = [\![e]\!]_{\mathsf{CKA}}$ for some term $e \in \mathcal{T}$, then $\mathcal{U}$ is closed by definition.

To axiomatise semantic equivalence between terms, we build the following relations, which match the axioms proposed in [20]. The axioms of CKA as defined in [8] come from a double quantale structure mediated by the exchange law; these imply the ones given here. The converse implication does not hold; in particular, our syntax does not include an infinitary greatest lower bound operator. However, BKA (as defined in this paper) does have a *finitary* greatest lower bound [20], and by the existence of closure, so does CKA.

Definition 3.9. *The relation $\equiv_{\mathsf{BKA}}$ is the smallest congruence on $\mathcal{T}$ (with respect to all operators) such that for all $e, f, g \in \mathcal{T}$:*

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

$$e \cdot 1 \equiv_{\mathsf{BKA}} e \qquad 1 \cdot e \equiv_{\mathsf{BKA}} e \qquad e \cdot (f \cdot g) \equiv_{\mathsf{BKA}} (e \cdot f) \cdot g$$

$$e \cdot 0 \equiv_{\mathsf{BKA}} 0 \equiv_{\mathsf{BKA}} 0 \cdot e \qquad e \cdot (f + g) \equiv_{\mathsf{BKA}} e \cdot f + e \cdot h \qquad (e + f) \cdot g \equiv_{\mathsf{BKA}} e \cdot g + f \cdot g$$

$$e \parallel f \equiv_{\mathsf{BKA}} f \parallel e \qquad e \parallel 1 \equiv_{\mathsf{BKA}} e \qquad e \parallel (f \parallel g) \equiv_{\mathsf{BKA}} (e \parallel f) \parallel g$$

$$e \parallel 0 \equiv_{\mathsf{BKA}} 0 \qquad e \parallel (f + g) \equiv_{\mathsf{BKA}} e \parallel f + e \parallel g \qquad 1 + e \cdot e^\star \equiv_{\mathsf{BKA}} e^\star$$

$$e + f \cdot g \leqq_{\mathsf{BKA}} g \implies f^\star \cdot e \leqq_{\mathsf{BKA}} g$$

in which we use $e \leqq_{\mathsf{BKA}} f$ as a shorthand for $e + f \equiv_{\mathsf{BKA}} f$. The final (conditional) axiom is referred to as the least fixpoint axiom.

The relation $\equiv_{\mathsf{CKA}}$ is the smallest congruence on $\mathcal{T}$ that satisfies the rules of $\equiv_{\mathsf{BKA}}$, and furthermore satisfies the exchange law *for all $e, f, g, h \in \mathcal{T}$:*

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

where we similarly use $e \leqq_{\mathsf{CKA}} f$ as a shorthand for $e + f \equiv_{\mathsf{CKA}} f$.

We can see that $\equiv_{\mathsf{BKA}}$ includes the familiar axioms of KA, and stipulates that $\parallel$ is commutative and associative with unit 1 and annihilator 0, as well as distributive over $+$. When using CKA to model concurrent program flow, the exchange law models sequentialisation: if we have two programs, the first of which executes e followed by g, and the second of which executes f followed by h, then we can sequentialise this by executing e and f in parallel, followed by executing g and h in parallel.

We use the symbol T in statements that are true for $\mathsf{T} \in \{\mathsf{BKA}, \mathsf{CKA}\}$. The relation $\equiv_{\mathsf{T}}$ is sound for equivalence of terms under T [13].

Lemma 3.8. *Let $e, f \in \mathcal{T}$. If $e \equiv_{\mathsf{T}} f$, then $[\![e]\!]_{\mathsf{T}} = [\![f]\!]_{\mathsf{T}}$.*

Since all binary operators are associative (up to $\equiv_{\mathsf{T}}$), we drop parentheses when writing terms like $e + f + g$—this does not incur ambiguity with regard to $[\![-]\!]_{\mathsf{T}}$. We furthermore consider $\cdot$ to have precedence over $\parallel$, which has precedence over $+$; as usual, the Kleene star has the highest precedence of all operators. For instance, when we write $e + f \cdot g^\star \parallel h$, this should be read as $e + ((f \cdot (g^\star)) \parallel h)$.

In case of BKA, the implication in Lemma 3.8 is an equivalence [20], and thus gives a complete axiomatisation of semantic BKA-equivalence of terms.[2]

Theorem 3.1. *Let $e, f \in \mathcal{T}$. Then $e \equiv_{\mathsf{BKA}} f$ if and only if $[\![e]\!]_{\mathsf{BKA}} = [\![f]\!]_{\mathsf{BKA}}$.*

Given a term $e \in \mathcal{T}$, we can determine syntactically whether its (BKA or CKA) semantics contains the empty pomset, using the function defined below.

[2] Strictly speaking, the proof in [20] includes the parallel star operator in BKA. Since this is a conservative extension of BKA, this proof applies to BKA as well.

Definition 3.10. *The* nullability function $\epsilon : \mathcal{T} \to 2$ *is defined as follows:*

$$\epsilon(0) \triangleq 0 \qquad \epsilon(e + f) \triangleq \epsilon(e) \vee \epsilon(f) \qquad \epsilon(e^\star) \triangleq 1$$
$$\epsilon(1) \triangleq 1 \qquad \epsilon(e \cdot f) \triangleq \epsilon(e) \wedge \epsilon(f)$$
$$\epsilon(a) \triangleq 0 \qquad \epsilon(e \parallel f) \triangleq \epsilon(e) \wedge \epsilon(f)$$

in which $\vee$ *and* $\wedge$ *are understood as the usual lattice operations on* 2.

That ϵ encodes the presence of 1 in the semantics is witnessed by the following.

Lemma 3.9. *Let* $e \in \mathcal{T}$*. Then* $\epsilon(e) \leqq_{\mathsf{T}} e$ *and* $1 \in [\![e]\!]_{\mathsf{T}}$ *if and only if* $\epsilon(e) = 1$*.*

In the sequel, we need the *(parallel) width* of a term. This is defined as follows.

Definition 3.11. *Let* $e \in \mathcal{T}$*. The* (parallel) width *of* e*, denoted by* $|e|$*, is defined as* 0 *when* $e \equiv_{\mathsf{BKA}} 0$*; for all other cases, it is defined inductively, as follows:*

$$|1| \triangleq 0 \qquad |e + f| \triangleq \max(|e|, |f|) \qquad |e \parallel f| \triangleq |e| + |f|$$
$$|a| \triangleq 1 \qquad |e \cdot f| \triangleq \max(|e|, |f|) \qquad |e^\star| \triangleq |e|$$

The width of a term is invariant with respect to equivalence of terms.

Lemma 3.10. *Let* $e, f \in \mathcal{T}$*. If* $e \equiv_{\mathsf{BKA}} f$*, then* $|e| = |f|$*.*

The width of a term is related to its semantics as demonstrated below.

Lemma 3.11. *Let* $e \in \mathcal{T}$*, and let* $U \in [\![e]\!]_{\mathsf{BKA}}$ *be such that* $U \neq 1$*. Then* $|e| > 0$*.*

3.3 Linear Systems

KA is equipped to find the least solutions to linear inequations. For instance, if we want to find X such that $e \cdot X + f \leqq_{\mathsf{KA}} X$, it is not hard to show that $e^\star \cdot f$ is the *least solution* for X, in the sense that this choice of X satisfies the inequation, and for any choice of X that also satisfies this inequation it holds that $e^\star \cdot f \leqq_{\mathsf{KA}} X$. Since KA is contained in BKA and CKA, the same constructions also apply there. These axioms generalise to systems of linear inequations in a straightforward manner; indeed, Kozen [18] exploited this generalisation to axiomatise KA. In this paper, we use systems of linear inequations to construct particular expressions. To do this, we introduce vectors and matrices of terms.

For the remainder of this section, we fix I as a finite set.

Definition 3.12. *An* I-vector *is a function from* I *to* $\mathcal{T}$*. Addition of* I*-vectors is defined pointwise, i.e., if* p *and* q *are* I*-vectors, then* $p + q$ *is the* I*-vector defined for* $i \in I$ *by* $(p + q)(i) \triangleq p(i) + q(i)$*.*

An I-matrix *is a function from* I^2 *to* $\mathcal{T}$*. Left-multiplication of an* I*-vector by an* I*-matrix is defined in the usual fashion, i.e., if* M *is an* I*-matrix and* p *is an* I*-vector, then* $M \cdot p$ *is the* I*-vector defined for* $i \in I$ *by*

$$(M \cdot p)(i) \triangleq \sum_{j \in I} M(i, j) \cdot p(j)$$

Equivalence between terms extends pointwise to I-vectors. More precisely, we write $p \equiv_{\mathsf{T}} q$ for I-vectors p and q when $p(i) \equiv_{\mathsf{T}} q(i)$ for all $i \in I$, and $p \leqq_{\mathsf{T}} q$ when $p + q \equiv_{\mathsf{T}} q$.

Definition 3.13. *An* I-linear system $\mathfrak{L}$ *is a pair* $\langle M, p\rangle$ *where* M *is an* I*-matrix and* p *is an* I*-vector. A* solution *to* $\mathfrak{L}$ *in* T *is an* I*-vector* s *such that* $M \cdot s + p \leqq_{\mathsf{T}} s$*. A* least solution *to* $\mathfrak{L}$ *in* T *is a solution* s *in* T *such that for any solution* t *in* T *it holds that* $s \leqq_{\mathsf{T}} t$*.*

It is not very hard to show that least solutions of a linear system are unique, up to $\equiv_{\mathsf{T}}$; we therefore speak of *the* least solution of a linear system.

Interestingly, *any* I-linear system has a least solution, and one can construct this solution using only the operators of KA. The construction proceeds by induction on $|I|$. In the base, where I is empty, the solution is trivial; for the inductive step it suffices to reduce the problem to finding the least solution of a strictly smaller linear system. This construction is not unlike Kleene's procedure to obtain a regular expression from a finite automaton [17]. Alternatively, we can regard the existence of least solutions as a special case of Kozen's proof of the fixpoint for matrices over a KA, as seen in [18, Lemma 9].

As a matter of fact, because this construction uses the axioms of KA exclusively, the least solution that is constructed is the same for both BKA and CKA.

Lemma 3.12. *Let* $\mathfrak{L}$ *be an* I*-linear system. One can construct a single* I*-vector* x *that is the least solution to* $\mathfrak{L}$ *in both* BKA *and* CKA*.*

We include a full proof of the lemma above using the notation of this paper in the full version of this paper [15].

4 Completeness of CKA

We now turn our attention to proving that $\equiv_{\mathsf{CKA}}$ is complete for CKA-semantic equivalence of terms, i.e., that if $e, f \in \mathcal{T}$ are such that $[\![e]\!]_{\mathsf{CKA}} = [\![f]\!]_{\mathsf{CKA}}$, then $e \equiv_{\mathsf{CKA}} f$. In the interest of readability, proofs of technical lemmas in this section can be found in the full version of this paper [15].

As mentioned before, our proof of completeness is based on the completeness result for BKA reproduced in Theorem 3.1. Recall that $[\![e]\!]_{\mathsf{CKA}} = [\![e]\!]_{\mathsf{BKA}}{\downarrow}$. To reuse completeness of BKA, we construct a syntactic variant of the closure operator, which is formalised below.

Definition 4.1. *Let* $e \in \mathcal{T}$*. We say that* $e{\downarrow}$ *is a* closure *of* e *if both* $e \equiv_{\mathsf{CKA}} e{\downarrow}$ *and* $[\![e{\downarrow}]\!]_{\mathsf{BKA}} = [\![e]\!]_{\mathsf{BKA}}{\downarrow}$ *hold.*

Example 4.1. Let $e = a \parallel b$; as proposed in Sect. 2, we claim that $e{\downarrow} = a \parallel b + b \cdot a + a \cdot b$ is a closure of e. To see why, first note that $e \leqq_{\mathsf{CKA}} e{\downarrow}$ by construction. Furthermore,

$$ab \equiv_{\mathsf{CKA}} (a \parallel 1) \cdot (1 \parallel b) \leqq_{\mathsf{CKA}} (a \cdot 1) \parallel (1 \cdot b) \equiv_{\mathsf{CKA}} a \parallel b$$

and similarly $ba \leqq_{\mathsf{CKA}} e$; thus, $e \equiv_{\mathsf{CKA}} e{\downarrow}$. Lastly, the pomsets in $[\![e]\!]_{\mathsf{BKA}}{\downarrow}$ and $[\![e{\downarrow}]\!]_{\mathsf{BKA}}$ are simply $a \parallel b$, ab and ba, and therefore $[\![e{\downarrow}]\!]_{\mathsf{BKA}} = [\![e]\!]_{\mathsf{BKA}}{\downarrow}$.

Laurence and Struth observed that the existence of a closure for every term implies a completeness theorem for CKA, as follows.

Lemma 4.1. *Suppose that we can construct a closure for every element of* $\mathcal{T}$. *If* $e, f \in \mathcal{T}$ *such that* $[\![e]\!]_{\mathsf{CKA}} = [\![f]\!]_{\mathsf{CKA}}$, *then* $e \equiv_{\mathsf{CKA}} f$.

Proof. Since $[\![e]\!]_{\mathsf{CKA}} = [\![e]\!]_{\mathsf{BKA}}\downarrow = [\![e\downarrow]\!]_{\mathsf{BKA}}$ and similarly $[\![f]\!]_{\mathsf{CKA}} = [\![f\downarrow]\!]_{\mathsf{BKA}}$, we have $[\![e\downarrow]\!]_{\mathsf{BKA}} = [\![f\downarrow]\!]_{\mathsf{BKA}}$. By Theorem 3.1, we get $e\downarrow \equiv_{\mathsf{BKA}} f\downarrow$, and thus $e\downarrow \equiv_{\mathsf{CKA}} f\downarrow$, since all axioms of BKA are also axioms of CKA. By $e \equiv_{\mathsf{CKA}} e\downarrow$ and $f\downarrow \equiv_{\mathsf{CKA}} f$, we can then conclude that $e \equiv_{\mathsf{CKA}} f$. □

The remainder of this section is dedicated to showing that the premise of Lemma 4.1 holds. We do this by explicitly constructing a closure $e\downarrow$ for every $e \in \mathcal{T}$. First, we note that closure can be constructed for the base terms.

Lemma 4.2. *Let* $e \in 2$ *or* $e = a$ *for some* $a \in \Sigma$. *Then* e *is a closure of itself.*

Furthermore, closure can be constructed compositionally for all operators except parallel composition, in the following sense.

Lemma 4.3. *Suppose that* $e_0, e_1 \in \mathcal{T}$, *and that* e_0 *and* e_1 *have closures* $e_0\downarrow$ *and* $e_1\downarrow$. *Then (i)* $e_0\downarrow + e_1\downarrow$ *is a closure of* $e_0 + e_1$, *(ii)* $e_0\downarrow \cdot e_1\downarrow$ *is a closure of* $e_0 \cdot e_1$, *and (iii)* $(e_0\downarrow)^\star$ *is a closure of* $e_0^\star$.

Proof. Since $e_0\downarrow \equiv_{\mathsf{CKA}} e_0$ and $e_1\downarrow \equiv_{\mathsf{CKA}} e_1$, by the fact that $\equiv_{\mathsf{CKA}}$ is a congruence we obtain $e_0\downarrow + e_1\downarrow \equiv_{\mathsf{CKA}} e_0 + e_1$. Similar observations hold for the other operators. We conclude using Lemma 3.7. □

It remains to consider the case where $e = e_0 \parallel e_1$. In doing so, our induction hypothesis is that any $f \in \mathcal{T}$ with $|f| < |e_0 \parallel e_1|$ has a closure, as well as any strict subterm of $e_0 \parallel e_1$.

4.1 Preclosure

To get to a closure of a parallel composition, we first need an operator on terms that is not a closure quite yet, but whose BKA-semantics is "closed enough" to cover the non-sequential elements of the CKA-semantics of the term.

Definition 4.2. *Let* $e \in \mathcal{T}$. *A* preclosure *of* e *is a term* $\tilde{e} \in \mathcal{T}$ *such that* $\tilde{e} \equiv_{\mathsf{CKA}} e$. *Moreover, if* $U \in [\![e]\!]_{\mathsf{CKA}}$ *is non-sequential, then* $U \in [\![\tilde{e}]\!]_{\mathsf{BKA}}$.

Example 4.2. Suppose that $e_0 \parallel e_1 = (a \parallel b) \parallel c$. A preclosure of $e_0 \parallel e_1$ could be

$$\tilde{e} = a \parallel b \parallel c + (a \cdot b + b \cdot a) \parallel c + (b \cdot c + c \cdot b) \parallel a + (a \cdot c + c \cdot a) \parallel b$$

To verify this, note that $e \leqq_{\mathsf{CKA}} \tilde{e}$ by construction; remains to show that $\tilde{e} \leqq_{\mathsf{CKA}} e$. This is fairly straightforward: since $a{\cdot}b{+}b{\cdot}a \leqq_{\mathsf{CKA}} a \parallel b$, we have $(a{\cdot}b{+}b{\cdot}a) \parallel c \leqq_{\mathsf{CKA}} e$; the other terms are treated similarly. Consequently, $e \equiv_{\mathsf{CKA}} \tilde{e}$. Furthermore, there are seven non-sequential pomsets in $[\![e]\!]_{\mathsf{CKA}}$; they are

$$a \parallel b \parallel c \qquad ab \parallel c \qquad ba \parallel c \qquad bc \parallel a \qquad cb \parallel a \qquad ac \parallel b \qquad ca \parallel b$$

Each of these pomsets is found in $[\![\tilde{e}]\!]_{\mathsf{BKA}}$. It should be noted that $\tilde{e}$ is *not* a closure of e; to see this, consider for instance that $abc \in [\![e]\!]_{\mathsf{CKA}}$, while $abc \notin [\![\tilde{e}]\!]_{\mathsf{BKA}}$.

The remainder of this section is dedicated to showing that, under the induction hypothesis, we can construct a preclosure for any parallelly composed term. This is not perfectly straightforward; for instance, consider the term $e_0 \parallel e_1$ discussed in Example 4.2. At first glance, one might be tempted to choose $e_0{\downarrow} \parallel e_1{\downarrow}$ as a preclosure, since $e_0{\downarrow}$ and $e_1{\downarrow}$ exist by the induction hypothesis. In that case, $e_0{\downarrow} = a \parallel b + a \cdot b + b \cdot a$ is a closure of e_0. Furthermore, $e_1{\downarrow} = c$ is a closure of e_1, by Lemma 4.2. However, $e_0{\downarrow} \parallel e_1{\downarrow}$ is not a preclosure of $e_0 \parallel e_1$, since $(a \cdot c) \parallel b$ is non-sequential and found in $[\![e_0 \parallel e_1]\!]_{\mathsf{CKA}}$, but not in $[\![e_0{\downarrow} \parallel e_1{\downarrow}]\!]_{\mathsf{BKA}}$.

The problem is that the preclosure of e_0 and e_1 should also allow (partial) sequentialisation of *parallel parts* of e_0 and e_1; in this case, we need to sequentialise the a part of $a \parallel b$ with c, and leave b untouched. To do so, we need to be able to *split* $e_0 \parallel e_1$ into pairs of constituent terms, each of which represents a possible way to divvy up its parallel parts. For instance, we can split $e_0 \parallel e_1 = (a \parallel b) \parallel c$ parallelly into $a \parallel b$ and c, but also into a and $b \parallel c$, or into $a \parallel c$ and b. The definition below formalises this procedure.

Definition 4.3. *Let $e \in \mathcal{T}$; Δ_e is the smallest relation on $\mathcal{T}$ such that*

$$\frac{}{1 \;\Delta_e\; e} \qquad \frac{}{e \;\Delta_e\; 1} \qquad \frac{\ell \;\Delta_{e_0}\; r}{\ell \;\Delta_{e_1+e_0}\; r} \qquad \frac{\ell \;\Delta_{e_1}\; r}{\ell \;\Delta_{e_0+e_1}\; r} \qquad \frac{\ell \;\Delta_e\; r}{\ell \;\Delta_{e^\star}\; r}$$

$$\frac{\ell \;\Delta_{e_0}\; r \qquad \epsilon(e_1) = 1}{\ell \;\Delta_{e_0 \cdot e_1}\; r} \qquad \frac{\ell \;\Delta_{e_1}\; r \qquad \epsilon(e_0) = 1}{\ell \;\Delta_{e_0 \cdot e_1}\; r} \qquad \frac{\ell_0 \;\Delta_{e_0}\; r_0 \qquad \ell_1 \;\Delta_{e_1}\; r_1}{\ell_0 \parallel \ell_1 \;\Delta_{e_0 \parallel e_1}\; r_0 \parallel r_1}$$

Given $e \in \mathcal{T}$, we refer to Δ_e as the *parallel splitting relation* of e, and to the elements of Δ_e as *parallel splices* of e. Before we can use Δ_e to construct the preclosure of e, we go over a number of properties of the parallel splitting relation. The first of these properties is that a given $e \in \mathcal{T}$ has only finitely many parallel splices. This will be useful later, when we involve *all* parallel splices of e in building a new term, i.e., to guarantee that the constructed term is finite.

Lemma 4.4. *For $e \in \mathcal{T}$, Δ_e is finite.*

We furthermore note that the parallel composition of any parallel splice of e is ordered below e by $\leqq_{\mathsf{BKA}}$. This guarantees that parallel splices never contain extra information, i.e., that their semantics do not contain pomsets that do not occur in the semantics of e. It also allows us to bound the width of the parallel splices by the width of the term being split, as a result of Lemma 3.10.

Lemma 4.5. *Let $e \in \mathcal{T}$. If $\ell \;\Delta_e\; r$, then $\ell \parallel r \leqq_{\mathsf{BKA}} e$.*

Corollary 4.1. *Let $e \in \mathcal{T}$. If $\ell \;\Delta_e\; r$, then $|\ell| + |r| \leq |e|$.*

Finally, we show that Δ_e is *dense* when it comes to parallel pomsets, meaning that if we have a parallelly composed pomset in the semantics of e, then we can find a parallel splice where one parallel component is contained in the semantics of one side of the pair, and the other component in that of the other.

Lemma 4.6. *Let $e \in \mathcal{T}$, and let V, W be pomsets such that $V \parallel W \in [\![e]\!]_{\mathsf{BKA}}$. Then there exist $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\Delta_e} r$ such that $V \in [\![\ell]\!]_{\mathsf{BKA}}$ and $W \in [\![r]\!]_{\mathsf{BKA}}$.*

Proof. The proof proceeds by induction on e. In the base, we can discount the case where $e = 0$, for then the claim holds vacuously. This leaves us two cases.

- If $e = 1$, then $V \parallel W \in [\![e]\!]_{\mathsf{BKA}}$ entails $V \parallel W = 1$. By Lemma 3.1, we find that $V = W = 1$. Since $1 \mathrel{\Delta_e} 1$ by definition of Δ_e, the claim follows when we choose $\ell = r = 1$.
- If $e = a$ for some $a \in \Sigma$, then $V \parallel W \in [\![e]\!]_{\mathsf{BKA}}$ entails $V \parallel W = a$. By Lemma 3.1, we find that either $V = 1$ and $W = a$, or $V = a$ and $W = 1$. In the former case, we can choose $\ell = 1$ and $r = a$, while in the latter case we can choose $\ell = a$ and $r = 1$. It is then easy to see that our claim holds in either case.

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

- If $e = e_0 + e_1$, then $U_0 \parallel U_1 \in [\![e_i]\!]_{\mathsf{BKA}}$ for some $i \in 2$. But then, by induction, we find $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\Delta_{e_i}} r$ such that $V \in [\![\ell]\!]_{\mathsf{BKA}}$ and $W \in [\![r]\!]_{\mathsf{BKA}}$. Since this implies that $\ell \mathrel{\Delta_e} r$, the claim follows.
- If $e = e_0 \cdot e_1$, then there exist pomsets U_0, U_1 such that $V \parallel W = U_0 \cdot U_1$, and $U_i \in [\![e_i]\!]_{\mathsf{BKA}}$ for all $i \in 2$. By Lemma 3.1, there are two cases to consider.
  - Suppose that $U_i = 1$ for some $i \in 2$, meaning that $V \parallel W = U_0 \cdot U_1 = U_{1-i} \in [\![e_{1-i}]\!]_{\mathsf{BKA}}$ for this i. By induction, we find $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\Delta_{e_{1-i}}} r$, and $V \in [\![\ell]\!]_{\mathsf{BKA}}$ as well as $W \in [\![r]\!]_{\mathsf{BKA}}$. Since $U_i = 1 \in [\![e_i]\!]_{\mathsf{BKA}}$, we have that $\epsilon(e_i) = 1$ by Lemma 3.9, and thus $\ell \mathrel{\Delta_e} r$.
  - Suppose that $V = 1$ or $W = 1$. In the former case, $V \parallel W = W = U_0 \cdot U_1 \in [\![e]\!]_{\mathsf{CKA}}$. We then choose $\ell = 1$ and $r = e$ to satisfy the claim. In the latter case, we can choose $\ell = e$ and $r = 1$ to satisfy the claim analogously.
- If $e = e_0 \parallel e_1$, then there exist pomsets U_0, U_1 such that $V \parallel W = U_0 \parallel U_1$, and $U_i \in [\![e_i]\!]_{\mathsf{BKA}}$ for all $i \in 2$. By Lemma 3.5, we find pomsets V_0, V_1, W_0, W_1 such that $V = V_0 \parallel V_1$, $W = W_0 \parallel W_1$, and $U_i = V_i \parallel W_i$ for $i \in 2$. For $i \in 2$, we then find by induction $\ell_i, r_i \in \mathcal{T}$ with $\ell_i \mathrel{\Delta_{e_i}} r_i$ such that $V_i \in [\![\ell_i]\!]_{\mathsf{BKA}}$ and $W_i \in [\![r_i]\!]_{\mathsf{BKA}}$. We then choose $\ell = \ell_0 \parallel \ell_1$ and $r = r_0 \parallel r_1$. Since $V = V_0 \parallel V_1$, it follows that $V \in [\![\ell]\!]_{\mathsf{BKA}}$, and similarly we find that $W \in [\![r]\!]_{\mathsf{BKA}}$. Since $\ell \mathrel{\Delta_e} r$, the claim follows.
- If $e = e_0^\star$, then there exist $U_0, U_1, \ldots, U_{n-1} \in [\![e_0]\!]_{\mathsf{BKA}}$ such that $V \parallel W = U_0 \cdot U_1 \cdots U_{n-1}$. If $n = 0$, i.e., $V \parallel W = 1$, then $V = W = 1$. In that case, we can choose $\ell = e$ and $r = 1$ to find that $\ell \mathrel{\Delta_e} r$, $V \in [\![\ell]\!]_{\mathsf{BKA}}$ and $W \in [\![r]\!]_{\mathsf{BKA}}$, satisfying the claim.
  If $n > 0$, we can assume without loss of generality that, for $0 \leq i < n$, it holds that $U_i \neq 1$. By Lemma 3.1, there are two subcases to consider.
  - Suppose that $V, W \neq 1$; then $n = 1$ (for otherwise $U_j = 1$ for some $0 \leq j < n$ by Lemma 3.1, which contradicts the above). Since $V \parallel W = U_0 \in [\![e_0]\!]_{\mathsf{BKA}}$, we find by induction $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\Delta_{e_0}} r$ such that $V \in [\![\ell]\!]_{\mathsf{BKA}}$ and $W \in [\![r]\!]_{\mathsf{BKA}}$. The claim then follows by the fact that $\ell \mathrel{\Delta_e} r$.

- Suppose that $V = 1$ or $W = 1$. In the former case, $V \parallel W = W = U_0 \cdot U_1 \cdots U_{n-1} \in [\![e]\!]_{\mathsf{CKA}}$. We then choose $\ell = 1$ and $r = e$ to satisfy the claim. In the latter case, we can choose $\ell = e$ and $r = 1$ to satisfy the claim analogously. □

Example 4.3. Let $U = a \parallel c$ and $V = b$, and note that $U \parallel V \in [\![e_0 \parallel e_1]\!]_{\mathsf{CKA}}$. We can then find that $a \; \Delta_a \; 1$ and $1 \; \Delta_b \; b$, and thus $a \parallel 1 \; \Delta_{e_0} \; 1 \parallel b$. Since also $c \; \Delta_c \; 1$, it follows that $(a \parallel 1) \parallel c \; \Delta_{e_0 \parallel e_1} \; (1 \parallel b) \parallel 1$. We can then choose $\ell = (a \parallel 1) \parallel c$ and $r = (1 \parallel b) \parallel 1$ to find that $U \in [\![\ell]\!]_{\mathsf{BKA}}$ and $V \in [\![r]\!]_{\mathsf{BKA}}$, while $\ell \; \Delta_{e_0 \parallel e_1} \; r$.

With parallel splitting in hand, we can define an operator on terms that combines all parallel splices of a parallel composition in a way that accounts for all of their downward closures.

Definition 4.4. *Let $e, f \in \mathcal{T}$, and suppose that, for every $g \in \mathcal{T}$ such that $|g| < |e| + |f|$, there exists a closure $g{\downarrow}$. The term $e \odot f$ is defined as follows:*

$$e \odot f \triangleq e \parallel f + \sum_{\substack{\ell \Delta_{e \parallel f} r \\ |\ell|, |r| < |e \parallel f|}} \ell{\downarrow} \parallel r{\downarrow}$$

Note that $e \odot f$ is well-defined: the sum is finite since $\Delta_{e \parallel f}$ is finite by Lemma 4.4, and furthermore $\ell{\downarrow}$ and $r{\downarrow}$ exist, as we required that $|\ell|, |r| < |e \parallel f|$.

Example 4.4. Let us compute $e_0 \odot e_1$ and verify that we obtain a preclosure of $e_0 \parallel e_1$. Working through the definition, we see that $\Delta_{e_0 \parallel e_1}$ consists of the pairs

$$\langle (1 \parallel 1) \parallel 1, (a \parallel b) \parallel c \rangle \qquad \langle (1 \parallel 1) \parallel c, (a \parallel b) \parallel 1 \rangle \qquad \langle (1 \parallel b) \parallel 1, (a \parallel 1) \parallel c \rangle$$

$$\langle (1 \parallel b) \parallel c, (a \parallel 1) \parallel 1 \rangle \qquad \langle (a \parallel 1) \parallel 1, (1 \parallel b) \parallel c \rangle \qquad \langle (a \parallel 1) \parallel c, (1 \parallel b) \parallel 1 \rangle$$

Since closure is invariant with respect to $\equiv_{\mathsf{CKA}}$, we can simplify these terms by applying the axioms of CKA. After folding the unit subterms, we are left with

$$\langle 1, a \parallel b \parallel c \rangle \qquad \langle c, a \parallel b \rangle \qquad \langle b, a \parallel c \rangle \qquad \langle b \parallel c, a \rangle \qquad \langle a, b \parallel c \rangle \qquad \langle a \parallel c, b \rangle$$

Recall that $a \parallel b + a \cdot b + b \cdot a$ is a closure of $a \parallel b$. Now, we find that

$$\begin{aligned} e_0 \odot e_1 = {} & (a \parallel b) \parallel c + c \parallel (a \parallel b + a \cdot b + b \cdot a) \\ & + b \parallel (a \parallel c + a \cdot c + c \cdot a) + (b \parallel c + b \cdot c + c \cdot b) \parallel a \\ & + a \parallel (b \parallel c + b \cdot c + c \cdot b) + (a \parallel c + a \cdot c + c \cdot a) \parallel b \\ \equiv_{\mathsf{CKA}} {} & a \parallel b \parallel c + a \parallel (b \cdot c + c \cdot b) + b \parallel (a \cdot c + c \cdot a) + c \parallel (a \cdot b + b \cdot a) \end{aligned}$$

which was shown to be a preclosure of $e_0 \parallel e_1$ in Example 4.2.

The general proof of correctness for $\odot$ as a preclosure plays out as follows.

Lemma 4.7. *Let $e, f \in \mathcal{T}$, and suppose that, for every $g \in \mathcal{T}$ with $|g| < |e|+|f|$, there exists a closure $g\downarrow$. Then $e \odot f$ is a preclosure of $e \parallel f$.*

Proof. We start by showing that $e \odot f \equiv_{\mathsf{CKA}} e \parallel f$. First, note that $e \parallel f \leqq_{\mathsf{BKA}} e \odot f$ by definition of $e \odot f$. For the other direction, suppose that $\ell, r \in \mathcal{T}$ are such that $\ell \; \Delta_{e \parallel f} \; r$. By definition of closure, we know that $\ell\downarrow \parallel r\downarrow \equiv_{\mathsf{CKA}} \ell \parallel r$. By Lemma 4.5, we have $\ell \parallel r \leqq_{\mathsf{BKA}} e \parallel f$. Since every subterm of $e \odot f$ is ordered below $e \parallel f$ by $\leqq_{\mathsf{CKA}}$, we have that $e \odot f \leqq_{\mathsf{CKA}} e \parallel f$. It then follows that $e \parallel f \equiv_{\mathsf{CKA}} e \odot f$.

For the second requirement, suppose that $X \in [\![e \parallel f]\!]_{\mathsf{CKA}}$ is non-sequential. We then know that there exists a $Y \in [\![e \parallel f]\!]_{\mathsf{BKA}}$ such that $X \sqsubseteq Y$. This leaves us two cases to consider.

- If X is empty or primitive, then $Y = X$ by Lemma 3.2, thus $X \in [\![e \parallel f]\!]_{\mathsf{BKA}}$. By the fact that $e \parallel f \leqq_{\mathsf{BKA}} e \odot f$ and by Lemma 3.8, we find $X \in [\![e \odot f]\!]_{\mathsf{BKA}}$.
- If $X = X_0 \parallel X_1$ for non-empty pomsets X_0 and X_1, then by Lemma 3.3 we find non-empty pomsets Y_0 and Y_1 with $Y = Y_0 \parallel Y_1$ such that $X_i \sqsubseteq Y_i$ for $i \in 2$. By Lemma 4.6, we find $\ell, r \in \mathcal{T}$ with $\ell \; \Delta_{e \parallel f} \; r$ such that $Y_0 \in [\![\ell]\!]_{\mathsf{BKA}}$ and $Y_1 \in [\![r]\!]_{\mathsf{BKA}}$. By Lemma 3.11, we find that $|\ell|, |r| \geq 1$. Corollary 4.1 then allows us to conclude that $|\ell|, |r| < |e \parallel f|$.
  This means that $\ell\downarrow \parallel r\downarrow \leqq_{\mathsf{BKA}} e \odot f$. Since $X_0 \in [\![\ell\downarrow]\!]_{\mathsf{BKA}}$ and $X_1 \in [\![r\downarrow]\!]_{\mathsf{BKA}}$ by definition of closure, we can derive by Lemma 3.8 that

$$X = X_0 \parallel X_1 \in [\![\ell\downarrow \parallel r\downarrow]\!]_{\mathsf{BKA}} \subseteq [\![e \odot f]\!]_{\mathsf{BKA}}$$

□

4.2 Closure

The preclosure operator discussed above covers the non-sequential pomsets in the language $[\![e \parallel f]\!]_{\mathsf{CKA}}$; it remains to find a term that covers the sequential pomsets contained in $[\![e \parallel f]\!]_{\mathsf{CKA}}$.

To better give some intuition to the construction ahead, we first explore the observations that can be made when a sequential pomset $W \cdot X$ appears in the language $[\![e \parallel f]\!]_{\mathsf{CKA}}$; without loss of generality, assume that W is non-sequential. In this setting, there must exist $U \in [\![e]\!]_{\mathsf{BKA}}$ and $V \in [\![f]\!]_{\mathsf{BKA}}$ such that $W \cdot X \sqsubseteq U \parallel V$. By Lemma 3.6, we find pomsets U_0, U_1, V_0, V_1 such that

$$W \sqsubseteq U_0 \parallel V_0 \qquad X \sqsubseteq U_1 \parallel V_1 \qquad U_0 \cdot U_1 \sqsubseteq U \qquad V_0 \cdot V_1 \sqsubseteq V$$

This means that $U_0 \cdot U_1 \in [\![e]\!]_{\mathsf{CKA}}$ and $V_0 \cdot V_1 \in [\![f]\!]_{\mathsf{CKA}}$. Now, suppose we could find $e_0, e_1, f_0, f_1 \in \mathcal{T}$ such that

$$e_0 \cdot e_1 \leqq_{\mathsf{CKA}} e \qquad U_0 \in [\![e_0]\!]_{\mathsf{CKA}} \qquad U_1 \in [\![e_1]\!]_{\mathsf{CKA}}$$

$$f_0 \cdot f_1 \leqq_{\mathsf{CKA}} f \qquad V_0 \in [\![f_0]\!]_{\mathsf{CKA}} \qquad V_1 \in [\![f_1]\!]_{\mathsf{CKA}}$$

Then we have $W \in [\![e_0 \odot f_0]\!]_{\mathsf{BKA}}$, and $X \in [\![e_1 \parallel f_1]\!]_{\mathsf{CKA}}$. Thus, if we can find a closure of $e_1 \parallel f_1$, then we have a term whose BKA-semantics contains $W \cdot X$.

There are two obstacles that need to be resolved before we can use the observations above to find the closure of $e \parallel f$. The first problem is that we need to be sure that this process of splitting terms into sequential components is at all possible, i.e., that we can split e into e_0 and e_1 with $e_0 \cdot e_1 \leqq_{\mathsf{CKA}} e$ and $U_i \in [\![e_i]\!]_{\mathsf{CKA}}$ for $i \in 2$. We do this by designing a sequential analogue to the parallel splitting relation seen before. The second problem, which we will address later in this section, is whether this process of splitting a parallel term $e \parallel f$ according to the exchange law and finding a closure of remaining term $e_1 \parallel f_1$ is well-founded, i.e., if we can find "enough" of these terms to cover all possible ways of sequentialising $e \parallel f$. This will turn out to be possible, by using the fixpoint axioms of KA as in Sect. 3.3 with linear systems.

We start by defining the sequential splitting relation.[3]

Definition 4.5. *Let $e \in \mathcal{T}$; ∇_e is the smallest relation on $\mathcal{T}$ such that*

$$\frac{}{1 \mathrel{\nabla_1} 1} \qquad \frac{}{a \mathrel{\nabla_a} 1} \qquad \frac{}{1 \mathrel{\nabla_a} a} \qquad \frac{}{1 \mathrel{\nabla_{e_0^\star}} 1} \qquad \frac{\ell \mathrel{\nabla_{e_0}} r}{\ell \mathrel{\nabla_{e_0+e_1}} r} \qquad \frac{\ell \mathrel{\nabla_{e_1}} r}{\ell \mathrel{\nabla_{e_0+e_1}} r}$$

$$\frac{\ell \mathrel{\nabla_{e_0}} r}{\ell \mathrel{\nabla_{e_0 \cdot e_1}} r \cdot e_1} \qquad \frac{\ell \mathrel{\nabla_{e_1}} r}{e_0 \cdot \ell \mathrel{\nabla_{e_0 \cdot e_1}} r} \qquad \frac{\ell_0 \mathrel{\nabla_{e_0}} r_0 \qquad \ell_1 \mathrel{\nabla_{e_1}} r_1}{\ell_0 \parallel \ell_1 \mathrel{\nabla_{e_0 \parallel e_1}} r_0 \parallel r_1} \qquad \frac{\ell \mathrel{\nabla_{e_0}} r}{e_0^\star \cdot \ell \mathrel{\nabla_{e_0^\star}} r \cdot e_0^\star}$$

Given $e \in \mathcal{T}$, we refer to ∇_e as the *sequential splitting relation* of e, and to the elements of ∇_e as *sequential splices* of e. We need to establish a few properties of the sequential splitting relation that will be useful later on. The first of these properties is that, as for parallel splitting, ∇_e is finite.

Lemma 4.8. *For $e \in \mathcal{T}$, ∇_e is finite.*

We also have that the sequential composition of splices is provably below the term being split. Just like the analogous lemma for parallel splitting, this guarantees that our sequential splices never give rise to semantics not contained in the split term. This lemma also yields an observation about the width of sequential splices when compared to the term being split.

Lemma 4.9. *Let $e \in \mathcal{T}$. If $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\nabla_e} r$, then $\ell \cdot r \leqq_{\mathsf{CKA}} e$.*

Corollary 4.2. *Let $e \in \mathcal{T}$. If $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\nabla_e} r$, then $|\ell|, |r| \leq |e|$.*

Lastly, we show that the splices cover every way of (sequentially) splitting up the semantics of the term being split, i.e., that ∇_e is dense when it comes to sequentially composed pomsets.

Lemma 4.10. *Let $e \in \mathcal{T}$, and let V and W be pomsets such that $V \cdot W \in [\![e]\!]_{\mathsf{CKA}}$. Then there exist $\ell, r \in \mathcal{T}$ with $\ell \mathrel{\nabla_e} r$ such that $V \in [\![\ell]\!]_{\mathsf{CKA}}$ and $W \in [\![r]\!]_{\mathsf{CKA}}$.*

Proof. The proof proceeds by induction on e. In the base, we can discount the case where $e = 0$, for then the claim holds vacuously. This leaves us two cases.

[3] The contents of this relation are very similar to the set of *left- and right-spines* of a NetKAT expression as used in [5].

– If $e = 1$, then $V \cdot W = 1$; by Lemma 3.1, we find that $V = W = 1$. Since $1 \nabla_e 1$ by definition of ∇_e, the claim follows when we choose $\ell = r = 1$.
– If $e = a$ for some $a \in \Sigma$, then $V \cdot W = a$; by Lemma 3.1, we find that either $V = a$ and $W = 1$ or $V = 1$ and $W = a$. In the former case, we can choose $\ell = a$ and $r = 1$ to satisfy the claim; the latter case can be treated similarly.

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

– If $e = e_0 + e_1$, then $V \cdot W \in [\![e_i]\!]_{\mathsf{CKA}}$ for some $i \in 2$. By induction, we find $\ell, r \in \mathcal{T}$ with $\ell \nabla_{e_i} r$ such that $V \in [\![\ell]\!]_{\mathsf{CKA}}$ and $W \in [\![r]\!]_{\mathsf{CKA}}$. Since $\ell \nabla_e r$ in this case, the claim follows.
– If $e = e_0 \cdot e_1$, then there exist $U_0 \in [\![e_0]\!]_{\mathsf{CKA}}$ and $U_1 \in [\![e_1]\!]_{\mathsf{CKA}}$ such that $V \cdot W = U_0 \cdot U_1$. By Lemma 3.4, we find a series-parallel pomset X such that either $V \sqsubseteq U_0 \cdot X$ and $X \cdot W \sqsubseteq U_1$, or $V \cdot X \sqsubseteq U_0$ and $W \sqsubseteq X \cdot U_1$. In the former case, we find that $X \cdot W \in [\![e_1]\!]_{\mathsf{CKA}}$, and thus by induction $\ell', r \in \mathcal{T}$ with $\ell' \nabla_{e_1} r$ such that $X \in [\![\ell']\!]_{\mathsf{CKA}}$ and $W \in [\![r]\!]_{\mathsf{CKA}}$. We then choose $\ell = e_0 \cdot \ell'$ to find that $\ell \nabla_e r$, as well as $V \sqsubseteq U_0 \cdot X \in [\![e_0]\!]_{\mathsf{CKA}} \cdot [\![\ell']\!]_{\mathsf{CKA}} = [\![\ell]\!]_{\mathsf{CKA}}$ and thus $V \in [\![\ell]\!]_{\mathsf{CKA}}$. The latter case can be treated similarly; here, we use the induction hypothesis on e_0.
– If $e = e_0 \parallel e_1$, then there exist $U_0 \in [\![e_0]\!]_{\mathsf{CKA}}$ and $U_1 \in [\![e_1]\!]_{\mathsf{CKA}}$ such that $V \cdot W \sqsubseteq U_0 \parallel U_1$. By Lemma 3.6, we find series-parallel pomsets V_0, V_1, W_0, W_1 such that $V \sqsubseteq V_0 \parallel V_1$ and $W \sqsubseteq W_0 \parallel W_1$, as well as $V_i \cdot W_i \sqsubseteq U_i$ for all $i \in 2$. In that case, $V_i \cdot W_i \in [\![e_i]\!]_{\mathsf{CKA}}$ for all $i \in 2$, and thus by induction we find $\ell_i, r_i \in \mathcal{T}$ with $\ell_i \nabla_{e_i} r_i$ such that $V_i \in [\![\ell_i]\!]_{\mathsf{CKA}}$ and $W_i \in [\![r_i]\!]_{\mathsf{CKA}}$. We choose $\ell = \ell_0 \parallel \ell_1$ and $r = r_0 \parallel r_1$ to find that $V \in [\![\ell_0 \parallel r_0]\!]_{\mathsf{CKA}}$ and $W \in [\![\ell_1 \parallel r_1]\!]_{\mathsf{CKA}}$, as well as $\ell \nabla_e r$.
– If $e = e_0^\star$, then there exist $U_0, U_1, \ldots, U_{n-1} \in [\![e_0]\!]_{\mathsf{CKA}}$ such that $V \cdot W = U_0 \cdot U_1 \cdots U_{n-1}$. Without loss of generality, we can assume that for $0 \leq i < n$ it holds that $U_i \neq 1$. In the case where $n = 0$ we have that $V \cdot W = 1$, thus $V = W = 1$, we can choose $\ell = r = 1$ to satisfy the claim.
For the case where $n > 0$, we find by Lemma 3.4 an $0 \leq m < n$ and series-parallel pomsets X, Y such that $X \cdot Y \sqsubseteq U_m$, and $V \sqsubseteq U_0 \cdot U_1 \cdots U_{m-1} \cdot X$ and $W \sqsubseteq Y \cdot U_{m+1} \cdot U_{m+2} \cdots U_n$. Since $X \cdot Y \sqsubseteq U_m \in [\![e_0]\!]_{\mathsf{CKA}}$ and thus $X \cdot Y \in [\![e_0]\!]_{\mathsf{CKA}}$, we find by induction $\ell', r' \in \mathcal{T}$ with $\ell' \nabla_{e_0} r'$ and $X \in [\![\ell']\!]_{\mathsf{CKA}}$ and $Y \in [\![r']\!]_{\mathsf{CKA}}$. We can then choose $\ell = e_0^\star \cdot \ell'$ and $r = r' \cdot e_0^\star$ to find that $V \sqsubseteq U_0 \cdot U_1 \cdots U_{m-1} \cdot X \in [\![e_0^\star]\!]_{\mathsf{CKA}} \cdot [\![\ell']\!]_{\mathsf{CKA}} = [\![\ell]\!]_{\mathsf{CKA}}$ and $W \sqsubseteq Y \cdot U_{m+1} \cdot U_{m+2} \cdots U_n \in [\![r']\!]_{\mathsf{CKA}} \cdot [\![e_0^\star]\!]_{\mathsf{CKA}} = [\![r]\!]_{\mathsf{CKA}}$, and thus that $V \in [\![\ell]\!]_{\mathsf{CKA}}$ and $W \in [\![r]\!]_{\mathsf{CKA}}$. Since $\ell \nabla_e r$ holds, the claim follows. □

Example 4.5. Let U be the pomset ca and let V be bc. Furthermore, let e be the term $(a \cdot b + c)^\star$, and note that $U \cdot V \in [\![e]\!]_{\mathsf{CKA}}$. We then find that $a \nabla_a 1$, and thus $a \nabla_{a \cdot b} 1 \cdot b$. We can now choose $\ell = (a \cdot b + c)^\star \cdot a$ and $r = (1 \cdot b) \cdot (a \cdot b + c)^\star$ to find that $U \in [\![\ell]\!]_{\mathsf{CKA}}$ and $V \in [\![r]\!]_{\mathsf{CKA}}$, while $\ell \nabla_e r$.

We know how to split a term sequentially. To resolve the second problem, we need to show that the process of splitting terms repeatedly ends somewhere. This is formalised in the notion of *right-hand remainders*, which are the terms that can appear as the right hand of a sequential splice of a term.

Definition 4.6. *Let $e \in \mathcal{T}$. The set of* (right-hand) remainders *of e, written $R(e)$, is the smallest satisfying the rules*

$$\frac{}{e \in R(e)} \qquad \frac{f \in R(e) \qquad \ell \nabla_f r}{r \in R(e)}$$

Lemma 4.11. *Let $e \in \mathcal{T}$. $R(e)$ is finite.*

With splitting and remainders we are in a position to define the linear system that will yield the closure of a parallel composition. Intuitively, we can think of this system as an automaton: every variable corresponds to a state, and every row of the matrix describes the "transitions" of the corresponding state, while every element of the vector describes the language "accepted" by that state without taking a single transition. Solving the system for a least fixpoint can be thought of as finding an expression that describes the language of the automaton.

Definition 4.7. *Let $e, f \in \mathcal{T}$, and suppose that, for every $g \in \mathcal{T}$ such that $|g| < |e| + |f|$, there exists a closure $g{\downarrow}$. We choose*

$$I_{e,f} = \{g \parallel h : g \in R(e), h \in R(f)\}$$

The $I_{e,f}$-vector $p_{e,f}$ and $I_{e,f}$-matrix $M_{e,f}$ are chosen as follows.

$$p_{e,f}(g \parallel h) \triangleq g \parallel f \qquad M_{e,f}(g \parallel h, g' \parallel h') \triangleq \sum_{\substack{\ell_g \nabla_g g' \\ \ell_h \nabla_h h'}} \ell_g \odot \ell_h$$

$I_{e,f}$ is finite by Lemma 4.11. We write $\mathfrak{L}_{e,f}$ for the $I_{e,f}$-linear system $\langle M_{e,f}, p_{e,f} \rangle$.

We can check that $M_{e,f}$ is well-defined. First, the sum is finite, because ∇_g and ∇_h are finite by Lemma 4.8. Second, if $g \parallel h \in I$ and $\ell_g, r_g, \ell_h, r_h \in \mathcal{T}$ such that $\ell_g \nabla_g r_g$ and $\ell_h \nabla_h r_h$, then $|\ell_g| \leq |g| \leq |e|$ and $|\ell_h| \leq |h| \leq |f|$ by Corollary 4.2, and thus, if $d \in \mathcal{T}$ such that $|d| < |\ell_g| + |\ell_h|$, then $|d| < |e| + |f|$, and therefore a closure of d exists, meaning that $\ell_g \odot \ell_h$ exists, too.

The least solution to $\mathfrak{L}_{e,f}$ obtained through Lemma 3.12 is the I-vector denoted by $s_{e,f}$. We write $e \otimes f$ for $s_{e,f}(e \parallel f)$, i.e., the least solution at $e \parallel f$.

Using the previous lemmas, we can then show that $e \otimes f$ is indeed a closure of $e \parallel f$, provided that we have closures for all terms of strictly lower width. The intuition of this proof is that we use the uniqueness of least fixpoints to show that $e \parallel f \equiv_{\mathsf{CKA}} e \otimes f$, and then use the properties of preclosure and the normal form of series-parallel pomsets to show that $[\![e \parallel f]\!]_{\mathsf{CKA}} = [\![e \otimes f]\!]_{\mathsf{BKA}}$.

Lemma 4.12. *Let $e, f \in \mathcal{T}$, and suppose that, for every $g \in \mathcal{T}$ with $|g| < |e| + |f|$, there exists a closure $g{\downarrow}$. Then $e \otimes f$ is a closure of $e \parallel f$.*

Proof. We begin by showing that $e \parallel f \equiv_{\mathsf{CKA}} e \otimes f$. We can see that $p_{e,f}$ is a solution to $\mathfrak{L}_{e,f}$, by calculating for $g \parallel h \in I_{e,f}$:

$$
\begin{aligned}
&(p_{e,f} + M_{e,f} \cdot p_{e,f})(g \parallel h) \\
&= g \parallel h + \sum_{r_g \parallel r_h \in I} \Big(\sum_{\substack{\ell_g \nabla_g r_g \\ \ell_h \nabla_h r_h}} \ell_g \odot \ell_h \Big) \cdot (r_g \parallel r_h) && \text{(def. } M_{e,f}, p_{e,f}\text{)} \\
&\equiv_{\mathsf{CKA}} g \parallel h + \sum_{r_g \parallel r_h \in I} \sum_{\substack{\ell_g \nabla_g r_g \\ \ell_h \nabla_h r_h}} (\ell_g \odot \ell_h) \cdot (r_g \parallel r_h) && \text{(distributivity)} \\
&\equiv_{\mathsf{CKA}} g \parallel h + \sum_{r_g \parallel r_h \in I} \sum_{\substack{\ell_g \nabla_g r_g \\ \ell_h \nabla_h r_h}} (\ell_g \parallel \ell_h) \cdot (r_g \parallel r_h) && \text{(Lemma 4.7)} \\
&\leqq_{\mathsf{CKA}} g \parallel h + \sum_{r_g \parallel r_h \in I} \sum_{\substack{\ell_g \nabla_g r_g \\ \ell_h \nabla_h r_h}} (\ell_g \cdot r_g) \parallel (\ell_h \cdot r_h) && \text{(exchange)} \\
&\leqq_{\mathsf{CKA}} g \parallel h + \sum_{r_g \parallel r_h \in I} \sum_{\substack{\ell_g \nabla_g r_g \\ \ell_h \nabla_h r_h}} g \parallel h && \text{(Lemma 4.9)} \\
&\equiv_{\mathsf{CKA}} g \parallel h && \text{(idempotence)} \\
&= p_{e,f}(g \parallel h) && \text{(def. } p_{e,f}\text{)}
\end{aligned}
$$

To see that $p_{e,f}$ is the *least* solution to $\mathfrak{L}_{e,f}$, let $q_{e,f}$ be a solution to $\mathfrak{L}_{e,f}$. We then know that $M_{e,f} \cdot q_{e,f} + p_{e,f} \leqq_{\mathsf{CKA}} q_{e,f}$; thus, in particular, $p_{e,f} \leqq_{\mathsf{CKA}} q_{e,f}$. Since the least solution to a linear system is unique up to $\equiv_{\mathsf{CKA}}$, we find that $s_{e,f} \equiv_{\mathsf{CKA}} p_{e,f}$, and therefore that $e \otimes f = s_{e,f}(e \parallel f) \equiv_{\mathsf{CKA}} p_{e,f}(e \parallel f) = e \parallel f$.

It remains to show that if $U \in [\![e \parallel f]\!]_{\mathsf{CKA}}$, then $U \in [\![e \otimes f]\!]_{\mathsf{BKA}}$. To show this, we show the more general claim that if $g \parallel h \in I$ and $U \in [\![g \parallel h]\!]_{\mathsf{CKA}}$, then $U \in [\![s_{e,f}(g \parallel h)]\!]_{\mathsf{BKA}}$. Write $U = U_0 \cdot U_1 \cdots U_{n-1}$ such that for $0 \leq i < n$, U_i is non-sequential (as in Corollary 3.1). The proof proceeds by induction on n. In the base, we have that $n = 0$. In this case, $U = 1$, and thus $U \in [\![g \parallel h]\!]_{\mathsf{BKA}}$ by Lemma 3.2. Since $g \parallel h = p_{e,f}(g \parallel h) \leqq_{\mathsf{BKA}} s_{e,f}(g \parallel h)$, it follows that $U \in [\![s_{e,f}(g \parallel h)]\!]_{\mathsf{BKA}}$ by Lemma 3.8.

For the inductive step, assume the claim holds for $n-1$. We write $U = U_0 \cdot U'$, with $U' = U_1 \cdot U_2 \cdots U_{n-1}$. Since $U_0 \cdot U' \in [\![g \parallel h]\!]_{\mathsf{CKA}}$, there exist $W \in [\![g]\!]_{\mathsf{CKA}}$ and $X \in [\![h]\!]_{\mathsf{CKA}}$ such that $U_0 \cdot U' \sqsubseteq W \parallel X$. By Lemma 3.6, we find pomsets W_0, W_1, X_0, X_1 such that $W_0 \cdot W_1 \sqsubseteq W$ and $X_0 \cdot X_1 \sqsubseteq X$, as well as $U_0 \sqsubseteq W_0 \parallel X_0$ and $U' \sqsubseteq W_1 \parallel X_1$. By Lemma 4.10, we find $\ell_g, r_g, \ell_h, r_h \in \mathcal{T}$ with $\ell_g \nabla_g r_g$ and $\ell_h \nabla_h r_h$, such that $W_0 \in [\![\ell_g]\!]_{\mathsf{CKA}}$, $W_1 \in [\![r_g]\!]_{\mathsf{CKA}}$, $X_0 \in [\![\ell_h]\!]_{\mathsf{CKA}}$ and $X_1 \in [\![r_h]\!]_{\mathsf{CKA}}$.

From this, we know that $U_0 \in [\![\ell_g \parallel \ell_h]\!]_{\mathsf{CKA}}$ and $U' \in [\![r_g \parallel r_h]\!]_{\mathsf{CKA}}$. Since U_0 is non-sequential, we have that $U_0 \in [\![\ell_g \odot \ell_h]\!]_{\mathsf{BKA}}$. Moreover, by induction we find that $U' \in [\![s_{e,f}(r_g \parallel r_h)]\!]_{\mathsf{BKA}}$. Since $\ell_g \odot \ell_h \leqq_{\mathsf{BKA}} M_{e,f}(g \parallel h, r_g \parallel r_h)$ by definition of $M_{e,f}$, we furthermore find that

$$(\ell_g \odot \ell_h) \cdot s_{e,f}(r_g \parallel r_h) \leqq_{\mathsf{BKA}} M_{e,f}(g \parallel h, r_g \parallel r_h) \cdot s_{e,f}(r_g \parallel r_h)$$

Since $r_g \parallel r_h \in I$, we find by definition of the solution to a linear system that

$$M_{e,f}(g \parallel h, r_g \parallel r_h) \cdot s_{e,f}(r_g \parallel r_h) \leqq_{\mathsf{BKA}} s_{e,f}(g \parallel h)$$

By Lemma 3.8 and the above, we conclude that $U = U_0 \cdot U' \in [\![s_{e,f}(g \parallel h)]\!]_{\mathsf{BKA}}$. □

For a concrete example where we find a closure of a (non-trivial) parallel composition by solving a linear system, we refer to Appendix A.

With closure of parallel composition, we can construct a closure for any term and therefore conclude completeness of CKA.

Theorem 4.1. *Let $e \in \mathcal{T}$. We can construct a closure $e{\downarrow}$ of e.*

Proof. The proof proceeds by induction on $|e|$ and the structure of e, i.e., by considering f before g if $|f| < |g|$, or if f is a strict subterm of g (in which case $|f| \leq |g|$ also holds). It is not hard to see that this induces a well-ordering on $\mathcal{T}$.

Let e be a term of width n, and suppose that the claim holds for all terms of width at most $n-1$, and for all strict subterms of e. There are three cases.

- If $e = 0$, $e = 1$ or $e = a$ for some $a \in \Sigma$, the claim follows from Lemma 4.2.
- If $e = e_0 + e_1$, or $e = e_0 \cdot e_1$, or $e = e_0^\star$, the claim follows from Lemma 4.3.
- If $e = e_0 \parallel e_1$, then $e_0 \otimes e_1$ exists by the induction hypothesis. By Lemma 4.12, we then find that $e_0 \otimes e_1$ is a closure of e. □

Corollary 4.3. *Let $e, f \in \mathcal{T}$. If $[\![e]\!]_{\mathsf{CKA}} = [\![f]\!]_{\mathsf{CKA}}$, then $e \equiv_{\mathsf{CKA}} f$.*

Proof. Follows from Theorem 4.1 and Lemma 4.1. □

5 Discussion and Further Work

By building a syntactic closure for each series-rational expression, we have shown that the standard axiomatisation of CKA is complete with respect to the CKA-semantics of series-rational terms. Consequently, the algebra of closed series-rational pomset languages forms the free CKA.

Our result leads to several decision procedures for the equational theory of CKA. For instance, one can compute the closure of a term as described in the present paper, and use an existing decision procedure for BKA [3,12,20]. Note however that although this approach seems suited for theoretical developments (such as formalising the results in a proof assistant), its complexity makes it less appealing for practical use. More practically, one could leverage recent work by Brunet et al. [3], which provides an algorithm to compare closed series-rational pomset languages. Since this is the free concurrent Kleene algebra, this algorithm can now be used to decide the equational theory of CKA. We also obtain from the latter paper that this decision problem is EXPSPACE-complete.

We furthermore note that the algorithm to compute downward closure can be used to extend half of the result from [14] to a Kleene theorem that relates the CKA-semantics of expressions to the pomset automata proposed there: if $e \in \mathcal{T}$, we can construct a pomset automaton A with a state q such that $L_A(q) = [\![e]\!]_{\mathsf{CKA}}$.

Having established pomset automata as an operational model of CKA, a further question is whether these automata are amenable to a bisimulation-based equivalence algorithm, as is the case for finite automata [10]. If this is the case, optimisations such as those in [2] might have analogues for pomset automata that can be found using the coalgebraic method [23].

While this work was in development, an unpublished draft by Laurence and Struth [19] appeared, with a first proof of completeness for CKA. The general outline of their proof is similar to our own, in that they prove that closure of pomset languages preserves series-rationality, and hence there exists a syntactic closure for every series-rational expression. However, the techniques used to establish this fact are quite different from the developments in the present paper. First, we build the closure via syntactic methods: explicit splitting relations and solutions of linear systems. Instead, their proof uses automata theoretic constructions and algebraic closure properties of regular languages; in particular, they rely on congruences of finite index and language homomorphisms. We believe that our approach leads to a substantially simpler and more transparent proof. Furthermore, even though Laurence and Struth do not seem to use any fundamentally non-constructive argument, their proof does not obviously yield an algorithm to effectively compute the closure of a given term. In contrast, our proof is explicit enough to be implemented directly; we wrote a simple Python script (under six hundred lines) to do just that [16].

A crucial ingredient in this work was the computation of least solutions of linear systems. This kind of construction has been used on several occasions for the study of Kleene algebras [1,4,18], and we provide here yet another variation of such a result. We feel that linear systems may not have yet been used to their full potential in this context, and could still lead to interesting developments.

A natural extension of the work conducted here would be to turn our attention to the signature of concurrent Kleene algebra that includes a "parallel star" operator $e^{\parallel}$. The completeness result of Laurence and Struth [20] holds for BKA with the parallel star, so in principle one could hope to extend our syntactic closure construction to include this operator. Unfortunately, using the results of Laurence and Struth, we can show that this is not possible. They defined a notion of *depth* of a series-parallel pomset, intuitively corresponding to the nesting of parallel and sequential components. An important step in their development consists of proving that for every series-parallel-rational language there exists a finite upper bound on the depth of its elements. However, the language $[\![a^{\parallel}]\!]_{\mathsf{CKA}}$ does not enjoy this property: it contains every series-parallel pomset exclusively labelled with the symbol a. Since we can build such pomsets with arbitrary depth, it follows that there does not exist a syntactic closure of the term $a^{\parallel}$. New methods would thus be required to tackle the parallel star operator.

Another aspect of CKA that is not yet developed to the extent of KA is the coalgebraic perspective. We intend to investigate whether the coalgebraic tools developed for KA can be extended to CKA, which will hopefully lead to efficient bisimulation-based decision procedures [2,5].

Acknowledgements. We thank the anonymous reviewers for their insightful comments. This work was partially supported by the ERC Starting Grant ProFoundNet (grant code 679127).

A Worked Example: A Non-trivial Closure

In this appendix, we solve an instance of a linear system as defined in Definition 4.7 for a given parallel composition. For the sake of brevity, the steps are somewhat coarse-grained; the reader is encouraged to reproduce the steps by hand.

Consider the expression $e \parallel f = a^* \parallel b$. The linear system $\mathfrak{L}_{e,f}$ that we obtain from this expression consists of six inequations; in matrix form (with zeroes omitted), this system is summarised as follows:[4]

$$\begin{array}{r} 1 \parallel 1 \\ 1 \parallel b \\ a \cdot a^\star \parallel 1 \\ a^\star \parallel 1 \\ a \cdot a^\star \parallel b \\ a^\star \parallel b \end{array} \left(\begin{array}{cccccc|c} 1 & & & & & & 1 \\ b & 1 & & & & & b \\ a & & a^\star & a \cdot a^\star & & & a \cdot a^\star \\ 1 & & a^\star & a^\star \cdot a & & & a^\star \\ a \parallel b & a & a^\star \parallel b & a \cdot a^\star \parallel b & a^\star & a \cdot a^\star & a \cdot a^\star \parallel b \\ b & 1 & a^\star \parallel b & a \cdot a^\star \parallel b & a^\star & a \cdot a^\star & a^\star \parallel b \end{array} \right)$$

Let us proceed under the assumption that x is a solution to the system; the constraint imposed on x by the first two rows is given by the inequations

$$x(1 \parallel 1) + 1 \leqq_{\mathsf{CKA}} x(1 \parallel 1) \tag{1}$$

$$b \cdot x(1 \parallel 1) + x(1 \parallel b) + b \leqq_{\mathsf{CKA}} x(1 \parallel b) \tag{2}$$

Because these inequations do not involve the other positions of the system, we can solve them in isolation, and use their solutions to find solutions for the remaining positions; it turns out that choosing $x(1 \parallel 1) = 1$ and $x(1 \parallel b) = b$ suffices here.

We carry on to fill these values into the inequations given by the third and fourth row of the linear system. After some simplification, these work out to be

$$a \cdot a^\star + a \cdot a^\star \cdot x(a^\star \parallel 1) + a^\star \cdot x(a \cdot a^\star \parallel 1) \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel 1) \tag{3}$$

$$a^\star + a^\star \cdot a \cdot x(a^\star \parallel 1) + a^\star \cdot x(a \cdot a^\star \parallel 1) \leqq_{\mathsf{CKA}} x(a^\star \parallel 1) \tag{4}$$

Applying the least fixpoint axiom to (3) and simplifying, we obtain

$$a \cdot a^\star + a \cdot a^\star \cdot x(a^\star \parallel 1) \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel 1) \tag{5}$$

Substituting this into (4) and simplifying, we find that

$$a^\star + a \cdot a^\star \cdot x(a^\star \parallel 1) \leqq_{\mathsf{CKA}} x(a^\star \parallel 1) \tag{6}$$

This inequation, in turn, gives us that $a^\star \leqq_{\mathsf{CKA}} x(a^\star \parallel 1)$ by the least fixpoint axiom. Plugging this back into (3) and simplifying, we find that

$$a \cdot a^\star + a^\star \cdot x(a \cdot a^\star \parallel 1) \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel 1) \tag{7}$$

[4] Actually, the system obtained from $a^\star \parallel b$ as a result of Definition 4.7 is slightly larger; it also contains rows and columns labelled by $1 \cdot a^\star \parallel 1$ and $1 \cdot a^\star \parallel b$; these turn out to be redundant. We omit these rows from the example for simplicity.

Again by the least fixpoint axiom, this tells us that $a \cdot a^\star \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel 1)$. One easily checks that $x(a \cdot a^\star \parallel 1) = a \cdot a^\star$ and $x(a^\star \parallel 1) = a^\star$ are solutions to (3) and (4); by the observations above, they are also the least solutions.

It remains to find the least solutions for the final two positions. Filling in the values that we already have, we find the following for the fifth row:

$$\begin{aligned} &a \parallel b + a \cdot b + (a^\star \parallel b) \cdot a \cdot a^\star + (a \cdot a^\star \parallel b) \cdot a^\star \\ &\quad + a^\star \cdot x(a \cdot a^\star \parallel b) + a \cdot a^\star \cdot x(a^\star \parallel b) + a \cdot a^\star \parallel b \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel b) \end{aligned} \tag{8}$$

Applying the exchange law[5] to the first three terms, we find that they are contained in $(a \cdot a^\star \parallel b) \cdot a^\star$, as is the last term; (8) thus simplifies to

$$(a \cdot a^\star \parallel b) \cdot a^\star + a^\star \cdot x(a \cdot a^\star \parallel b) + a \cdot a^\star \cdot x(a^\star \parallel b) \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel b) \tag{9}$$

By the least fixpoint axiom, we find that

$$a^\star \cdot (a \cdot a^\star \parallel b) \cdot a^\star + a \cdot a^\star \cdot x(a^\star \parallel b) \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel b) \tag{10}$$

For the sixth row, we find that after filling in the solved positions, we have

$$\begin{aligned} &b + b + (a^\star \parallel b) \cdot a \cdot a^\star + (a \cdot a^\star \parallel b) \cdot a^\star \\ &\quad + a^\star \cdot x(a \cdot a^\star \parallel b) + a \cdot a^\star \cdot x(a^\star \parallel b) + a^\star \parallel b \leqq_{\mathsf{CKA}} x(a^\star \parallel b) \end{aligned} \tag{11}$$

Simplifying and applying the exchange law as before, it follows that

$$(a^\star \parallel b) \cdot a^\star + a^\star \cdot x(a \cdot a^\star \parallel b) + a \cdot a^\star \cdot x(a^\star \parallel b) \leqq_{\mathsf{CKA}} x(a^\star \parallel b) \tag{12}$$

We then subsitute (10) into (12) to find that

$$(a^\star \parallel b) \cdot a^\star + a \cdot a^\star \cdot x(a^\star \parallel b) \leqq_{\mathsf{CKA}} x(a^\star \parallel b) \tag{13}$$

which, by the least fixpoint axiom, tells us that $a^\star \cdot (a^\star \parallel b) \cdot a^\star \leqq_{\mathsf{CKA}} x(a^\star \parallel b)$. Plugging the latter back into (9), we find that

$$a^\star \cdot (a \cdot a^\star \parallel b) \cdot a^\star + a \cdot a^\star \cdot a^\star \cdot (a^\star \parallel b) \cdot a^\star \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel b) \tag{14}$$

which can, using the exchange law, be reworked into

$$a^\star \cdot (a \cdot a^\star \parallel b) \cdot a^\star \leqq_{\mathsf{CKA}} x(a \cdot a^\star \parallel b) \tag{15}$$

Now, if we choose $x(a \cdot a^\star \parallel b) = a^\star \cdot (a \cdot a^\star \parallel b) \cdot a^\star$ and $x(a^\star \parallel b) = a^\star \cdot (a^\star \parallel b) \cdot a^\star$, we find that these choices satisfy (9) and (12)—making them part of a solution; by construction, they are also the least solutions.

In summary, x is a solution to the linear system, and by construction it is also the least solution. The reader is encouraged to verify that our choice of $x(a^\star \parallel b)$ is indeed a closure of $a^\star \parallel b$.

[5] A caveat here is that applying the exchange law indiscriminately may lead to a term that is not a closure (specifically, it may violate the semantic requirement in Definition 4.1). The algorithm used to solve arbitrary linear systems in Lemma 3.12 does not make use of the exchange law to simplify terms, and thus avoids this pitfall.

References

1. Backhouse, R.: Closure algorithms and the star-height problem of regular languages. Ph.D. thesis, University of London (1975)
2. Bonchi, F., Pous, D.: Checking NFA equivalence with bisimulations up to congruence. In: Proceedings of the Principles of Programming Languages (POPL), pp. 457–468 (2013)
3. Brunet, P., Pous, D., Struth, G.: On decidability of concurrent Kleene algebra. In: Proceedings of the Concurrency Theory (CONCUR), pp. 28:1–28:15 (2017)
4. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall Ltd., London (1971)
5. Foster, N., Kozen, D., Milano, M., Silva, A., Thompson, L.: A coalgebraic decision procedure for NetKAT. In: Proceedings of the Principles of Programming Languages (POPL), pp. 343–355 (2015)
6. Gischer, J.L.: The equational theory of pomsets. Theor. Comput. Sci. **61**, 199–224 (1988)
7. Grabowski, J.: On partial languages. Fundam. Inform. **4**(2), 427 (1981)
8. Hoare, T., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra. In: Proceedings of the Concurrency Theory (CONCUR), pp. 399–414 (2009)
9. 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)
10. Hopcroft, J.E., Karp, R.M.: A linear algorithm for testing equivalence of finite automata. Technical report, TR71-114, December 1971
11. Horn, A., Kroening, D.: On partial order semantics for SAT/SMT-based symbolic encodings of weak memory concurrency. In: Graf, S., Viswanathan, M. (eds.) FORTE 2015. LNCS, vol. 9039, pp. 19–34. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19195-9_2
12. Jategaonkar, L., Meyer, A.R.: Deciding true concurrency equivalences on safe, finite nets. Theor. Comput. Sci. **154**(1), 107–143 (1996)
13. Jipsen, P., Moshier, M.A.: Concurrent Kleene algebra with tests and branching automata. J. Log. Algebr. Methods Program. **85**(4), 637–652 (2016)
14. Kappé, T., Brunet, P., Luttik, B., Silva, A., Zanasi, F.: Brzozowski goes concurrent—a Kleene theorem for pomset languages. In: Proceedings of the Concurrency Theory (CONCUR), pp. 25:1–25:16 (2017)
15. Kappé, T., Brunet, P., Silva, A., Zanasi, F.: Concurrent Kleene algebra: free model and completeness. https://arxiv.org/abs/1710.02787
16. Kappé, T., Brunet, P., Silva, A., Zanasi, F.: Tools for concurrent Kleene algebra, Sep 2017. https://doi.org/10.5281/zenodo.926823
17. Kleene, S.C.: Representation of events in nerve nets and finite automata. In: Shannon, C.E., McCarthy, J. (eds.) Automata Studies, pp. 3–41. Princeton University Press, Princeton (1956)
18. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Inf. Comput. **110**(2), 366–390 (1994)
19. Laurence, M.R., Struth, G.: Completeness theorems for pomset languages and concurrent Kleene algebras. https://arxiv.org/abs/1705.05896
20. Laurence, M.R., Struth, G.: Completeness theorems for Bi-Kleene algebras and series-parallel rational pomset languages. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M.E. (eds.) RAMICS 2014. LNCS, vol. 8428, pp. 65–82. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06251-8_5
21. Levi, F.W.: On semigroups. Bull. Calcutta Math. Soc. **36**(141–146), 82 (1944)

22. Lodaya, K., Weil, P.: Series-parallel languages and the bounded-width property. Theor. Comput. Sci. **237**(1), 347–380 (2000)
23. Rot, J., Bonsangue, M., Rutten, J.: Coalgebraic bisimulation-up-to. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 369–381. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35843-2_32