Completeness and the Finite Model Property for Kleene Algebra, Reconsidered

Tobias Kappé[1,2]

[1] Open University of the Netherlands
[2] ILLC, University of Amsterdam
tobias.kappe@ou.nl

Abstract. Kleene Algebra (KA) is the algebra of regular expressions. Central to the study of KA is Kozen's (1994) *completeness* result, which says that any equivalence valid in the language model of KA follows from the axioms of KA. Also of interest is the *finite model property* (FMP), which says that false equivalences always have a finite counterexample. Palka (2005) showed that, for KA, the FMP is equivalent to completeness.

We provide a unified and elementary proof of both properties. In contrast with earlier completeness proofs, this proof does not rely on minimality or bisimilarity techniques for deterministic automata. Instead, our approach avoids deterministic automata altogether, and uses Antimirov's derivatives and the well-known transition monoid construction.

Our results are fully verified in the Coq proof assistant.

1 Introduction

Kleene Algebra (KA) [10,17] provides an algebraic perspective on the equivalence of regular expressions. It is the foundation for Kleene Algebra with Tests (KAT) [18,23,9,19], which has been applied to reason about equivalence of programs in general [21,24], and programming languages such as NetKAT [1,33].

Central to the study of Kleene Algebra and its extensions is the *completeness* property, which says that every equivalence that is valid in the language model can be proved using the laws of KA. An important precursor to this result was first shown by Salomaa [31], who provided a finite set of axioms in the form of quasi-equations, sufficient for proving all valid equivalences of regular expressions. Later works by various authors [10,6,25,13] studied alternative axiomatizations.

The axiomatization most commonly used today is due to Kozen [17], and has the advantage of being *algebraic*, i.e., it allows one to define a "Kleene algebra" as a mathematical object that may verify or falsify equations. A number of alternative proofs of the same result have been proposed [20,14,12,22]; notably, it was shown that one of the quasi-equations can be dropped from Kozen's axioms [12,22].

Another phenomenon of interest is the *finite model property* (FMP) [5]. For KA, the FMP states that any *invalid* equivalence is witnessed by some finite Kleene algebra where it does not hold — contrapositively, equivalences valid in any finite Kleene algebra are also valid in any (possibly infinite) Kleene algebra.

Palka [28] showed that the FMP is a consequence of completeness for KA, and moreover that completeness can be recovered if one assumes the FMP. This equivalence raises a question: can we provide an elementary proof of the FMP for KA, i.e., one that does not rely on completeness? Indeed, Palka writes that "an independent proof of [the FMP] would provide a quite different proof of the Kozen completeness theorem, based on purely logical tools" [28].

Our main contribution is a positive answer to this question, providing a proof of the FMP for KA. More specifically, our argument weaves together considerations from Palka's proof as well as classical facts from automata theory, in such a way that both the FMP and completeness can be concluded.

In contrast with earlier completeness proofs, our method does not center on minimality [17], bisimilarity [22,14] or the construction of a cyclic proof system [12]. Instead, we rely purely on the fact that KA allows one to find least solutions to linear systems [10,3,18], or in our case, to automata. The arguments towards our main result exploit this property in concert with various ideas around automata, such as the transition monoid [26], and Antimirov's construction [2], eventually building a particular finite Kleene algebra with sufficient structure to conclude both completeness and the finite model property.

The remainder of this paper is organized as follows. Section 2 provides an overview of the context, and defines fundamental notions. Section 3 recalls the notion of *solutions to an automaton*, a technique that will be leveraged repeatedly. Sections 4 and 5 provide an algebraic perspective on transformation automata [26], and Antimirov's construction [2] respectively. Section 6 shows how to construct a particularly useful Kleene algebra, and Section 7 shows how to conclude completeness and the FMP using the notions discussed up to that point. Section 8 concludes with some discussion and suggestions for future work.

To save space, most proofs of auxiliary facts appear in the appendices. Our formalization of the proofs in Coq is available online [15].

2 Overview

Our primary objects of study are *Kleene algebras*. The equations that hold in a Kleene algebra correspond well to properties expected of program composition, which makes them a suitable semantic domain for programs.

Definition 2.1. *A* (weak[3]) Kleene algebra (KA) *is a tuple* $(K, +, \cdot, ^*, 0, 1)$*, where* K *is a set (the* carrier*),* $+$ *and* $\cdot$ *are binary operators on* K*,* * *is a unary operator on* K*, and* $0, 1 \in K$ *are constants, satisfying the following for all* $x, y, z \in K$*:*

$$x + 0 = x \qquad x + x = x \qquad x + y = y + x \qquad x + (y + z) = (x + y) + z$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \qquad x \cdot (y + z) = x \cdot y + x \cdot z \qquad (x + y) \cdot z = x \cdot z + y \cdot z$$

$$x \cdot 1 = x = 1 \cdot x \qquad x \cdot 0 = 0 = 0 \cdot x \qquad 1 + x \cdot x^* = x^* \qquad x + y \cdot z \leq z \implies y^* \cdot x \leq z$$

[3] This is a "weak" KA in the sense that it does not require the right-unrolling and right-fixpoint laws from [18]. As it turns out, this does not change the equational theory [12,22]. For the sake of brevity, we omit "weak" in the remainder of this paper.

Here, we use $\leq$ to denote the natural order *induced by $+$, that is, $x \leq y$ if and only if $x + y = y$; it is straightforward to verify that this makes $\leq$ a partial order on K, and that all operators are monotone w.r.t. this order.*

We often denote a generic KA $(K, +, \cdot, ^*, 0, 1)$ by its carrier K, and simply write $+$, $\cdot$, etc. for the operators and constants when there is no risk of ambiguity.

Typically, the additive operator $+$ is used to implement nondeterministic composition, the multiplicative operator $\cdot$ corresponds to sequential composition, the *Kleene star* operator * implements iteration, 0 represents a program that fails immediately, and 1 is the program that does nothing and terminates successfully. The equations of KA correspond well to what might be expected of such operators on programs — for instance, and iteration is characterized as a least fixpoint.

One very natural instance of Kleene algebras, which we will connect to the interpretation of programs shortly, is given by the *relational model*

Example 2.2 (KA of relations). Let X be a set. The set of relations on X, i.e., $\mathcal{P}(X \times X)$, can be equipped with a KA $\mathcal{R}_X = (\mathcal{P}(X \times X), \cup, \circ, ^*, \emptyset, \mathsf{id}_X)$, in which $\circ$ is relational composition; * is the reflexive-transitive closure operator on relations; and id_X is the diagonal or identity relation on X given by $\{(x, x) : x \in X\}$.

When interpreting programs in $\mathcal{R}_X$, we think of the relations on X as a way of representing how a program may transform the machine states represented by X. To make this more precise, we need a syntax and semantics for programs.

Definition 2.3 (Expressions). *We fix a set of* actions $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \dots\}$ *called the* alphabet. *The set of* regular expressions $\mathbb{E}$ *is given by*

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

Given a KA K and a function $h : \Sigma \to K$, we can define $\widehat{h} : \mathbb{E} \to K$ inductively:

$$\widehat{h}(0) = 0 \qquad \widehat{h}(\mathsf{a}) = h(\mathsf{a}) \qquad \widehat{h}(e \cdot f) = \widehat{h}(e) \cdot \widehat{h}(f)$$
$$\widehat{h}(1) = 1 \qquad \widehat{h}(e + f) = \widehat{h}(e) + \widehat{h}(f) \qquad \widehat{h}(e^*) = \widehat{h}(e)^*$$

Example 2.4. Consider a programming language with integer variables $\mathsf{Var} = \{\mathtt{x}, \mathtt{y}, \dots\}$, and statements Σ comprised of (for all $\mathtt{x}, \mathtt{y} \in \mathsf{Var}$, $n \in \mathbb{N}$ and $v \in \mathsf{Var} \cup \mathbb{N}$) *assignments* $\mathtt{x} \leftarrow n$, *increments* $\mathtt{x} \leftarrow \mathtt{x} + v$, and *comparisons* $\mathtt{x} < \mathtt{y}, \mathtt{x} \geq \mathtt{y}$.

The state of the machine is defined by the value of each variable, and so we choose $S = \{\sigma : \mathsf{Var} \to \mathbb{N}\}$ as the state space. The semantics of the actions are relations that represent their effect, i.e., we define $h : \Sigma \to \mathcal{P}(S \times S)$ by

$$\begin{aligned}
h(\mathtt{x} \leftarrow n) &= \{(\sigma, \sigma[n/\mathtt{x}]) : \sigma \in S\} \\
h(\mathtt{x} \leftarrow \mathtt{x} + n) &= \{(\sigma, \sigma[\sigma(\mathtt{x}) + n/\mathtt{x}]) : \sigma \in S\} \\
h(\mathtt{x} \leftarrow \mathtt{x} + \mathtt{y}) &= \{(\sigma, \sigma[\sigma(\mathtt{x}) + \sigma(\mathtt{y})/\mathtt{x}]) : \sigma \in S\} \\
h(\mathtt{x} < \mathtt{y}) &= \{(\sigma, \sigma) : \sigma \in S, \sigma(\mathtt{x}) < \sigma(\mathtt{y})\} \\
h(\mathtt{x} \geq \mathtt{y}) &= \{(\sigma, \sigma) : \sigma \in S, \sigma(\mathtt{x}) \geq \sigma(\mathtt{y})\}
\end{aligned}$$

Here, $\sigma[n/\mathtt{x}]$ is denotes the function that assigns n to $\mathtt{x}$, and $\sigma(\mathtt{y})$ to all $\mathtt{y} \neq \mathtt{x}$.

This gives us a semantics $\widehat{h} : \mathbb{E} \to \mathcal{P}(S \times S)$ for regular expressions over Σ, and allows us to express and interpret programs like the following:

$$\mathtt{x} \leftarrow 1 \cdot \mathtt{y} \leftarrow 0 \cdot \mathtt{i} \leftarrow 0 \cdot (\mathtt{i} < \mathtt{n} \cdot \mathtt{y} \leftarrow \mathtt{y} + \mathtt{x} \cdot \mathtt{x} \leftarrow \mathtt{x} + 2 \cdot \mathtt{i} \leftarrow \mathtt{i} + 1)^* \cdot (\mathtt{i} \geq \mathtt{n})$$

which will compute the square of $\mathtt{n}$ and store it in $\mathtt{y}$.

Of course, one can build more involved programming languages based on KA; one elaborate and well-studied instance is NetKAT [1], a programming language for specifying and reasoning about software-defined networks.

Let $e, f \in \mathbb{E}$. When $\widehat{h}(e) = \widehat{h}(f)$ for all $h : \Sigma \to K$, we write $K \models e = f$. If $\mathfrak{C}$ is a class of KAs and $K \models e = f$ for each KA K in $\mathfrak{C}$, then we write $\mathfrak{C} \models e = f$. We use $\equiv$ for the smallest congruence on $\mathbb{E}$ that satisfies the axioms of KA, and use $e \leqq f$ as shorthand for $e + f \equiv f$. One can show that all operators are monotone w.r.t. the preorder $\leqq$, and that $e \leqq f$ and $f \leqq e$ together imply $e \equiv f$.

We use $[\phi]$ for some logical condition ϕ to denote $1 \in \mathbb{E}$ when ϕ holds, and $0 \in \mathbb{E}$ otherwise. We also use the familiar $\sum$ notation to generalize $+$. The empty sum is defined to be 0, the unit of $+$. Note that the sum notation is well-defined up to $\equiv$, because $+$ is associative, commutative and idempotent in any KA.

The following is a standard fact of universal algebra — see, e.g., [8].

Lemma 2.5. *Let $e, f \in \mathbb{E}$. We have $e \equiv f$ iff $K \models e = f$ for all KAs K.*

Given that KA provides such a suitable semantic domain, can we characterize the *equational theory* of KA, i.e., the equations valid in all models (programming languages) captured by $\equiv$, as the equations that hold for a particular model or class of models? Conversely, can we guarantee any properties of countermodels (pathological programming languages) that witness invalid equations?

Kozen [18] answered these questions by showing that the valid equations of KA are characterized by the *language model*. Intuitively, this model assigns to each expression the set of possible sequences of actions that may be executed by the program it represents. We will now make this more precise.

A *word* is a finite sequence of actions $\mathtt{a}_1 \cdots \mathtt{a}_n$; the *empty word* (with no letters) is denoted ϵ. We write Σ^* for the set of words, and denote its elements by $w, x, y, \dots$. We can *concatenate* words by juxtaposing them, i.e., if $w = \mathtt{a}_1 \cdots \mathtt{a}_n$ and $x = \mathtt{b}_1 \cdots \mathtt{b}_m$, then wx is the word given by $\mathtt{a}_1 \cdots \mathtt{a}_n \mathtt{b}_1 \cdots \mathtt{b}_m$.

A set of words $L, K, M, \dots$ is called a *language*. We can combine languages as one would combine sets (e.g., by taking their union). The concatenation of words can also be lifted to languages in a pointwise manner, writing $L \cdot K$ for the set $\{wx : w \in L, x \in K\}$. Finally, the *Kleene star* of a language L, denoted L^*, is the set $\{w_1 \cdots w_n : w_1, \dots, w_n \in L\}$. Note that L^* includes the empty word.

Definition 2.6. *The* KA of languages $\mathcal{L}$ *is given by* $(\mathcal{P}(\Sigma^*), \cup, \cdot, {}^*, \emptyset, \{\epsilon\})$*, where $\cdot$ is language concatenation, and * is the Kleene star of a language as above. We furthermore define the function $\ell : \Sigma \to \mathcal{P}(\Sigma^*)$ by $\ell(\mathtt{a}) = \{\mathtt{a}\}$.*

Remark 2.7. Readers familiar with regular languages will recognize that $\widehat{\ell} : \mathbb{E} \to \mathcal{P}(\Sigma^*)$ is the standard language interpretation of regular expressions.

Algebraically, Kozen's theorem can now be stated as follows.

Theorem 2.8 (Kozen [18]). *For all $e, f \in \mathbb{E}$, we have that $e \equiv f$ iff $\mathcal{L} \models e = f$.*

One of the payoffs of this result is that we can decide $e \equiv f$ by checking whether $\mathcal{L} \models e = f$, which turns out to be a PSPACE-complete problem [35].

Another property, known as the *finite model property* and proved by Palka [28], states that the equational theory of KA can also be characterized by the class of *finite* KAs, denoted $\mathfrak{F}$. Her result can be stated as follows:

Theorem 2.9 (Palka [28]). *For all $e, f \in \mathbb{E}$, we have that $e \equiv f$ iff $\mathfrak{F} \models e = f$.*

In her proof of the above, Palka applied Kozen's theorem. The central contribution of this paper is that *both theorems can be proved independently of each other*, by a generic construction that allows one to conclude either result.

3 Solutions to automata

In this section, we recall *automata* as a way of defining a language, as well as the notion of the *least solution* to an automaton. Both of these are well-known, but since they play such a central role for our results we discuss them in detail.

Definition 3.1 (Automaton). *A (*non-deterministic finite*)* automaton *A is a tuple (Q, δ, I, F) where Q is a finite set of* states*, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the* transition function *and $I \subseteq Q$ (resp. $F \subseteq Q$) holds the* initial *(resp.* final*)* states.

For $\mathsf{a} \in \Sigma$, we write δ_{a} for the relation given by $\{(q, q') : q' \in \delta(q, \mathsf{a})\}$. This family of relations can be extended to words, as follows:

$$\delta_\epsilon = \mathsf{id}_Q \qquad\qquad \delta_{w\mathsf{a}} = \delta_w \circ \delta_{\mathsf{a}}$$

The language *of a state $q \in Q$, denoted $L(A, q)$, is the set of words w such that q can reach a final state through δ_w, given by $L(A, q) = \{w \in \Sigma^* : q \ \delta_w \ q_f \in F\}$. The language of A is defined by its initial states, i.e., $L(A) = \bigcup_{q_i \in I} L(A, q_i)$.*

It is well known that the set of languages defined by regular expressions is the same as the set of languages described by (finite) automata [16]. In fact, the translations that demonstrate this equivalence will play an important role in the remainder of this paper, and we will outline one of these now.

Definition 3.2 (Solutions). *Let $A = (Q, \delta, I, F)$ be an automaton, and $e \in \mathbb{E}$. An* e-solution *to A is a function $s : Q \to \mathbb{E}$ s.t. the following hold for all $q, q' \in Q$:*

$$q \in F \implies e \leqq s(q) \qquad\qquad q' \in \delta(q, \mathsf{a}) \implies \mathsf{a} \cdot s(q') \leqq s(q)$$

A 1-solution to A is simply called a solution *to A. We say that s is the* least e-solution *to A if for all e-solutions s' it holds for all $q \in Q$ that $s(q) \leqq s'(q)$.*

Least e-solutions are unique up to the laws of KA; this explains why we can speak of *the* least e-solution to an automaton.

3.1 Computing solutions

It is well-known that least e-solutions always exist for (finite) automata; the process to compute these [10,17] closely resembles the *state elimination* technique from [16,27], which computes a regular expression representing the language accepted by an automaton.

Theorem 3.3 (Computing solutions). *Let $A = (Q, \delta, I, F)$ be an automaton, and let $e \in \mathbb{E}$. We can compute the least e-solution to A, denoted $\overline{A^e}$.*

In fact, the above statement can be strengthened: as it turns out, the least solution to A gives rise to *all* of the least e-solutions [17], in the following sense.

Theorem 3.4 (Relating solutions). *Let $A = (Q, \delta, I, F)$ be an automaton, and let $e \in \mathbb{E}$. For all $q \in Q$, it holds that $\overline{A^1}(q) \cdot e \equiv \overline{A^e}(q)$.*

The two results above form the technical nexus of this paper, and will be applied repeatedly throughout the coming three sections. The second result in particular, which connects solutions to e-solutions, will prove to be rather useful.

To lighten notation, we will simply write $\overline{A}$ for $\overline{A^1}$, which we call the *least solution to A*. We also write $\lfloor A \rfloor$ for the expression $\sum_{q \in I} \overline{A}(q)$.

3.2 Properties of solutions

We conclude this section by recording three more properties of solutions. For the remainder of this section, we fix two automata $A_i = (Q_i, \delta_i, I_i, F_i)$ for $i \in \{1, 2\}$.

For the first property, we need to define *morphisms* of automata.

Definition 3.5. *A* morphism *from A_1 to A_2 is a function $h : Q_1 \to Q_2$ where (1) if $q \in F_1$ then $h(q) \in F_2$, and (2) if $q' \in \delta_1(q, \mathtt{a})$, then $h(q') \in \delta_2(h(q), \mathtt{a})$. Furthermore, h is* strong *when for all $q \in I_1$ we have that $h(q) \in I_2$.*

Morphisms between automata relate their least solutions, as follows.

Lemma 3.6. *Let $h : Q_1 \to Q_2$ be a morphism from A_1 to A_2. For all $q \in Q$, it holds that $\overline{A_1}(q) \leqq \overline{A_2}(h(q))$. Furthermore, if h is strong, then $\lfloor A_1 \rfloor \leqq \lfloor A_2 \rfloor$.*

For the second property, we need the notion of a subautomaton.

Definition 3.7. *We say A_1 is a* subautomaton *of A_2 when $Q_1 \subseteq Q_2$, and furthermore for all $\mathtt{a} \in \Sigma$ we have that $\delta_1(q, \mathtt{a}) = \delta_2(q, \mathtt{a})$.*

Unsurprisingly, the least solution to a subautomaton coincides with the least solution of the automaton that contains it, on the states where they overlap.

Lemma 3.8. *If A_1 is a subautomaton of A_2 and $q \in Q_1$, then $\overline{A_1}(q) \equiv \overline{A_2}(q)$.*

The third and last property that we will use connects the least solution of an automaton to the languages of that automaton.

Lemma 3.9. *Both of the following hold for all $q \in Q_1$:*

$$\overline{A_1}(q) \equiv [q \in F] + \sum_{q' \in \delta(q, \mathtt{a})} \mathtt{a} \cdot \overline{A_1}(q') \qquad\qquad \widehat{\ell}(\overline{A_1}(q)) = L(A_1, q)$$

Here, $[q \in F]$ is shorthand for 1 if $q \in F$, and 0 otherwise.

4 Transformation automata

Throughout this section, we fix an automaton $A = (Q, \delta, I, F)$.

We now turn our attention to *transformation automata* [26]. Intuitively, the states of a transformation automaton A' obtained from A are relations on Q, with the intention that reading $w \in \Sigma^*$ starting from a state R in A' leads (uniquely) to $R \circ \delta_w$. In particular, reading w in A' from id_Q takes us to δ_w, which is why we will pay special attention to the solutions to id_Q in transformation automata.

Definition 4.1. *We define $\delta^\tau : \mathcal{P}(Q \times Q) \times \Sigma \to \mathcal{P}(\mathcal{P}(Q \times Q))$ by setting*

$$\delta^\tau(R, \mathsf{a}) = \{R \circ \delta_\mathsf{a}\}$$

*For each $R \subseteq Q \times Q$, write $A[R]$ for the R-*transformation automaton

$$(\mathcal{P}(Q \times Q), \delta^\tau, \{\mathsf{id}_Q\}, \{R\})$$

Note that the above still fits our definition of an automaton, since if Q is finite then so is the set of relations on Q, i.e., $\mathcal{P}(Q \times Q)$. It is also useful to point out that transformation automata are *deterministic*, in that each state leads to one (and only one) next state for a given letter.

Remark 4.2. Readers familiar with formal language theory may recognize transformation automata as the construction used to show that each language accepted by an automaton can also be recognized by a (finite) monoid [26].

In the remainder of this section, we characterize the solution to A in terms of solutions to its transformation automata. To this end, we first analyze the solutions to transformation automata in general. A useful first observation is that, for each $\mathsf{a} \in \Sigma$, words read from id_Q to δ_a in the transformation automaton include a. This gives rise to the following property on the level of solutions.

Lemma 4.3. *For all $\mathsf{a} \in \Sigma$, it holds that $\mathsf{a} \leqq \lfloor A[\delta_\mathsf{a}] \rfloor$.*

Furthermore, if R_1, R_2 and R_3 are relations, and if we can read w by moving from R_1 to R_2, then we can also read w by moving from $R_3 \circ R_1$ to $R_3 \circ R_2$. This can be expressed in terms of solutions to transformation automata, as follows.

Lemma 4.4. *For all $R_1, R_2, R_3 \subseteq Q \times Q$, it holds that*

$$\overline{A[R_2]}(R_1) \leqq \overline{A[R_3 \circ R_2]}(R_3 \circ R_1)$$

Proof sketch. Let's fix R_2 and R_3. We choose $s : \mathcal{P}(Q \times Q) \to \mathbb{E}$ by setting $s(R) = \overline{A[R_3 \circ R_2]}(R_3 \circ R)$. Now show that s is a solution to $A[R_2]$. □

We can think of the least solution to id_Q in the R-transformation automaton of A as an expression representing all words w such that $\delta_w = R$. This explains the next property, which is an algebraic encoding of the fact that if $w_1, w_2 \in \Sigma^*$ are such that $\delta_{w_1} = R_1$ and $\delta_{w_2} = R_2$, then $\delta_{w_1 \cdot w_2} = \delta_{w_1} \circ \delta_{w_2} = R_1 \circ R_2$.

Lemma 4.5. *For all $R_1, R_2 \subseteq Q$ it holds that $\lfloor A[R_1] \rfloor \cdot \lfloor A[R_2] \rfloor \leqq \lfloor A[R_1 \circ R_2] \rfloor$*

Proof sketch. Using Lemmas 4.3 and 4.4, one can show that $\overline{A[R_1 \circ R_2]}$ is an $\lfloor A[R_2] \rfloor$-solution to $A[R_1]$, which implies the claim. □

With this property in hand, we can now express the least solution to A in terms of the least solutions to its transformation automata, as follows.

Lemma 4.6. *For all $q \in Q$ it holds that $\overline{A}(q) \equiv \sum_{qRq_f \in F} \lfloor A[R] \rfloor$*

Proof sketch. To show that the left-hand side is contained in the right-hand side, we use the preceding lemmas to show that it constitutes a solution to A. For the converse containment, we argue that the solution of A gives rise to a solution to each of the automata $A[R]$ that appear on the right-hand side. □

5 Antimirov's construction

We now discuss the least solution to an automaton A_e that accepts $\widehat{\ell}(e)$, for each $e \in \mathbb{E}$. Many methods to obtain such an automaton exist (for instance [37,7]; see [38] for a good overview). We focus on Antimirov's construction [2], and show that an expression e can be recovered from its Antimirov automaton.

Remark 5.1. In a sense, the property we prove is analogous to the one shown by Kozen [20] (see also [14]), who proved that e can recovered from the solution to its Brzozowski automaton [7]. We diverge from this for two reasons.

1. Antimirov's construction produces non-deterministic automata, which makes it a bit easier to express than Brzozowski's construction, which uses *deterministic* automata. In particular, this saves us from having to consider the theory necessary to make Brzozowski's construction produce a *finite* automaton.
2. Kozen's result about Brzozowski's construction leverages the fact that *bisimilar automata have equivalent solutions.* This is a very powerful (and somewhat tricky to prove) observation, which also underlies the completeness proof in [20,14]. For Antimirov automata, however, it turns out that we can rely on *morphisms* of automata instead, which are fairly easy to establish.[4]

Having said that, the structure of the proof that follows is very much inspired by the strategy employed in [20,14], especially when it comes to Lemma 5.15.

5.1 Recalling Antimirov's automata

The main idea behind Antimirov's construction is that expressions are endowed with the structure of an automaton. The language of a state in this automaton is meant to be $\widehat{\ell}(e)$, the language denoted by its expression. From this perspective, the accepting states should be those representing expressions whose language contains the empty word. This set of expressions is fairly easy to describe.

Definition 5.2. *The set $\mathbb{F}$ is defined as the smallest subset of $\mathbb{E}$ satisfying:*

$$\frac{}{1 \in \mathbb{F}} \qquad \frac{e_1 \in \mathbb{F} \qquad e_2 \in \mathbb{E}}{e_1 + e_2, e_2 + e_1 \in \mathbb{F}} \qquad \frac{e_1, e_2 \in \mathbb{F}}{e_1 \cdot e_2 \in \mathbb{F}} \qquad \frac{e_1 \in \mathbb{E}}{e_1^* \in \mathbb{F}}$$

[4] Kozen's bisimilarity property would also not help us obtain a completeness proof using Antimirov automata instead of Brzozowski automata, because some non-deterministic automata are not bisimilar despite having the same language; see also Remark 5.13.

Next, we recall Antimirov's transition function. The intuition is that an expression e has an $\mathtt{a}$-transition to an expression e' when e' denotes remainders of words in $\widehat{\ell}(e)$ that start with $\mathtt{a}$ — i.e., if $w \in \widehat{\ell}(e')$, then $\mathtt{a}w \in \widehat{\ell}(e)$. Together, the expressions reachable by $\mathtt{a}$-transitions from e should describe *all* such words.

In the following, when $S \subseteq \mathbb{E}$ and $e \in \mathbb{E}$, we write $S \fatsemi e$ for $\{e' \cdot e : e' \in S\}$. We are now ready to define Antimirov's transition function, as follows.

Definition 5.3. *We define $\partial : \mathbb{E} \times \Sigma \to \mathcal{P}(\mathbb{E})$ recursively, as follows*

$$\begin{aligned} \partial(0, \mathtt{a}) &= \emptyset & \partial(e_1 + e_2, \mathtt{a}) &= \partial(e_1, \mathtt{a}) \cup \partial(e_2, \mathtt{b}) \\ \partial(1, \mathtt{a}) &= \emptyset & \partial(e_1 \cdot e_2, \mathtt{a}) &= \partial(e_1, \mathtt{a}) \fatsemi e_2 \cup e_1 \star \partial(e_2, \mathtt{a}) \\ \partial(\mathtt{b}, \mathtt{a}) &= \{1 : \mathtt{a} = \mathtt{b}\} & \partial(e_1^*, \mathtt{a}) &= \partial(e_1, \mathtt{a}) \fatsemi e_1^* \end{aligned}$$

Here, we use $e \star S$ as a shorthand for S when $e \in \mathbb{F}$, and $\emptyset$ otherwise.

Of course, the expression e could serve as the sole initial state in the automaton for e. However, our automata allow multiple initial states, and distributing this task among them will simplify some of the arguments that follow.

Definition 5.4. *We define $\iota : \mathbb{E} \to \mathcal{P}(\mathbb{E})$ recursively, as follows:*

$$\begin{aligned} \iota(0) &= \emptyset & \iota(e_1 + e_2) &= \iota(e_1) \cup \iota(e_2) \\ \iota(1) &= \{1\} & \iota(e_1 \cdot e_2) &= \iota(e_1) \fatsemi e_2 \\ \iota(\mathtt{a}) &= \{\mathtt{a}\} & \iota(e_1^*) &= \iota(e_1) \fatsemi e_1^* \cup \{1\} \end{aligned}$$

We could now try to package these parts into an automaton $(\mathbb{E}, \partial, \iota(e), \mathbb{F})$ for each expression e. Unfortunately, we have defined our automata to be finite, so a little more work is necessary to identify the expressions that are relevant (i.e., represented by reachable states) for a starting expression e.

Definition 5.5. *We define $\rho : \mathbb{E} \to \mathcal{P}(\mathbb{E})$ recursively, as follows:*

$$\begin{aligned} \rho(0) &= \emptyset & \rho(e_1 + e_2) &= \rho(e_1) \cup \rho(e_2) \\ \rho(1) &= \{1\} & \rho(e_1 \cdot e_2) &= \rho(e_1) \fatsemi e_2 \cup \rho(e_2) \\ \rho(\mathtt{a}) &= \{\mathtt{a}, 1\} & \rho(e_1^*) &= \rho(e_1) \fatsemi e_1^* \cup \{1\} \end{aligned}$$

With this function in hand, we can verify that it fits all of the requirements of the state space of an automaton with respect to the other parts identified above.

Lemma 5.6. *For all $e \in \mathbb{E}$, the set $\rho(e)$ is finite and closed under ∂, i.e., if $e' \in \rho(e)$ and $e'' \in \partial(e', \mathtt{a})$, then $e'' \in \rho(e)$ as well. Furthermore, $\iota(e) \subseteq \rho(e)$.*

In light of this, we write ∂_e for the function $\partial_e : \rho(e) \times \Sigma \to \mathcal{P}(\rho(e))$ obtained by restricting ∂. We can now define Antimirov automata, as follows.

Definition 5.7. *Let $e \in \mathbb{E}$. We write A_e for the* Antimirov automaton

$$(\rho(e), \partial_e, \iota(e), \mathbb{F} \cap \rho(e))$$

Antimirov's transition function can be used to decompose an expression e into several "derivatives" e', which can then reconstitute e. This validates the intuition that the derivatives collectively contain (only) the "tails" of words denoted by e. Similarly, the initial expressions $\iota(e)$ can also be used to reconstitute e.

Theorem 5.8. *Let $e \in \mathbb{E}$. The following two equivalences hold:*

$$e \equiv [e \in \mathbb{F}] + \sum_{e' \in \partial(e, \mathtt{a})} \mathtt{a} \cdot e' \qquad\qquad e \equiv \sum_{e' \in \iota(e)} e'$$

The first property above is usually referred to as the *fundamental theorem* of Kleene algebra [30], because of its close resemblance to the fundamental theorem of calculus. One caveat is that one needs to prove that the sums on the right-hand sides are in fact finite, but this turns out to be the case.

We end this subsection by recording two more useful properties of ι.

Lemma 5.9. *Let $e \in \mathbb{E}$. We have $e \in \mathbb{F}$ if and only if there exists an $e' \in \iota(e)$ such that $e' \in \mathbb{F}$. Also, $e'' \in \partial(e, \mathtt{a})$ if and only if $e'' \in \partial(e', \mathtt{a})$ for some $e' \in \iota(e)$.*

5.2 Solving Antimirov's automata

Having fully described Antimirov's construction, we resume with the proof of the main technical point of this section, which is that the solution to A_e can be used to construct an expression equivalent to e.

More precisely, we will prove that e is equivalent to the sum of the solutions to its initial states, $\lfloor A_e \rfloor$, by showing that $\lfloor A_e \rfloor \leqq e$ and $e \leqq \lfloor A_e \rfloor$. The former property is easy to prove using the theory established up to this point.

Lemma 5.10. *For all $e \in \mathbb{E}$, it holds that $\lfloor A_e \rfloor \leqq e$.*

Proof sketch. By Theorem 5.8, the injection of $\rho(e)$ into $\mathbb{E}$ is a solution to A_e. □

To show that $e \leqq \lfloor A_e \rfloor$, we cannot exploit the fact that $\overline{A_e}$ is the least solution to A_e, as above. Our proof will instead operate by induction on e. First, we will need to develop some theory; the following abstraction is useful.

Definition 5.11. *Let $e, f \in \mathbb{E}$. We write $e \lesssim f$ when there exists a strong automaton morphism $h : \rho(e) \to \rho(f)$ from A_e to A_f.*

By Lemma 3.6, if we want to show that $\lfloor A_e \rfloor \leqq \lfloor A_f \rfloor$, it is sufficient to prove $e \lesssim f$. We record the following instances of expressions being related by $\lesssim$.

Lemma 5.12. *The following hold for all $e_0, e_1, e_2 \in \mathbb{E}$:*

$$e_0 \lesssim e_0 \cdot 1 \qquad e_0 \lesssim e_0 + e_1 \qquad e_0 \lesssim e_1 \implies e_0 \cdot e_2 \lesssim e_1 \cdot e_2$$

$$e_0 \cdot e_0^* \lesssim e_0^* \qquad 1 \lesssim e_0^* \qquad e_0 \cdot (e_1 \cdot e_2) \lesssim (e_0 \cdot e_1) \cdot e_2$$

Proof sketch. In all cases, a map can be gleaned from the structure of the relevant state spaces; checking that it is a term morphism is routine. □

Remark 5.13. Kozen [20] and Jacobs [14] show that if $e, f \in \mathbb{E}$ are such that $e \leqq f$, then the Brzozowski automaton of e is simulated by that of f, and hence these automata yield solutions e' and f' such that $e' \leqq f'$.

It is tempting to try and prove a similar property for Antimirov automata, along the lines of "if $e \leqq f$, then $e \lesssim f$". Unfortunately, this is not true. For instance, if $e = \mathtt{a} \cdot (\mathtt{b} + \mathtt{c})$ and $f = \mathtt{a} \cdot \mathtt{b} + \mathtt{a} \cdot \mathtt{c}$, then $e \leqq f$, but there is no strong morphism from A_e to A_f. Fortunately, Lemma 5.12 is sufficient for our purposes.

The solutions to the automata for e, $e \cdot 1$ and $1 \cdot e$ are also related.

Lemma 5.14. *Let $e \in \mathbb{E}$. It holds that $\lfloor A_{e \cdot 1} \rfloor \leqq \lfloor A_e \rfloor$ and $\lfloor A_e \rfloor \leqq \lfloor A_{1 \cdot e} \rfloor$.*

Proof sketch. We show that the solution to one automaton gives rise to a solution to the other automaton, using Lemmas 5.9 and 3.9 for the latter claim. □

The next lemma is the main workhorse that we need to show that $e \leqq \lfloor A_e \rfloor$. The proof is very similar to that of [20, Lemma 3].

Lemma 5.15. *Let $e, f \in \mathbb{E}$. It holds that $e \cdot \lfloor A_f \rfloor \leqq \lfloor A_{e \cdot f} \rfloor$*

Proof sketch. As in [20, Lemma 3], we proceed by induction on e; we use Lemmas 3.6 and 5.14 in the base, and Lemma 5.12 in the inductive cases. □

With this in hand, we now have everything required to conclude the desired property of solutions to Antimirov automata, which we record below.

Lemma 5.16. *For all $e \in \mathbb{E}$, we have that $e \equiv \lfloor A_e \rfloor$.*

Proof. We already knew that $\lfloor A_e \rfloor \leqq e$ by Lemma 5.10. To show that $e \leqq \lfloor A_e \rfloor$, we derive using Lemmas 5.14 and 5.15, as follows:

$$e \equiv e \cdot 1 \leqq e \cdot \lfloor A_1 \rfloor \leqq \lfloor A_{e \cdot 1} \rfloor \leqq \lfloor A_e \rfloor$$

The second step is valid because $1 \in \iota(1)$ and $1 \in \mathbb{F}$, so $1 \leqq \overline{A_1}(1) \leq \lfloor A_1 \rfloor$. □

6 From monoids to Kleene algebras

Recall that our objective was to derive a finite KA for two expressions, whose properties can then be used to conclude completeness and the FMP. We already saw how an expression gives rise to an automaton, which can then be turned into a transformation automaton. As it happens, the states of this transformation automaton have the internal structure of a monoid — indeed, this was the original motivation for the construction [26] — but we still do not have a KA.

In this section, we recall a straightforward translation from monoids to KAs proposed by Palka [28], and prove a useful property that we will leverage in the proof later on. Let us start by recalling the definition of a monoid.

Definition 6.1. *A* monoid *is a tuple $(M, \cdot, 1)$ where M is a set, $\cdot$ is a binary operator and $1 \in M$ such that the following hold for all $m_0, m_1, m_2 \in M$:*

$$m_1 \cdot (m_2 \cdot m_3) = (m_1 \cdot m_2) \cdot m_3 \qquad m_1 \cdot 1 = m_1 \qquad 1 \cdot m_1 = m_1$$

A function $h : \Sigma \to M$ gives rise to the function $\widetilde{h} : \Sigma^* \to M$, defined by

$$\widetilde{h}(\mathtt{a}_1 \cdots \mathtt{a}_n) = h(\mathtt{a}_1) \cdot \cdots \cdot h(\mathtt{a}_\mathtt{n})$$

As for KAs, we may identify a monoid $(M, \cdot, 1)$ with its carrier M, if the accompanying operator and unit are clear from context.

As stated above, if $A = (Q, \delta, I, F)$ is an automaton, then the state space of its transition automata is given by $\mathcal{P}(Q \times Q)$ — i.e., the relations on Q — which has a monoidal structure: the operator is given by relational composition, and the unit is the identity relation on Q. In the sequel, we write M_A for this monoid.

The composition operator of a monoid can be lifted to sets of its elements, which can then be used to derive a fixed point operator, as follows.

Lemma 6.2 (Palka [28]). *If $(M, \cdot, 1)$ is a monoid, then $(\mathcal{P}(M), \cup, \otimes, ^{\circledast}, \emptyset, \{1\})$ is a KA, where $\otimes$ and $^{\circledast}$ are defined by choosing for $U, V \subseteq M$:*

$$U \otimes V = \{m \cdot n : m \in U, n \in V\} \qquad U^{\circledast} = \{u_1 \cdots u_n : u_1, \dots, u_n \in U\}$$

As an example of this construction, note that applying this construction to the free monoid $(\Sigma^*, \cdot, \epsilon)$ precisely yields the free KA of languages.

Now, given an expression $e \in \mathbb{E}$, a monoid $(M, \cdot, 1)$, and a map $h : \Sigma \to M$, we have two ways of interpreting e inside of the KA that arises from this monoid:

1. We lift the map $\mathtt{a} \mapsto \{h(\mathtt{a})\}$ to obtain a map $\mathbb{E} \to \mathcal{P}(M)$.
2. We map each $w \in \widehat{\ell}(e)$ to an element of M via $\widetilde{h} : \Sigma \to M$.

The next lemma shows that these two interpretations of expressions inside the KA for $(M, \cdot, 1)$ are actually the same; it can be thought of as a generalization of [28, Lemma 3.1], which covers the special case for the syntactic monoid.

Lemma 6.3. *Let $(M, \cdot, 1)$ be a monoid and let $(\mathcal{P}(M), \cup, \otimes, ^{\circledast}, \emptyset, \{1\})$ be the KA obtained from it, per Lemma 6.2. Furthermore, let $h_1 : \Sigma \to M$ and $h_2 : \Sigma \to \mathcal{P}(M)$ be such that for all $\mathtt{a} \in \Sigma$, we have that $h_2(\mathtt{a}) = \{h_1(\mathtt{a})\}$. Then for $e \in \mathbb{E}$:*

$$\widehat{h_2}(e) = \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e)\}$$

We conclude this section by leveraging the above to prove a pivotal lemma: the solution to a state q of an automaton A can be recovered by interpreting this expression inside of the KA obtained from the transformation automata of A, and looking at the solutions to the relations inside that interpretation.

Lemma 6.4. *Let $A = (Q, \delta, I, F)$ be an automaton and $q \in Q$. Furthermore, let $h : \Sigma \to \mathcal{P}(M_A)$ be given by $h(\mathtt{a}) = \{\delta_{\mathtt{a}}\}$. The following holds.*

$$\overline{A}(q) \equiv \sum_{R \in \widehat{h}(\overline{A}(q))} \lfloor A[R] \rfloor$$

Proof. We start by massaging the proof goal. Lemma 6.3 tells us that $R \in \widehat{h}(\overline{A}(q))$ if and only if there exists a $w \in \widehat{\ell}(\overline{A}(q))$ such that $R = \delta_w$. Using this observation and Lemmas 4.6 and 3.9, it suffices to show

$$\sum_{q R q' \in F} \lfloor A[R] \rfloor \equiv \sum_{w \in L(A,q)} \lfloor A[\delta_w] \rfloor$$

For the inclusion from left to right, let $R \subseteq Q \times Q$ and $q' \in F$ be such that $q \; R \; q'$. On the one hand, if $\widehat{\ell}(\lfloor A[R] \rfloor) = \emptyset$, then an easy inductive argument shows that $\lfloor A[R] \rfloor \equiv 0$, which means that the term $\lfloor A[R] \rfloor$ does not contribute to the sum. Otherwise, let $w \in \widehat{\ell}(\lfloor A[R] \rfloor)$. By Lemma 3.9, we know that $w \in L(A[R], \mathsf{id}_Q)$, and thus $\delta_w = R$. Since $q \; \delta_w \; q' \in F$, we also know that $w \in L(\overline{A}(q))$. Therefore $\lfloor A[R] \rfloor = \lfloor A[\delta_w] \rfloor$ appears in the sum on the right-hand side.

For the inclusion from right to left, let $w \in L(A, q)$. In that case, $q \delta_w q'$ for some $q' \in F$. Thus $\lfloor A[\delta_w] \rfloor$ appears in the sum on the left-hand side. $\square$

7 Completeness and the FMP

We are now ready to prove our main claims. In a nutshell, our proof will take two expressions e and f, apply the transformation automaton construction to A_{e+f}, and then use the resulting state space monoid to obtain a KA. The previously derived facts connect e and f to their interpretation inside this KA, which we will use in two different ways to conclude both completeness and the FMP.

Throughout this section, we fix $e, f \in \mathbb{E}$. For brevity, we also write ∂ for the transition function of A_{e+f}. We fix $h : \Sigma \to \mathcal{P}(M_{A_{e+f}})$ by $h(\mathtt{a}) = \{\partial_{\mathtt{a}}\}$. Note that $M_{A_{e+f}}$ is a finite monoid, and hence $\mathcal{P}(M_{A_{e+f}})$ is a finite KA by Lemma 6.2.

The next lemma puts the results of the previous sections together to connect e and f to their interpretations inside $\mathcal{P}(M_{A_{e+f}})$, in the following way.

Lemma 7.1. *The following two equivalences hold:*

$$e \equiv \sum_{R \in \widehat{h}(e)} \lfloor A_{e+f}[R] \rfloor \qquad\qquad f \equiv \sum_{R \in \widehat{h}(f)} \lfloor A_{e+f}[R] \rfloor$$

Proof. Without loss of generality, we prove the first property by deriving:

$$\begin{aligned}
e &\equiv \lfloor A_e \rfloor && \text{(Lemma 5.16)} \\
&\equiv \sum_{e' \in \iota(e)} \overline{A_e}(e') && \text{(def. } \lfloor A_e \rfloor) \\
&\equiv \sum_{e' \in \iota(e)} \overline{A_{e+f}}(e') && \text{(Lemma 3.8)} \\
&\equiv \sum_{e' \in \iota(e)} \sum_{R \in \widehat{h}(\overline{A_{e+f}}(e'))} \lfloor A_{e+f}[R] \rfloor && \text{(Lemma 6.4)} \\
&\equiv \sum_{R \in \widehat{h}(e)} \lfloor A_{e+f}[R] \rfloor && \text{(see below)}
\end{aligned}$$

The last equivalence holds because by Lemmas 3.8 and 5.16, we have that:

$$e \equiv \lfloor A_e \rfloor \equiv \sum_{e' \in \iota(e)} \overline{A_e}(e') \equiv \sum_{e' \in \iota(e)} \overline{A_{e+f}}(e')$$

and thus $\widehat{h}(e) = \bigcup_{e' \in \iota(e)} \widehat{h}(\overline{A_{e+f}}(e'))$ by Lemma 2.5. □

We are now ready to conclude our first main claim: the finite model property holds for KA. Recall that $\mathfrak{F}$ denotes the class of all *finite* Kleene algebras, to which $\mathcal{P}(M_{A_{e+f}})$ belongs. This allows us to apply Lemma 7.1, as follows.

Theorem 7.2 (Finite model property). *If $\mathfrak{F} \models e = f$, then $e \equiv f$.*

Proof. By the premise, we have that $\widehat{h}(e) = \widehat{h}(f)$, and so by Lemmas 7.1 and 2.5:

$$e \equiv \sum_{R \in \widehat{h}(e)} \lfloor A_{e+f}[R] \rfloor = \sum_{R \in \widehat{h}(f)} \lfloor A_{e+f}[R] \rfloor \equiv f$$

□

Finally, we note that a very similar proof also allows us to conclude that the axioms of KA are complete w.r.t. its language model, thanks to the connection between interpretations in lifted monoids given by Lemma 6.3.

Theorem 7.3 (Completeness). *If $\mathcal{L} \models e = f$, then $e \equiv f$.*

Proof. Let $h' : \Sigma \to M_{A_{e+f}}$ be given by $h'(\mathtt{a}) = \partial_{\mathtt{a}}$. By the premise $\widehat{\ell}(e) = \widehat{\ell}(f)$, and thus $\widehat{h}(e) = \widehat{h}(f)$ because we can use Lemma 6.3 to derive

$$\widehat{h}(e) = \{\widetilde{h'}(w) : w \in \widehat{\ell}(e)\} = \{\widetilde{h'}(w) : w \in \widehat{\ell}(f)\} = \widehat{h}(f)$$

We can then conclude by leveraging Lemma 7.1, as for Theorem 7.2. □

8 Discussion

We leave the reader with some final considerations regarding our formalization and directions for possible further work.

Coq formalization We have formalized all of our results in Coq [4,11]. The trusted base comes down to (1) the axioms of the Calculus of Inductive Constructions, (2) injectivity of dependent equality (equivalent to Streicher's axiom K [36]), and (3) dependent functional extensionality. The latter is a result of our encoding of subsets, and can most likely be factored out with better data structures.

All proofs as presented here are faithful to the insights underlying the claim, although some encodings differ slightly. For instance, the definition of $\rho(e)$ in the development is more accurately rendered using disjoint union.

Possible extensions Guarded Kleene Algebra with Tests (GKAT) [34,32] is a fragment of KAT with favorable decidability properties. GKAT in particular admits a set of axioms that are complete w.r.t. its language (resp. relational, probabilistic) model, but this set is infinite as a result of an axiom scheme. We wonder whether the techniques discussed here could be applied to arrive at a more satisfactory completeness result. To start answering this question, one would first have to devise an analogue transformation automata and monoids for GKAT.

Relational models Pratt [29] connected the language model of KA to the relational model — essentially saying that if $\mathfrak{R}$ is the class of relational KAs (as in Example 2.2), then $\mathfrak{R} \models e = f$ if and only if $\mathcal{L} \models e = f$ for all $e, f \in \mathbb{E}$. By Theorem 7.3, this means that relational models are also complete for KA.

In light of Theorem 7.2, we wonder: can this form of completeness be strengthened to *finite* relational models? A positive answer would mean that the finite countermodel accompanying an invalid equation would correspond to an interpretation of the primitive actions as state transformers on a finite state space.

There is a tantalizing candidate for a canonical model that might be able to fill the role of $\mathcal{P}(M_{A_{e+f}})$ in the previous section: simply use the relational KA with the carrier $M_{A_{e+f}}$. For this to work, we would have to connect e and f with their interpretations inside this KA, which will require further research.

Acknowledgements The author wishes to thank Nick Bezhanishvili for his suggestion to investigate the FMP for KA, Alexandra Silva for general advice about this manuscript, and the anonymous reviewers for their careful comments.

References

1. Anderson, C.J., Foster, N., Guha, A., Jeannin, J., Kozen, D., Schlesinger, C., Walker, D.: NetKAT: semantic foundations for networks. In: POPL. pp. 113–126 (2014). https://doi.org/10.1145/2535838.2535862
2. 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
3. Backhouse, R.: Closure algorithms and the star-height problem of regular languages. Ph.D. thesis, University of London (1975), http://hdl.handle.net/10044/1/22243
4. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. An EATCS Series (2004). https://doi.org/10.1007/978-3-662-07964-5
5. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic, Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001). https://doi.org/10.1017/CBO9781107050884
6. Boffa, M.: Une remarque sur les systèmes complets d'identités rationnelles. RAIRO Theor. Informatics Appl. **24**, 419–423 (1990). https://doi.org/10.1051/ita/1990240404191
7. Brzozowski, J.A.: Derivatives of regular expressions. J. ACM **11**(4), 481–494 (1964). https://doi.org/10.1145/321239.321249
8. Burris, S., Sankappanavar, H.P..: A Course in Universal Algebra. Graduate Texts in Mathematics, Springer (1981)
9. Cohen, E., Kozen, D., Smith, F.: The complexity of Kleene algebra with tests. Tech. Rep. TR96-1598 (July 1996), https://hdl.handle.net/1813/7253
10. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall, Ltd., London (1971)
11. Coq Development Team: The Coq Reference Manual, version 8.15 (2022), available electronically at http://coq.inria.fr/doc
12. Das, A., Doumane, A., Pous, D.: Left-handed completeness for Kleene algebra, via cyclic proofs. In: LPAR. pp. 271–289 (2018). https://doi.org/10.29007/hzq3
13. 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
14. Jacobs, B.: A bialgebraic review of deterministic automata, regular expressions and languages. In: Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday. pp. 375–404 (2006). https://doi.org/10.1007/11780274_20
15. Kappé, T.: Completeness and the finite model property for Kleene algebra, reconsidered — Coq formalization (2022). https://doi.org/10.5281/zenodo.7467245
16. Kleene, S.C.: Representation of events in nerve nets and finite automata. Automata Studies pp. 3–41 (1956)
17. 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

18. Kozen, D.: Kleene algebra with tests and commutativity conditions. In: TACAS. pp. 14–33 (1996). https://doi.org/10.1007/3-540-61042-1_35
19. Kozen, D.: Kleene algebra with tests. ACM Trans. Program. Lang. Syst. **19**(3), 427–443 (1997). https://doi.org/10.1145/256167.256195
20. Kozen, D.: Myhill-Nerode relations on automatic systems and the completeness of Kleene algebra. In: STACS. pp. 27–38 (2001). https://doi.org/10.1007/3-540-44693-1_3
21. Kozen, D., Patron, M.: Certification of compiler optimizations using Kleene algebra with tests. In: CL. pp. 568–582 (2000). https://doi.org/10.1007/3-540-44957-4_38
22. Kozen, D., Silva, A.: Left-handed completeness. Theor. Comput. Sci. **807**, 220–233 (2020). https://doi.org/10.1016/j.tcs.2019.10.040
23. Kozen, D., Smith, F.: Kleene algebra with tests: Completeness and decidability. In: CSL. pp. 244–259 (1996). https://doi.org/10.1007/3-540-63172-0_43
24. Kozen, D., Tseng, W.D.: The Böhm-Jacopini theorem is false, propositionally. In: MPC. pp. 177–192 (2008). https://doi.org/10.1007/978-3-540-70594-9_11
25. Krob, D.: A complete system of B-rational identities. In: ICALP. pp. 60–73 (1990). https://doi.org/10.1007/BFb0032022
26. McNaughton, R., Papert, S.: The syntactic monoid of a regular event. Algebraic Theory of Machines, Languages, and Semigroups pp. 297–312 (1968)
27. McNaughton, R., Yamada, H.: Regular expressions and state graphs for automata. IRE Trans. Electronic Computers **9**(1), 39–47 (1960). https://doi.org/10.1109/TEC.1960.5221603
28. Palka, E.: On finite model property of the equational theory of Kleene algebras. Fundam. Informaticae **68**(3), 221–230 (2005), http://content.iospress.com/articles/fundamenta-informaticae/fi68-3-02
29. Pratt, V.R.: Dynamic algebras and the nature of induction. In: STOC. pp. 22–28 (1980). https://doi.org/10.1145/800141.804649
30. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theor. Comput. Sci. **249**(1), 3–80 (2000). https://doi.org/10.1016/S0304-3975(00)00056-6
31. 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
32. Schmid, T., Kappé, T., Kozen, D., Silva, A.: Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. In: ICALP. pp. 142:1–142:14 (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.142
33. Smolka, S., Eliopoulos, S.A., Foster, N., Guha, A.: A fast compiler for NetKAT. In: ICFP. pp. 328–341 (2015). https://doi.org/10.1145/2784731.2784761
34. Smolka, S., Foster, N., Hsu, J., Kappé, T., Kozen, D., Silva, A.: Guarded Kleene algebra with tests: Verification of uninterpreted programs in nearly linear time. In: POPL (2020). https://doi.org/10.1145/3371129
35. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time: Preliminary report. In: STOC. pp. 1–9 (1973). https://doi.org/10.1145/800125.804029
36. Streicher, T.: Investigations into intensional type theory. Habilitiation Thesis, Ludwig Maximilian Universität (1993), https://www2.mathematik.tu-darmstadt.de/~streicher/HabilStreicher.pdf
37. Thompson, K.: Regular expression search algorithm. Commun. ACM **11**(6), 419–422 (1968). https://doi.org/10.1145/363347.363387
38. Watson, B.W.: A taxonomy of finite automata construction algorithms. Tech. rep., Technische Universiteit Eindhoven (1993), https://research.tue.nl/files/2482472/9313452

A Proofs for Section 3

Theorem 3.3 (Computing solutions). *Let $A = (Q, \delta, I, F)$ be an automaton, and let $e \in \mathbb{E}$. We can compute the least e-solution to A, denoted $\overline{A^e}$.*

Proof sketch. Given a matrix $M : Q \times Q \to \mathbb{E}$ and a vector $b : Q \to \mathbb{E}$, we say that $s : Q \to \mathbb{E}$ is a *solution* to (M, b) when for all $q, q' \in Q$, we have that

$$b(q) \leqq s(q) \qquad\qquad M(q, q') \cdot s(q') \leqq s(q)$$

We say a s is the *least* solution to (M, b) if for all solutions s' to (M, b) and $q \in Q$ it holds that $s(q) \leqq s'(q)$. Kozen [17] showed that one can always compute the least solution to (M, b), provided that Q is finite. We choose M and b as follows:

$$b(q) = \begin{cases} e & q \in F \\ 0 & q \notin F \end{cases} \qquad\qquad M(q, q') = \sum_{q' \in \delta(q, \mathtt{a})} \mathtt{a}$$

Now the least solution to (M, b) is in fact the least e-solution to A. □

Theorem 3.4 (Relating solutions). *Let $A = (Q, \delta, I, F)$ be an automaton, and let $e \in \mathbb{E}$. For all $q \in Q$, it holds that $\overline{A^1}(q) \cdot e \equiv \overline{A^e}(q)$.*

Proof sketch. It is not so hard to show that $\overline{A^e}(q) \leqq \overline{A^1}(q) \cdot e$ for all $q \in Q$. For the other equivalence, however, one needs to generalize the notion of solution introduced in the proof sketch for Theorem 3.3 to e-solutions. Kozen's result that matrices over a KA again form a KA [17] implies an analogous result to the claim about e-solutions to matrices, which can then be leveraged in the same way. □

Lemma 3.6. *Let $h : Q_1 \to Q_2$ be a morphism from A_1 to A_2. For all $q \in Q$, it holds that $\overline{A_1}(q) \leqq \overline{A_2}(h(q))$. Furthermore, if h is strong, then $\lfloor A_1 \rfloor \leqq \lfloor A_2 \rfloor$.*

Proof sketch. For the first claim, it suffices to prove that $\overline{A_2} \circ h$ is a solution to A_1. To this end, note that if $q \in F_1$, then $h(q) \in F_2$, and so $1 \leqq \overline{A_2}(h(q))$. Furthermore, if $q' \in \delta_1(q, \mathtt{a})$, then $h(q) \in \delta_2(h(q), \mathtt{a})$; thus, $\mathtt{a} \cdot \overline{A_2}(h(q')) \leqq \overline{A_2}(h(q))$.

For the second claim, we derive using the above that

$$\lfloor A_1 \rfloor \equiv \sum_{q \in F_1} \overline{A_1}(q) \leqq \sum_{q \in F_1} \overline{A_1}(h(q)) \leqq \sum_{q \in F_2} \overline{A_2}(q) \equiv \lfloor A_2 \rfloor \qquad □$$

Lemma 3.8. *If A_1 is a subautomaton of A_2 and $q \in Q_1$, then $\overline{A_1}(q) \equiv \overline{A_2}(q)$.*

Proof sketch. To show that $\overline{A_1}(q) \leqq \overline{A_2}(q)$, it suffices to note that the injection from Q_1 into Q_2 is an automaton morphism from A_1 to A_2, and apply Lemma 3.6.

For the other direction, one shows that $s : Q_2 \to \mathbb{E}$ given by $s(q) = \overline{A_1}(q)$ when $q \in Q_1$ and $s(q) = \overline{A_2}(q)$ when $q \in Q_2$ is a solution to A_2. □

Lemma 3.9. *Both of the following hold for all $q \in Q_1$:*

$$\overline{A_1}(q) \equiv [q \in F] + \sum_{q' \in \delta(q, \mathtt{a})} \mathtt{a} \cdot \overline{A_1}(q') \qquad\qquad \widehat{\ell}(\overline{A_1}(q)) = L(A_1, q)$$

Here, $[q \in F]$ is shorthand for 1 if $q \in F$, and 0 otherwise.

Proof sketch. For the first claim, we choose $s : Q \to \mathbb{E}$ by defining $s(q)$ as the expression on the right-hand side for each $q \in Q$. Using the fact that $\overline{A}(q)$ is a solution to A, it is straightforward to show that $s(q) \leqq \overline{A}(q)$ for all $q \in Q$. A standard argument from fixpoint theory also shows that s is a solution to A; since $\overline{A}$ is the least solution, it then follows that $\overline{A}(q) \leqq s(q)$ for all $q \in Q$.

The second claim follows by induction on the length of words; we use the first fact, which in particular tells us that $\widehat{\ell}(\overline{A}(q)) = \widehat{\ell}(s(q))$ for all $q \in Q$. □

B Proofs for Section 4

Lemma 4.3. *For all* $\mathsf{a} \in \Sigma$*, it holds that* $\mathsf{a} \leqq \lfloor A[\delta_\mathsf{a}] \rfloor$.

Proof. Using the fact that $\delta_\mathsf{a} \in \delta^\tau(\mathsf{id}_Q, \mathsf{a})$ and $\overline{A[\delta_\mathsf{a}]}$ solves $A[\delta_\mathsf{a}]$, we derive:

$$\mathsf{a} \equiv \mathsf{a} \cdot 1 \leqq \mathsf{a} \cdot \overline{A[\delta_\mathsf{a}]}(\delta_\mathsf{a}) \leqq \overline{A[\delta_\mathsf{a}]}(\mathsf{id}_Q) \equiv \lfloor A[\delta_\mathsf{a}] \rfloor$$ □

Lemma 4.4. *For all* $R_1, R_2, R_3 \subseteq Q \times Q$*, it holds that*

$$\overline{A[R_2]}(R_1) \leqq \overline{A[R_3 \circ R_2]}(R_3 \circ R_1)$$

Proof. Let's fix R_2 and R_3. We choose $s : \mathcal{P}(Q \times Q) \to \mathbb{E}$ by setting

$$s(R) = \overline{A[R_3 \circ R_2]}(R_3 \circ R)$$

It suffices to prove that s is a solution to $A[R_2]$.

- If $R = R_2$, then $1 \leqq \overline{A[R_3 \circ R_2]}(R_3 \circ R_2) = s(R_2)$.
- If $R' \in \delta^\tau(R, \mathsf{a})$, then $R' = R \circ \delta_\mathsf{a}$, which means that $R_3 \circ R' = R_3 \circ R \circ \delta_\mathsf{a}$, hence $R_3 \circ R' \in \delta^\tau(R_3 \circ R, \mathsf{a})$. This then allows us to derive as follows:

$$\mathsf{a} \cdot s(R') \equiv \mathsf{a} \cdot \overline{A[R_3 \circ R_2]}(R_3 \circ R') \leqq \overline{A[R_3 \circ R_2]}(R_3 \circ R) = s(R)$$ □

Lemma 4.5. *For all* $R_1, R_2 \subseteq Q$ *it holds that* $\lfloor A[R_1] \rfloor \cdot \lfloor A[R_2] \rfloor \leqq \lfloor A[R_1 \circ R_2] \rfloor$

Proof. It suffices to show that $\overline{A[R_1 \circ R_2]}$ is an $\lfloor A[R_2] \rfloor$-solution to $A[R_1]$.

- If $R = R_1$, then $\overline{A[R_2]}(\mathsf{id}_Q) \leqq \overline{A[R_1 \circ R_2]}(R_1)$ by Lemma 4.4.
- If $R' \in \delta^\tau(R, \mathsf{a})$, then we need to show that

$$\mathsf{a} \cdot \overline{A[R_1 \circ R_2]}(R') \leqq \overline{A[R_1 \circ R_2]}(R)$$

This is immediately true because $\overline{A[R_1 \circ R_2]}$ solves $A[R_1 \circ R_2]$. □

Lemma 4.6. *For all* $q \in Q$ *it holds that* $\overline{A}(q) \equiv \sum_{qRq_f \in F} \lfloor A[R] \rfloor$

Proof. Let $s : Q \to \mathbb{E}$ be given by defining $s(q)$ as the right-hand side of the claimed equivalence. To show that $\overline{A}(q) \leqq s(q)$ for all $q \in Q$, it suffices to show that s is a solution to A. We check the conditions as follows.

- Suppose $q \in F$. Since id_Q is accepting in $A[\mathsf{id}_Q]$, we have that $1 \leqq \lfloor A[\mathsf{id}_Q] \rfloor$. Because $q \; \mathsf{id}_Q \; q \in F$, we find $\lfloor A[\mathsf{id}_Q] \rfloor \leqq s(q)$ and hence $1 \leqq s(q)$.
- Suppose $q' \in \delta(q, \mathsf{a})$. We then derive as follows:

$$\begin{aligned}
\mathsf{a} \cdot s(q') &\leqq \lfloor A[\delta_\mathsf{a}] \rfloor \cdot s(q') && \text{(Lemma 4.3)} \\
&\equiv \sum_{q' R q_f \in F} \lfloor A[\delta_\mathsf{a}] \rfloor \cdot \lfloor A[R] \rfloor && \text{(def. } s\text{, distrib.)} \\
&\leqq \sum_{q' R q_f} \lfloor A[\delta_\mathsf{a} \circ R] \rfloor && \text{(Lemma 4.5)} \\
&\leqq \sum_{q R q_f} \lfloor A[R] \rfloor \equiv s(q) && \text{(see below)}
\end{aligned}$$

  The last inequation holds because if $R \subseteq Q \times Q$ and $q_f \in F$ are such that $q' \; R \; q_f$, then $q \; (\delta_\mathsf{a} \circ R) \; q_f$ because $q \; \delta_\mathsf{a} \; q'$; hence, terms in the second-to-last sum must also appear in the last sum.

To show that $s(q) \leq \overline{A}(q)$ for all $q \in Q$, consider a $q \in Q$, $q_f \in F$ and $R \subseteq Q \times Q$ such that $q \; R \; q_f$. It suffices to show that $\lfloor \overline{A[R]} \rfloor \leq \overline{A}(q)$. To this end, we define $s' : \mathcal{P}(Q \times Q) \to \mathbb{E}$ by choosing for $R' \subseteq Q \times Q$ that

$$s'(R') = \sum_{q R' q'} \overline{A}(q')$$

We claim that s' is a solution to $A[R]$. The conditions work out as follows:

- If $R' = R$, then since $q \; R \; q_f$, we have that $\overline{A}(q_f) \leqq s'(R)$. Since $q_f \in F$, we also know that $1 \leqq \overline{A}(q_f)$; this allows us to conclude $1 \leqq s'(R)$.
- If $R'' \in \delta^\tau(R', \mathsf{a})$, then $R'' = R' \circ \delta_\mathsf{a}$. We then derive as follows

$$\begin{aligned}
\mathsf{a} \cdot s'(R'') &\equiv \sum_{q R'' q''} \mathsf{a} \cdot \overline{A}(q'') && \text{(def. } s'\text{, distrib.)} \\
&\equiv \sum_{q R' q'} \sum_{q'' \in \delta(q', \mathsf{a})} \mathsf{a} \cdot \overline{A}(q'') && (R'' = R' \circ \delta_\mathsf{a}) \\
&\leqq \sum_{q R' q'} \overline{A}(q') = s'(R') && (\overline{A} \text{ solves } A)
\end{aligned}$$

Since s' is a solution to $A[R]$, we conclude by deriving that

$$\lfloor A[R] \rfloor \equiv \overline{A[R]}(\mathsf{id}_Q) \leqq s'(\mathsf{id}_Q) \equiv \sum_{q \mathsf{id}_Q q'} \overline{A}(q') \equiv \overline{A}(q) \qquad \square$$

C Proofs for Section 5

Lemma 5.10. *For all $e \in \mathbb{E}$, it holds that $\lfloor A_e \rfloor \leqq e$.*

Proof. Let $s : \rho(e) \to \mathbb{E}$ be the injection. Using the first part of Theorem 5.8, we can then show that s is a solution to A_e, and hence that for $e' \in \rho(e)$ we have that $\overline{A_e}(e') \leqq s(e') = e'$. By the second part of Theorem 5.8, we then conclude

$$\lfloor A_e \rfloor \equiv \sum_{e' \in \iota(e)} \overline{A_e}(e') \leqq \sum_{e' \in \iota(e)} e' \equiv e \qquad \square$$

Lemma 5.12. *The following hold for all $e_0, e_1, e_2 \in \mathbb{E}$:*

$$e_0 \lesssim e_0 \cdot 1 \qquad e_0 \lesssim e_0 + e_1 \qquad e_0 \lesssim e_1 \implies e_0 \cdot e_2 \lesssim e_1 \cdot e_2$$

$$e_0 \cdot e_0^* \lesssim e_0^* \qquad 1 \lesssim e_0^* \qquad e_0 \cdot (e_1 \cdot e_2) \lesssim (e_0 \cdot e_1) \cdot e_2$$

Proof. The first property is witnessed by the strong automaton morphism $h_1 : \rho(e) \to \rho(e \cdot 1)$ given by $h_1(e') = e' \cdot 1$. The strong morphisms required for the second, fourth and fifth property are the injections — e.g., $h_2 : \rho(e) \to \rho(e + f)$ given by $h_2(e') = e'$ for the second property. The sixth property is witnessed by the following strong morphism:

$$h_3(e') = \begin{cases} (e_0' \cdot e_1) \cdot e_2 & e' = e_0' \cdot (e_1 \cdot e_2),\ e_0' \in \rho(e_0) \\ e_1' \cdot e_2 & e' = e_1' \cdot e_2,\ e_1' \in \rho(e_1) \\ e_2' & e' = e_2' \in \rho(e_2) \end{cases}$$

For the third property, let $h : \rho(e_0) \to \rho(e_1)$ be the strong morphism from A_{e_0} to A_{e_1}. The following is then a strong morphism witnessing that $e_0 \cdot e_2 \lesssim e_1 \cdot e_2$:

$$h_6(e') = \begin{cases} h(e_0') \cdot e_2 & e' = e_0' \cdot e_2,\ e_0' \in \rho(e_0) \\ e_2' & e' = e_2' \in \rho(e_2) \end{cases} \qquad \square$$

Lemma 5.14. *Let $e \in \mathbb{E}$. It holds that $\lfloor A_{e \cdot 1} \rfloor \leqq \lfloor A_e \rfloor$ and $\lfloor A_e \rfloor \leqq \lfloor A_{1 \cdot e} \rfloor$.*

Proof. We start by showing that $\lfloor A_{e \cdot 1} \rfloor \leqq \lfloor A_e \rfloor$. To this end, we choose $s : \rho(e \cdot 1) \to \mathbb{E}$ by setting $s(e' \cdot 1) = \overline{A_e}(e')$ when $e' \in \rho(e)$, and $s(1) = 1$. It is fairly straightforward to show that this makes s a solution to $A_{e \cdot 1}$, and hence that $\overline{A_{1 \cdot e}}(e') \leq s(e')$ for all $e' \in \rho(e \cdot 1)$. We can then derive that

$$\lfloor A_{e \cdot 1} \rfloor = \sum_{e' \in \iota(e)} \overline{A_{e \cdot 1}}(e' \cdot 1) \leq \sum_{e' \in \iota(e)} s(e' \cdot 1) = \sum_{e' \in \iota(e)} \overline{A_e}(e') = \lfloor A_e \rfloor$$

It remains to show that $\lfloor A_e \rfloor \leqq \lfloor A_{1\cdot e} \rfloor$, which we do by deriving as follows:

$$\begin{aligned}
\lfloor A_e \rfloor &\equiv \sum_{e' \in \iota(e)} \overline{A_e}(e') && \text{(def. } \lfloor A_e \rfloor\text{)} \\
&\leqq \sum_{e' \in \iota(e)} \overline{A_{1\cdot e}}(e') && (\rho(e) \subseteq \rho(1 \cdot e)) \\
&\equiv \sum_{e' \in \iota(e)} \left([e' \in \mathbb{F}] + \sum_{e'' \in \partial(e', \mathtt{a})} \mathtt{a} \cdot \overline{A_{1\cdot e}}(e'') \right) && \text{(Lemma 3.9)} \\
&\leqq [1 \cdot e \in \mathbb{F}] + \sum_{e' \in \partial(1\cdot e, \mathtt{a})} \mathtt{a} \cdot \overline{A_{1\cdot e}}(e') && \text{(Lemma 5.9)} \\
&\leqq \overline{A_{1\cdot e}}(1 \cdot e) = \lfloor A_{1\cdot e} \rfloor && (\overline{A_{1\cdot e}} \text{ solves } A_{1\cdot e}) \quad \square
\end{aligned}$$

Lemma 5.15. *Let $e, f \in \mathbb{E}$. It holds that $e \cdot \lfloor A_f \rfloor \leqq \lfloor A_{e\cdot f} \rfloor$*

Proof. We proceed by induction on e. For the base, there are three cases.

- If $e = 0$, then the claim holds because $0 \cdot \lfloor A_f \rfloor \equiv 0 \leq \lfloor A_{0\cdot f} \rfloor$.
- If $e = 1$, then the claim follows by Lemma 5.12.
- If $e = \mathtt{a}$, then we derive as follows:

$$\begin{aligned}
\mathtt{a} \cdot \lfloor A_f \rfloor &\leqq \mathtt{a} \cdot \lfloor A_{1\cdot f} \rfloor && \text{(Lemma 5.14)} \\
&\equiv \mathtt{a} \cdot \overline{A_{1\cdot f}}(1 \cdot f) && \text{(def. } \lfloor A_{1\cdot f} \rfloor\text{)} \\
&\leqq \mathtt{a} \cdot \overline{A_{\mathtt{a}\cdot f}}(1 \cdot f) && \text{(see below)} \\
&\leqq \overline{A_{\mathtt{a}\cdot f}}(\mathtt{a} \cdot f) && (\overline{A_{\mathtt{a}\cdot f}} \text{ solves } A_{\mathtt{a}\cdot f}) \\
&\equiv \lfloor A_{\mathtt{a}\cdot f} \rfloor && \text{(def. } \lfloor A_{\mathtt{a}\cdot f} \rfloor\text{)}
\end{aligned}$$

  Here, the third step holds because $\rho(1 \cdot f) \subseteq \rho(\mathtt{a} \cdot f)$, and the relevant injection from $\rho(1 \cdot f)$ in $\rho(\mathtt{a} \cdot f)$ is an automaton morphism; we apply Lemma 3.6.

For the inductive step, there are three more cases.

- If $e = e_0 + e_1$, then we derive

$$(e_0 + e_1) \cdot \lfloor A_f \rfloor \equiv e_0 \cdot \lfloor A_f \rfloor + e_1 \cdot \lfloor A_f \rfloor \leqq \lfloor A_{e_0\cdot f} \rfloor + \lfloor A_{e_1\cdot f} \rfloor \leqq \lfloor A_{e\cdot f} \rfloor$$

  where the last step follows because $e_i \cdot f \lesssim e \cdot f$ for $i \in \{0, 1\}$ by Lemma 5.12.
- If $e = e_0 \cdot e_1$, then we derive

$$(e_0 \cdot e_1) \cdot \lfloor A_f \rfloor \equiv e_0 \cdot (e_1 \cdot \lfloor A_f \rfloor) \leqq e_0 \cdot \lfloor A_{e_1\cdot f} \rfloor \leqq \lfloor A_{e_0\cdot(e_1\cdot f)} \rfloor \leqq \lfloor A_{e\cdot f} \rfloor$$

  where the last step follows because $e_0 \cdot (e_1 \cdot f) \lesssim (e_0 \cdot e_1) \cdot f$ by Lemma 5.12.
- If $e = e_0^*$, then we derive

$$e_0 \cdot \lfloor A_{e_0^*\cdot f} \rfloor + \lfloor A_f \rfloor \leqq \lfloor A_{e_0\cdot e_0^*\cdot f} \rfloor + \lfloor A_{1\cdot f} \rfloor \leqq \lfloor A_{e_0^*\cdot f} \rfloor$$

  where the last step follows because $e_0 \cdot e_0^* \lesssim e_0^*$ as well as $1 \lesssim e_0^*$, and hence $e_0 \cdot e_0^* \cdot f \lesssim e_0^* \cdot f$ as well as $1 \cdot f \lesssim e_0^* \cdot f$ by Lemma 5.12. If we apply the left fixpoint rule to the above, we then find that $e_0^* \cdot \lfloor A_f \rfloor \leqq \lfloor A_{e_0^*\cdot f} \rfloor$ $\square$

D Proofs for Section 6

Lemma 6.3. *Let $(M, \cdot, 1)$ be a monoid and let $(\mathcal{P}(M), \cup, \otimes, ^{\circledast}, \emptyset, \{1\})$ be the KA obtained from it, per Lemma 6.2. Furthermore, let $h_1 : \Sigma \to M$ and $h_2 : \Sigma \to \mathcal{P}(M)$ be such that for all $\mathtt{a} \in \Sigma$, we have that $h_2(\mathtt{a}) = \{h_1(\mathtt{a})\}$. Then for $e \in \mathbb{E}$:*

$$\widehat{h_2}(e) = \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e)\}$$

Proof. We proceed by induction on $e \in \mathbb{E}$. In the base, there are three cases.

- If $e = 0$, then $\widehat{h_2}(0) = \emptyset = \{\widetilde{h_1}(w) : w \in \emptyset\} = \{\widetilde{h_1}(w) : w \in \widehat{\ell}(0)\}$.
- If $e = 1$, then $\widehat{h_2}(1) = \{1\} = \{\widetilde{h_1}(\epsilon)\} = \{\widetilde{h_1}(w) : w \in \widehat{\ell}(1)\}$.
- If $e = \mathtt{a}$, then $\widehat{h_2}(\mathtt{a}) = h_2(\mathtt{a}) = \{h_1(\mathtt{a})\} = \{\widetilde{h_2}(w) : w \in \widehat{\ell}(\mathtt{a})\}$.

For the inductive step, there are three more cases:

- If $e = e_0 + e_1$, then we derive

$$\begin{aligned}
\widehat{h_2}(e_0 + e_1) &= \widehat{h_2}(e_0) \cup \widehat{h_2}(e_1) \\
&= \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0)\} \cup \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_1)\} \\
&= \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0 + e_1)\}
\end{aligned}$$

- If $e = e_0 \cdot e_1$, then we derive

$$\begin{aligned}
\widehat{h_2}(e_0 \cdot e_1) &= \widehat{h_2}(e_0) \otimes \widehat{h_2}(e_1) \\
&= \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0)\} \otimes \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_1)\} \\
&= \{\widetilde{h_1}(w_0)\widetilde{h_1}(w_1) : w_0 \in \widehat{\ell}(e_0),\ w_1 \in \widehat{\ell}(e_1)\} \\
&= \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0 \cdot e_1)\}
\end{aligned}$$

- If $e = e_0^*$, then we derive

$$\begin{aligned}
\widehat{h_2}(e_0^*) &= \widehat{h_2}(e_0)^{\circledast} \\
&= \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0)\}^{\circledast} \\
&= \{w_1 \cdot \ldots \cdot w_n : w_1, \ldots, w_n \in \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0)\}\} \\
&= \{\widetilde{h_1}(w_1) \cdot \ldots \cdot \widetilde{h_1}(w_n) : w_1, \ldots, w_n \in \widehat{\ell}(e_0)\} \\
&= \{\widetilde{h_1}(w) : w \in \widehat{\ell}(e_0^*)\}\}
\end{aligned}$$

□