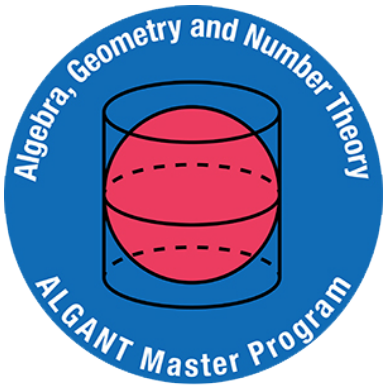
Axiomatizing Prefix: Closure on Regular Languages

Sugimoto, M.



ALGANT MASTER THESIS

AXIOMATIZING PREFIX-CLOSURE ON REGULAR LANGUAGES

Supervisors:

Prof. M. Bonsangue
Prof. P. Bruin

Student:

Melissa Sugimoto

Università degli Studi di Padova

Universiteit Leiden

Academic year 2021/2022

Contents

Chapter 1

Introduction

Formal language theory, at its inception, sits in the intersection of mathematics, computer science, and linguistics. Many people trace the founding of formal languages, in the sense that they are used today, to the work of Noam Chomsky, famous for the Chomsky hierarchy, formalized in 1964. However, this work builds upon that of immediate predecessors such as Alan Turing, who is sometimes called the father of theoretical computer science. In fact, Turing and Post defined and studied "[t]he types of machines and grammars used to define the class of recursively enumerable languages" used in the Chomsky hierarchy as early as the 1930s and 40s [5]. These types of languages are, respectively, recursively enumerable, context-sensitive, context-free, and regular.

Of course, one would be remiss to discuss the development of formal languages without mentioning automata, as an automaton is a model of a "computing device, which acts as a language acceptor" [5]. In fact, the classes of the Chomsky hierarchy "can be defined... by considering the languages accepted by non-deterministic machines with an increasingly restricted type of data structure (or working tape): Turing machine, linear-bounded automaton, pushdown-store automaton, and finite-state automaton" [5]. Though we do not discuss automata in any detail in this work, we will make a brief connection between the types of languages which we discuss and the kinds of finite-state automata that accept them as inputs.

Of the classes of the Chomsky hierarchy, our work will deal with subsets of regular languages, the most restrictive class, and grammars used to describe them. Regular languages were first defined in the 1950s by Stephen Cole Kleene, and were later shown to correspond to the eponymously named Kleene algebras in Dexter Kozen's 1994 paper "A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events."

To discuss this connection more formally, it is necessary to delve into the concepts of soundness and completeness in mathematical logic. A logical system is called sound if everything that can be derived in the system is true in every model. Conversely, a system is called complete if everything that is true in every model can be derived as a theorem in the system. One might formulate soundness and completeness together as "everything that can be proved is true, and everything that is true can be proved."

Then, when we say that Kleene algebras correspond to regular languages, what we mean is that the axioms of Kleene algebras given in [8] form a deductive system is sound and complete for regular languages over an alphabet Σ, sometimes also referred to as the

algebra of regular events. The goal of this thesis is to axiomatize what we call prefix-closed regular languages by adding any necessary axioms to the Kleene algebra axioms, and then to achieve soundness and completeness results for this new axiomatization. Following this, we hope to expand upon the work of [14] by axiomatizing prefix-closed synchronous Kleene algebras using the new axioms proposed in Section 5 of the same.

More specifically, we proceed as follows. In **Chapter 2**, we introduce the definitions of formal languages, regular expressions and languages, and Kleene algebras, and present some relevant examples and properties. In **Chapter 3**, we define prefix-closed languages and some of their properties. We attempt to define a prefix-closure operator, and then propose two methods of circumventing the difficulty that arises from this attempt. We end with an axiomatization of prefix-closed regular languages, including soundness and completeness results. In **Chapter 4**, we take a brief diversion to discuss the synchronous operator on languages and define the SF_1-algebra as presented in [14], alongside the relevant soundness and completeness results. In **Chapter 5**, we discuss the possibility of adapting our axiomatization of prefix-closed regular languages to create an analogous axiomatization of prefix-closed regular languages. We conclude in **Chapter 6** with a discussion on some applications of regular, and prefix-closed languages that offer some motivation to our work.

Chapter 2

Preliminaries

2.1 Formal Languages

Our subject matter deals with the concept of *formal languages* in the context of mathematics and theoretical computer science. The word language brings to mind systems of communication such as English or Japanese, with sets of characters used to form words, and sentences formed of words and additional symbols according to a strict set of rules. One may also think of programming languages, such as Python or C#, which also require the correct usage of symbols according to a particular set of rules – the language's syntax – in order to create a valid statement. Formal languages generalize these concepts of language in a way that can be described using mathematical and logical terminology. A formal language consists of words formed over an alphabet according to a set of rules. That is,

Definition 2.1.1. An *alphabet*, Σ, is a set of symbols. A *word* over Σ is a finite sequence of symbols in Σ, where the empty word is denoted by ε. The set of all words is denoted by Σ^*. A *formal language* is a subset of Σ^*.

It is common in this context to refer to a formal language simply as a "language" when the meaning is clear. We will adopt that convention going forward. The simplest example of a language L is the empty language, $L = \emptyset$, which is a language over any alphabet. Languages over an alphabet can be either finite or infinite. For example, take the alphabet $\Sigma = \{a, b\}$. The set $\{a, b, abab\}$ is a language containing only three elements, while the set of all words ending in a is infinite. To provide motivation for the use of the term "language," one can note that the set of all English words forms a language over the alphabet consisting of the twenty-six letters in the English alphabet.

Given two words e, f in a language L, the *concatenation* of e and f is denoted by ef. Given two languages L_1 and L_2 over an alphabet Σ, we can define some common operations on languages. The *union* $L_1 \cup L_2 = \{e \in \Sigma^* \mid e \in L_1 \vee e \in L_2\}$ is the set of all words that are contained in L_1 or L_2. The *intersection* $L_1 \cap L_2 = \{e \in \Sigma^* \mid e \in L_1 \wedge e \in L_2\}$ is the set of all words that are contained in both L_1 and L_2. If L_1 and L_2 have no words in common, then their intersection is the empty language. The *concatenation* $L_1 \cdot L_2 = \{ef \in \Sigma^* \mid e \in L_1 \wedge f \in L_2\}$ is the set of all words that arise by attaching a

word in L_2 to the end of a word in L_1. Finally, we define the *Kleene star* of a language recursively, as follows:

Definition 2.1.2. Given a language L over an alphabet Σ, define

$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^1 &= L \\ L^{k+1} &= \{ef \in \Sigma^* \mid e \in L^k \wedge f \in L\} \text{ for each } k \geq 0 \end{aligned}$$

Then the *Kleene star* of L is defined by

$$L^* = \bigcup_{k \geq 0} L^k.$$

A particularly notable application of the Kleene star comes in the form of Arden's Lemma, which provides a solution for linear equations of languages of a certain form [1]. We present two versions of this lemma.

Lemma 2.1.3. *(Arden's Lemma, version 1). Let X, L, and M be languages over the same alphabet, Σ. Suppose $\varepsilon \notin L$ and $X = L \cdot X \cup M$. Then $X = L^* \cdot M$.*

This first version of Arden's Lemma provides the conditions under which the equation $X = L \cdot X \cup M$ has a unique solution. However, this is rather restrictive in the requirement $\varepsilon \notin L$. The second version is more general, and allows for the existence of many solutions, of which $L^* \cdot M$ is the least.

Lemma 2.1.4. *(Arden's Lemma, version 2). Let L and M be languages over Σ, and let $X = L^* \cdot M$. Then,*

1. *$X = L \cdot X \cup M$,*
2. *for any other language Y such that $Y = L \cdot Y \cup M$, $X \subseteq Y$, and*
3. *If L does not contain ε, then the above inclusion is equality.*

Proof. First, let us prove that $L^* \cdot M \subseteq L(L^* \cdot M) \cup M$. Suppose $x = x_1 x_2 \in L^* \cdot M$, where $x_1 \in L^*$ and $x_2 \in M$. If $x_1 = \varepsilon$, then certainly $x = x_2 \in M$. Now let us assume $x_1 \neq \varepsilon$. Then $x_1 \in L^n$ for some $n \geq 1$. This implies $x_1 \in L \cdot L^{n-1} \subseteq L \cdot L^*$. Hence $x = x_1 x_2 \in L(L^* \cdot M)$.

For the reverse inequality, let us assume $x \in L(L^* \cdot M) \cup M$. We have two cases. First, $x \in M$, in which case $\varepsilon x = x \in L^* \cdot M$. Second, $x \in L(L^* \cdot M)$. Then $x = x_1 x_2 x_3$, where $x_1 \in L$, $x_2 \in L^*$, and $x_3 \in M$. This implies that $(x_1 x_2) \in L^*$, and hence $x \in L^* \cdot M$. Therefore $X = L^* \cdot M$ is a solution to the equation.

To prove the second result, we will first prove that for any solution Y of the equation, $L^* Y \subseteq Y$. It is obvious that $L^0 \cdot Y \subseteq Y$. Additionally, Since $Y = L \cdot Y \cup M$, it follows that $L \cdot Y \subseteq Y$, and that $M \subseteq Y$. Let us assume that for some $n \in Z_{\geq 1}$, $L^n \cdot Y \subseteq Y$.

Then $L^{n+1} \cdot Y = L^n(L \cdot Y) \subseteq L^n \cdot Y \subseteq Y$. Hence, for any $n \in \mathbb{Z}_{\geq 0}$, $L^n \cdot Y \subseteq Y$, and therefore $L^* \cdot Y \subseteq Y$.

Because we also have $M \subseteq Y$, we conclude $L^* \cdot M \subseteq L^* \cdot Y \subseteq Y$.

Finally, let us now assume $\varepsilon \notin L$. Then we will prove that $Y = X$. By the previous point, we already have $X \subseteq Y$, so we need only show $Y \subseteq X$. We perform a proof by contradiction, with a small intermediate induction.

Suppose $x \in Y$ and $x \notin L^* \cdot M$. Then certainly $x \in L \cdot Y \cup M$. Hence $x \in L \cdot Y$ since we cannot have $x \in M$. Now suppose $x \in L^k \cdot Y$ for some $k \geq 1$, an integer. We will show $x \in L^{k+1} \cdot Y$.

We have $x \in L^k(L \cdot Y \cup M) = L^{k+1} \cdot Y \cup L^k \cdot M$. Now since we cannot have $x \in L^* \cdot M$, we must have $x \in L^{k+1} \cdot Y$ and $x \notin L^k \cdot M$. Therefore we inductively obtain that $x \in L^n \cdot Y$ for every integer $n \geq 1$ (and in fact also $n \geq 0$ since we have already that $x \in Y = L^0 \cdot Y$). But since we assume in general that words are finite, this is impossible. Hence $x \in L^* \cdot M$. $\square$

One may notice a similarity between the kinds of operations that can be performed on languages and sets of operations that may be defined on algebraic structures.

2.2 Kleene Algebras

Kleene algebras are named for American mathematician Stephen Cole Kleene, a mathematician known for his work in logic, including providing the foundation of recursion theory and paving the way for theoretical computer science. Kleene himself did not define Kleene algebras but introduced the related concept of regular expressions, which will be discussed in the next section.

Definition 2.2.1. A *Kleene algebra* is a tuple $(A, +, \cdot, *, 0, 1)$ where A is a set, $+$ and $\cdot$ are binary operators, $*$ is a unary operator, and 0 and 1 are constants such that for all $e, f, g \in A$ the following axioms are satsified:

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

Additionally, we write $e \leq f$ for $e + f = f$ and require the *least fixpoint axioms*: That is, for all $e, f, g \in A$ we have

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

The definition of Kleene algebra as it is used in this work was introduced by Kozen in 1994 [8]. Kleene algebras may be more familiar to some described as a kind of closed semiring – that is, a semiring with a quasi-inverse, in this case satisfied by the elements e^*.

Proposition 2.2.2. *The set of all languages forms a Kleene Algebra.*

Proof. The proof is largely set theoretic in nature and forms an accessible exercise. Let us take as our tuple $(\mathcal{P}(\Sigma^*), \cup, \cdot, *, \emptyset, \{\varepsilon\})$, where $\cdot$ and $*$ are concatenation and Kleene star, respectively.

First, we note that set union is associative, commutative and idempotent, with an identity $\emptyset$. This satisfies the top row of the Kleene algebra axioms.

Now, because $x \cdot \varepsilon = x$ for any word x, it is clear that $\{\varepsilon\}$ is the identity for concatenation. Next, let $L \in \mathcal{P}(\Sigma^*)$. Then $L \cdot \emptyset = \{xy \in \Sigma^* \mid x \in L \wedge y \in \emptyset\} = \emptyset$. Clearly $\emptyset \cdot L = \emptyset$ as well. It is straightforward to see that concatenation is an associative operation.

Before proving that the Kleene star axiom holds, we prove one version of distributivity of union over concatenation. The other version will follow by analogous reasoning. Let L, J, K be languages. Then we have

$$\begin{aligned}(L \cup J) \cdot K &= \{xy \in \Sigma^* \mid x \in L \cup J \wedge y \in K\} \\ &= \{xy \in \Sigma^* \mid (x \in L \vee x \in J) \wedge y \in K\} \\ &= \{xy \in \Sigma^* \mid (x \in L \wedge y \in K) \vee (x \in J \wedge y \in K\} \\ &= L \cdot K \cup J \cdot K\end{aligned}$$

Next, we show that $L^* = \{\varepsilon\} \cup L \cdot L^*$. The second equality will follow in the same way.

First, suppose $x \in L^*$. Then $x \in L^n$ for some $n \in \mathbb{Z}_{\geq 0}$. If $n = 0$, then $x = \varepsilon$. Hence $x \in \{\varepsilon\} \cup L^* \cdot L$. On the other hand, if $n > 0$, then $x \in L^n = L^{n-1} \cdot L$ by the definition of L^k. Hence, by the definition of Kleene star, $x \in L^* \cdot L$. So, $L^* \subseteq \{\varepsilon\} \cup L^* \cdot L$.

For the reverse inclusion, suppose $x \in \{\varepsilon\} \cup L^* \cdot L$. If $x = \varepsilon$, then certainly $x \in L^*$. If $x \in L^* \cdot L$, then there exist $x_1 \in L^*$ and $x_2 \in L$ such that $x = x_1 x_2$. Then there exists some $n \in \mathbb{Z}_{\geq 0}$ such that $x_1 \in L^n$. Hence $x = x_1 x_2 \in L^{n+1} \subseteq L^*$. Thus $\{\varepsilon\} \cup L^* \cdot L \subseteq L^*$, and we have equality.

Finally, we prove the first of the least fixpoint axioms (as, again, the second uses the exact same method as the first). We note that the partial order on $\mathcal{P}(\Sigma^*)$ is given by set inclusion. Now, assume $L \cup J \cdot K \subseteq K$. Then we will show that $J^* \cdot L \subseteq K$.
First,

$$L \cup J \cdot K \subseteq K \implies L \subseteq K \wedge J \cdot K \subseteq K.$$

We also note that $J^0 \cdot K \subseteq K$ since $J^0 = \{\varepsilon\}$ Next, we have

$$J^n \cdot K \subseteq K \implies J^{n+1} \cdot K \subseteq K$$

for all $n \in \mathbb{Z}_{\geq 0}$, since

$$J^{n+1} \cdot K = J^n \cdot J \cdot K \subseteq J^n \cdot K \subseteq K.$$

It therefore follows that $J^* \cdot K \subseteq K$. But then $L \subseteq K$ implies that $J^* \cdot L \subseteq J^* \cdot K \subseteq K$, and we are finished. □

In the next section we discuss another important example of Kleene algebra, defined as the smaller class of languages that can be obtained from a given alphabet and closed under the language operations as in the above theorem.

2.3 Regular Languages

A *regular expression* is a syntactic means to denote languages over an alphabet Σ using the above defined operations of union, concatenation, and Kleene star. Languages that can be expressed by regular expressions are called *regular languages*. For example, take again the alphabet $\Sigma = \{a, b\}$. Let L be the set of all words over Σ ending in aa. Let $L_1 = \{a, b\}$ and $L_2 = \{aa\}$. Then $L = L_1^* L_2$.

All finite languages are regular. However, in general, not all languages are regular languages. For example, take again $\Sigma = \{a, b\}$. Then the language $L = \{a^n b^n \mid n \in \mathbb{Z}_{\geq 0}\}$, where a^n represents concatenation of a with itself n times, is not regular [9].

A *formal grammar*, or just *grammar*, describes how to form valid expressions from a set with some operations. We can use grammars to formalize the definitions of regular expression and regular language given above as follows:

Definition 2.3.1. The set of *regular expressions* of a Kleene algebra, denoted $\mathcal{T}_{\mathsf{KA}}$, is described by the grammar:

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

The regular expressions can be interpreted in terms of languages as follows: Define $[\![-]\!]_{\mathsf{KA}} : \mathcal{T}_{\mathsf{KA}} \to \mathcal{P}(\Sigma^*)$ inductively by

$$\begin{array}{lll} [\![0]\!]_{\mathsf{KA}} = \emptyset & [\![a]\!]_{\mathsf{KA}} = \{a\} & [\![e \cdot f]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}} \cdot [\![f]\!]_{\mathsf{KA}} \\ [\![1]\!]_{\mathsf{KA}} = \{\varepsilon\} & [\![e + f]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}} \cup [\![f]\!]_{\mathsf{KA}} & [\![e^*]\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}}^* \end{array}$$

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

Just as in the previous proposition, we notice the following:

Proposition 2.3.2. *The set of all regular languages forms a Kleene Algebra.*

This result follows from the fact that $\emptyset$ and $\{\varepsilon\}$ are regular languages, and that regular languages are closed under the operations of union, concatenation, and Kleene star.

Define $\equiv_{\mathsf{KA}}$ to be the smallest congruence on $\mathcal{T}_{\mathsf{KA}}$ induced by the Kleene algebra axioms. That is, for two regular expressions e and f, $e \equiv_{\mathsf{KA}} f$ if and only if e can be proved equivalent to f according to the Kleene algebra axioms. Then we can say that KA is *sound* if $e \equiv_{\mathsf{KA}} f$ implies $[\![e]\!]_{\mathsf{KA}} = [\![f]\!]_{\mathsf{KA}}$ and *complete* if $[\![e]\!]_{\mathsf{KA}} = [\![f]\!]_{\mathsf{KA}}$ implies $e \equiv_{\mathsf{KA}} f$. A well-known but certainly nontrivial result due to Kozen states the following:

Theorem 2.3.3. *Soundness and Completeness of* KA. *For all* $e, f \in \mathcal{T}_{\mathsf{KA}}$, $e \equiv_{\mathsf{KA}} f$ *if and only if* $[\![e]\!]_{\mathsf{KA}} = [\![f]\!]_{\mathsf{KA}}$.

The first implication, soundness, follows from induction on the length of the proof. However, the converse implication is more complicated and is covered thoroughly in [8].

Chapter 3

Prefix-closed Languages

3.1 Definition and Properties

A language is called *prefix-closed* if for any word in the language, every prefix of the word is also in the language.

Definition 3.1.1. Let L be a language over an alphabet Σ. We say that L is *prefix-closed* if for any $e = xy \in L$, we have that $x \in L$.

Note that this implies that $\varepsilon \in L$ for any non-empty prefix-closed language L, since $e = \varepsilon e$ for any word e. We also note that $\emptyset$ is prefix-closed since the condition is vacuously true. If we use the notation $a^n = a^{n-1}a$ for integers $n > 0$ with the convention $a^0 = \varepsilon$, then an example of a prefix-closed language is $L = \{a^n \mid n \in \mathbb{Z}_{\geq 0}\}$. On the other hand, the language $L = \{a, b\}^*\{aa\}$ is regular but not prefix-closed: $\varepsilon \notin L$.

Definition 3.1.2. The *prefix-closure* of a language L, denoted L^{pc}, over an alphabet Σ is defined by $L^{pc} := \{x \in \Sigma^* \mid xy \in L\}$. The prefix-closure is an idempotent operation, as $(L^{pc})^{pc} = L^{pc}$.

Another way to define a prefix-closed language is as a language that is equal to its prefix-closure. That is, a language L such that $L^{pc} = L$.

Proposition 3.1.3. *Let L_1 and L_2 be two prefix-closed languages over an alphabet Σ. Then,*

1. *$L_1 \cup L_2$ is prefix-closed,*
2. *$L_1 \cap L_2$ is prefix-closed, and*
3. *if $L_1 \neq \emptyset$ and $L_1 \neq \Sigma^*$, $\Sigma^* \setminus L_1$ is not prefix-closed.*

Proof. Assume L_1 and L_2 are prefix-closed languages. We begin with (1).

Let $xy \in L_1 \cup L_2$. Then $xy \in L_1$ or $xy \in L_2$. Without loss of generality, we may assume $xy \in L_1$. Then, since L_1 is prefix-closed, $x \in L_1$. Hence $x \in L_1 \cup L_2$.

Proceeding to (2). we let $xy \in L_1 \cap L_2$. Then $xy \in L_1$ and $xy \in L_2$. Hence, since L_1 and L_2 are prefix-closed, we have $x \in L_1$ and $x \in L_2$. Consequently $x \in L_1 \cap L_2$.

Finally, assume $L_1 \neq \emptyset$ and $L_1 \neq \Sigma^*$. Then $\Sigma^* \setminus L_1 \neq \emptyset$ and $\Sigma^* \setminus L_2 \neq \Sigma^*$. Furthermore, $\varepsilon \in L_1$, so $\varepsilon \notin \Sigma^* \setminus L_1$, and therefore $\Sigma^* \setminus L_1$ cannot be prefix-closed. □

Lemma 3.1.4. *Let L_1 and L_2 be two languages over an alphabet Σ. Then $L_1^{pc} \cup L_2^{pc} = (L_1 \cup L_2)^{pc}$.*

Proof. We may assume that L_1 and L_2 are both nonempty, as otherwise the problem becomes trivial.

First we will show that $L_1^{pc} \cup L_2^{pc} \subseteq (L_1 \cup L_2)^{pc}$. Suppose that $x \in L_1^{pc} \cup L_2^{pc}$. Without loss of generality, we may assume that $x \in L_1^{pc}$. Then there exists $y \in \Sigma^*$ such that $xy \in L_1$. Hence $xy \in L_1 \cup L_2$, and $x \in (L_1 \cup L_2)^{pc}$.

For the reverse inclusion, suppose that $x \in (L_1 \cup L_2)^{pc}$. Then there exists y such that $xy \in L_1 \cup L_2$. It follows that either $x \in L_1^{pc}$ or L_2^{pc}. □

We do not have an analogous result for $L_1^{pc} \cap L_2^{pc}$. To see this, suppose $L_1 = \{ab\}$ and $L_2 = \{ac\}$. Then $L_1^{pc} = \{\varepsilon, a, ab\}$ and $L_2^{pc} = \{\varepsilon, a, ac\}$, so $L_1^{pc} \cap L_2^{pc} = \{\varepsilon, a\}$. Meanwhile $L_1 \cap L_2 = \emptyset$, so $(L_1 \cap L_2)^{pc} = \emptyset$.

3.2 An Attempt at Axiomatizing Prefix Closure

Now, let $e, f \in \mathcal{T}_{\mathsf{KA}}$. Then we may wish to define the prefix-closure operator $(-)^{pc}$ at the level of regular expressions inductively by

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

However, this definition presents a difficulty:

Suppose $e = 0$. Then we may choose any a and write $e = a \cdot 0$. By definition, we have $e^{pc} = 0$, but we also have $e^{pc} = a^{pc} + f \cdot 0^{pc} = a^{pc}$. Hence $a^{pc} = 0 = 1 + a$, which is impossible. Therefore we cannot define a prefix-closure operator in this way.

In order to circumvent this difficulty, we propose two possible solutions: one by definition of a predicate that allows for a case analysis on each term to determine if the term is nonzero, and one by a modification of the grammar to exclude multiplication by zero on the right.

To that end, we define the following:

Definition 3.2.1. Let $e \in \mathcal{T}_{\mathsf{KA}}$. The *nonzero predicate*, denoted $NZ(e)$, is defined by

$NZ(1)$

$NZ(a)$

$NZ(e^*)$

$NZ(e_1 + e_2)$ if and only if $NZ(e_1)$ or $NZ(e_2)$

$NZ(e_1 \cdot e_2)$ if and only if $NZ(e_1)$ and $NZ(e_2)$.

We can likewise define the *zero predicate* by the logical negation of the non-zero predicate. That is, for $e \in \mathcal{T}_{\mathsf{KA}}$, we have

$Z(0)$

$Z(e_1 + e_2)$ if and only if $Z(e_1)$ and $Z(e_2)$

$Z(e_1 \cdot e_2)$ if and only if $Z(e_1)$ or $Z(e_2)$.

Proposition 3.2.2. *Let $e \in \mathcal{T}_{\mathsf{KA}}$. Then $Z(e)$ if and only if $[\![e]\!]_{\mathsf{KA}} = \emptyset$.*

Proof. Let $e \in \mathcal{T}_{\mathsf{KA}}$. First, suppose $Z(e)$ holds. We will prove the result by induction:

1. $e = 0$: Then certainly $[\![0]\!]_{\mathsf{KA}} = \emptyset$.
2. $e = e_1 + e_2$: $Z(e)$ implies $Z(e_1)$ and $Z(e_2)$. But by induction, $[\![e_1]\!]_{\mathsf{KA}} = \emptyset$ and $[\![e_2]\!] = \emptyset$. Hence $[\![e]\!]_{\mathsf{KA}} = [\![e_1]\!]_{\mathsf{KA}} \cup [\![e_2]\!]_{\mathsf{KA}} = \emptyset$.
3. $e = e_1 \cdot e_2$: $Z(e)$ implies $Z(e_1)$ or $Z(e_2)$. We look at the case $Z(e_1)$, as the other case is nearly identical. Then $[\![e_1]\!]_{\mathsf{KA}} = \emptyset$, so $[\![e]\!]_{\mathsf{KA}} = [\![e_1]\!]_{\mathsf{KA}}[\![e_2]\!]_{\mathsf{KA}} = \emptyset$.

Conversely, assume $[\![e]\!]_{\mathsf{KA}} = \emptyset$. Then by 2.3.3, we must have $e \equiv_{\mathsf{KA}} 0$. It is clear that $Z(e)$ follows from the Kleene Algebra axioms. □

We are now ready to define the prefix-closure operator ^{pc_1} using the nonzero predicate:

Let $e, f \in \mathcal{T}_{\mathsf{KA}}$. We define 0^{pc_1}, 1^{pc_1}, a^{pc_1}, $(e^*)^{pc_1}$, and $(e + f)^{pc_1}$ exactly as in our previous attempt. To handle the case $(e \cdot f)^{pc_1}$, we perform a case analysis and define

$(e \cdot f)^{pc_1} = e^{pc_1} + e \cdot f^{pc_1}$ if $NZ(f)$ and,

$(e \cdot f)^{pc_1} = 0$ if $Z(f)$.

We also define $(e^*)^{pc}$ by

$(e^*)^{pc_1} = e^* \cdot e^{pc_1}$ if $NZ(e)$ and,

$(e^*)^{pc_1} = 1$ otherwise.

We achieve the following result:

Lemma 3.2.3. *Let $e \in \mathcal{T}_{\mathsf{KA}}$ with $NZ(e)$. Then for any $n \in \mathbb{Z}_{\geq 0}$ we have $[\![(e^n)^{pc_1}]\!]_{\mathsf{KA}} \subseteq [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$.*

Proof. We go by induction on n. First consider the case $n = 0$. Then by definition $e^0 = 1$. It follows that $[\![1^{pc_1}]\!]_{\mathsf{KA}} = \{\varepsilon\} \subseteq [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$ since $\varepsilon \in [\![e^*]\!]_{\mathsf{KA}}$ and $\varepsilon \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$.

Now, suppose that $[\![(e^n)^{pc_1}]\!]_{\mathsf{KA}} \subseteq [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$. We must show that the claim holds for $n + 1$.

By the definition of ^{pc_1}, we know $[\![(e^{n+1})^{pc_1}]\!]_{\mathsf{KA}} = [\![(e^n \cdot e)^{pc_1}]\!]_{\mathsf{KA}} = [\![(e^n)^{pc_1}]\!]_{\mathsf{KA}} \cup [\![e^n]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$. By the inductive hypothesis it follows $[\![(e^n)^{pc_1}]\!]_{\mathsf{KA}} \subseteq [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$. Furthermore, we know $[\![e^n]\!]_{\mathsf{KA}} \subseteq [\![e^*]\!]_{\mathsf{KA}}$ through the definition of the Kleene star. Hence $[\![e^n]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}} \subseteq [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$.

Hence the result follows. □

Then, for any regular expression, the interpretation of the prefix closure operator is a prefix closed language. Formally,

Lemma 3.2.4. *Let $e \in \mathcal{T}_{\mathsf{KA}}$ be a regular expression. Then for any $xy \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$, we have that $x \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$.*

Proof. We proceed inductively. First, $0^{pc_1} = 0$, and $[\![0]\!]_{\mathsf{KA}} = \emptyset$, which is prefix-closed. Furthermore, $1^{pc_1} = 1$. Therefore, $[\![1^{pc_1}]\!]_{\mathsf{KA}} = \{\varepsilon\}$. The only prefix of ε is again ε, so $[\![1^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed. Next, $a^{pc_1} = 1 + a$. So $[\![a^{pc_1}]\!]_{\mathsf{KA}} = [\![1]\!]_{\mathsf{KA}} \cup [\![a]\!]_{\mathsf{KA}}$. But this is simply $\{\varepsilon, a\}$. The only possible prefixes in this language are ε and a, hence $[\![a^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed.

Next, take $e = e_1 + e_2$, with the assumption that $[\![f^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed for any f smaller than e. Then $[\![(e_1 + e_2)^{pc_1}]\!]_{\mathsf{KA}} = [\![e_1^{pc_1}]\!]_{\mathsf{KA}} \cup [\![e_2^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed by Lemma 3.1.3 as both $[\![e_1^{pc_1}]\!]_{\mathsf{KA}}$ and $[\![e_2^{pc_1}]\!]_{\mathsf{KA}}$ are prefix-closed.

Similarly, take $e = e_1 \cdot e_2$, with the assumption that $[\![f^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed for any f smaller than e. Now, we have two cases:

First, suppose that $NZ(e_2)$. Then $[\![(e_1 \cdot e_2)^{pc_1}]\!]_{\mathsf{KA}} = [\![e_1^{pc_1}]\!]_{\mathsf{KA}} \cup [\![e_1 \cdot e_2^{pc_1}]\!]_{\mathsf{KA}}$.

Suppose that $xy \in [\![e_1^{pc_1}]\!]_{\mathsf{KA}} \cup [\![e_1 \cdot e_2^{pc_1}]\!]_{\mathsf{KA}}$. If $xy \in [\![e_1^{pc_1}]\!]_{\mathsf{KA}}$, then certainly $x \in [\![e_1^{pc_1}]\!]_{\mathsf{KA}}$, since it is prefix-closed. Assume $xy \in [\![e_1 \cdot e_2^{pc_1}]\!]_{\mathsf{KA}} = [\![e_1]\!]_{\mathsf{KA}}[\![e_2^{pc_1}]\!]_{\mathsf{KA}}$. If $x \in [\![e_1]\!]_{\mathsf{KA}}$, then certainly $x \in [\![e_1^{pc_1}]\!]_{\mathsf{KA}}$, so, again, $x \in [\![e_1^{pc_1}]\!]_{\mathsf{KA}}$, and we are done. Next, we may assume that $x = x_1x_2$, where $x_1 \in [\![e_1]\!]_{\mathsf{KA}}$ and $x_2 \in [\![e_2^{pc_1}]\!]_{\mathsf{KA}}$. Then $x = x_1x_2 \in [\![e_1]\!]_{\mathsf{KA}}[\![e_2^{pc_1}]\!]_{\mathsf{KA}} \subseteq [\![e^{pc_1}]\!]_{\mathsf{KA}}$. Finally, we handle the case where $xy \in [\![e_2^{pc_1}]\!]_{\mathsf{KA}}$, which can only happen if $\varepsilon \in [\![e_1]\!]_{\mathsf{KA}}$. Since $[\![e_2^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed, it follows that $x \in [\![e_2^{pc_1}]\!]_{\mathsf{KA}}$. Hence $\varepsilon x \in [\![e_1]\!]_{\mathsf{KA}}[\![e_2^{pc_1}]\!]_{\mathsf{KA}}$.

In the second case, then $(e_1 \cdot e_2)^{pc_1} = 0$, so we are finished.

Finally, we must show that $[\![(e^*)^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed. First, we recall that $[\![(e^*)^{pc_1}]\!]_{\mathsf{KA}} = [\![e^* \cdot e^{pc_1}]\!]_{\mathsf{KA}} = [\![e]\!]^*_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$. Now suppose that $xy \in [\![e]\!]^*_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$. We now have a few cases.

1. Suppose $x \in [\![e]\!]^*_{\mathsf{KA}}$ and $y \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$. Now, we know by induction that $[\![e^{pc_1}]\!]_{\mathsf{KA}}$ is prefix-closed. Thus $\varepsilon \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$, and hence $x = x\varepsilon \in [\![e]\!]^*_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$.

2. Suppose $xy \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$. Note that this is possible since $\varepsilon \in [\![e^*]\!]_{\mathsf{KA}}$. Again, by induction, it follows $x \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$. Hence $x = \varepsilon \cdot x \in [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$.

3. Suppose $x = x_1x_2$ where $x_1 \in [\![e^*]\!]_{\mathsf{KA}}$ and $x_2y \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$. Therefore $x_2 \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$, and consequently $x = x_1x_2 \in [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$.

4. Finally, suppose $y = y_1y_2$ where $xy_1 \in [\![e^*]\!]_{\mathsf{KA}}$ and $y_2 \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$. Then there exists $n \in \mathbb{Z}_{\geq 0}$ such that $xy_1 \in [\![e^n]\!]_{\mathsf{KA}}$. This implies that $x \in [\![e^n]\!]^{pc}_{\mathsf{KA}}$. Now, $[\![e^n]\!]^{pc}_{\mathsf{KA}}$ is the smallest prefix-closed language containing $[\![e^n]\!]_{\mathsf{KA}}$, and $[\![(e^n)^{pc_1}]\!]_{\mathsf{KA}}$ is a prefix-closed language containing $[\![e^n]\!]_{\mathsf{KA}}$ by induction. Hence $x \in [\![(e^n)^{pc_1}]\!]_{\mathsf{KA}}$. But by Lemma 3.2.3, $[\![(e^n)^{pc_1}]\!]_{\mathsf{KA}} \subseteq [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$.

Hence $x \in [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$, and the proof is complete. □

We have shown now that $[\![e]\!]^{pc_1}_{\mathsf{KA}}$ is a subset of $[\![e^{pc_1}]\!]_{\mathsf{KA}}$, but we would like to have equality between the two phrases. The next Theorem proves exactly that.

Theorem 3.2.5. *Let $e \in \mathcal{T}_{\mathsf{KA}}$. Then we have $[\![e^{pc_1}]\!]_{\mathsf{KA}} = [\![e]\!]^{pc}_{\mathsf{KA}}$.*

Proof. Lemma 3.2.4 proves that $[\![e]\!]^{pc}_{\mathsf{KA}} \subseteq [\![e^{pc_1}]\!]_{\mathsf{KA}}$, so all that is left to prove is the reverse inclusion. Once again, we shall go by induction. It is clear that the statement holds for $[\![0^{pc_1}]\!]_{\mathsf{KA}}$ and $[\![1^{pc}]\!]_{\mathsf{KA}}$.

Now, look at the case $e = a$. As we know, $[\![a^{pc_1}]\!]_{\mathsf{KA}} = \{\varepsilon, a\}$. But $[\![a]\!]^{pc}_{\mathsf{KA}}$ is the smallest prefix-closed language containing a. Thus, both $\varepsilon \in [\![a]\!]^{pc}_{\mathsf{KA}}$ and $a \in [\![a]\!]^{pc}_{\mathsf{KA}}$.

Next, look at the case $e = e_1 + e_2$. Then $[\![(e_1 + e_2)^{pc_1}]\!]_{\mathsf{KA}} = [\![e_1^{pc_1}]\!]_{\mathsf{KA}} \cup [\![e_2^{pc_1}]\!]_{\mathsf{KA}}$. By induction, this is equal to $[\![e_1]\!]^{pc}_{\mathsf{KA}} \cup [\![e_2]\!]^{pc}_{\mathsf{KA}}$. The result follows from Theorem 3.1.4.

In the step $e = e_1 \cdot e_2$, we perform the necessary case analysis. The case where $Z(e_2)$ is trivial, so we focus on the case $NZ(e_2)$.

First, we recall that $[\![(e_1 \cdot e_2)^{pc_1}]\!]_{\mathsf{KA}} = [\![e_1^{pc_1}]\!]_{\mathsf{KA}} \cup [\![e_1]\!]_{\mathsf{KA}}[\![e_2^{pc_1}]\!]_{\mathsf{KA}}$. By induction, we then have that $[\![(e_1 \cdot e_2)^{pc_1}]\!]_{\mathsf{KA}} = [\![e_1]\!]^{pc}_{\mathsf{KA}} \cup [\![e_1]\!]_{\mathsf{KA}}[\![e_2]\!]^{pc}_{\mathsf{KA}}$.

Suppose $x \in [\![(e_1 \cdot e_2)^{pc_1}]\!]_{\mathsf{KA}}$. Now we have two cases:

First, suppose $x \in [\![e_1]\!]^{pc}$. Then there exists y such that $xy \in [\![e_1]\!]_{\mathsf{KA}}$. Then for any $z \in [\![e_2]\!]_{\mathsf{KA}}$, $xyz \in [\![e_1]\!]_{\mathsf{KA}}[\![e_2]\!]_{\mathsf{KA}} = [\![e_1 \cdot e_2]\!]_{\mathsf{KA}}$. Hence $x \in [\![e_1 \cdot e_2]\!]^{pc}$.

Second, suppose $x \in [\![e_1]\!]_{\mathsf{KA}}[\![e_2]\!]^{pc}_{\mathsf{KA}}$. Then $x = yz$ where $y \in [\![e_1]\!]_{\mathsf{KA}}$ and $z \in [\![e_2]\!]^{pc}_{\mathsf{KA}}$. Now, $z \in [\![e_2]\!]^{pc}_{\mathsf{KA}}$ implies that there exists z' such that $zz' \in [\![e_2]\!]_{\mathsf{KA}}$. Hence $xz' \in [\![e_1 \cdot e_2]\!]_{\mathsf{KA}}$, and therefore $x \in [\![e_1 \cdot e_2]\!]^{pc}_{\mathsf{KA}}$.

Finally, we look at the step e^* where $NZ(e)$. Again, the case $Z(e)$ is trivial. Let $x \in [\![(e^*)^{pc_1}]\!]_{\mathsf{KA}} = [\![e^*]\!]_{\mathsf{KA}}[\![e^{pc_1}]\!]_{\mathsf{KA}}$. We must show that there exists $y \in \Sigma^*$ such that $xy \in [\![e^*]\!]_{\mathsf{KA}}$. Of course, we have $x = x_1x_2$ where $x_1 \in [\![e^*]\!]_{\mathsf{KA}}$ and $x_2 \in [\![e^{pc_1}]\!]_{\mathsf{KA}}$. By induction, $[\![e^{pc_1}]\!]_{\mathsf{KA}} = [\![e]\!]^{pc}_{\mathsf{KA}}$. This implies that there exists $y \in \Sigma^*$ such that $x_2y \in [\![e]\!]_{\mathsf{KA}}$. Consequently $xy = x_1x_2y \in [\![e^*]\!]_{\mathsf{KA}}[\![e]\!]_{\mathsf{KA}} = [\![e^*e]\!]_{\mathsf{KA}} \subseteq [\![1 + e^*e]\!]_{\mathsf{KA}} = [\![e^*]\!]_{\mathsf{KA}}$. □

We see that the interpretation of this definition of the prefix-closure operator relying on case analysis using the nonzero predicate is equivalent to the prefix-closure at the level of regular languages. Alternatively, we can avoid the use of such a predicate by restricting the grammar so that 0-equivalent expressions cannot appears as suffix on a sequential composition. Therefore we define the following:

Definition 3.2.6. The set of *NZ-regular expressions* of a Kleene algebra is described by the grammar

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

where g is described by the grammar

$$g := 1 \mid a \in \Sigma \mid g + g \mid g \cdot g \mid g^*$$

Using this grammar prevents expressions of the form $e_1 \cdot e_2$, where e_2 is equivalent to zero according to the Kleene algebra axioms. Hence we may extend the grammar with the prefix-closure operator as we originally attempted to define it. That is, suppose $e, f \in \mathcal{T}_{\mathsf{NZ}}$. Then we define

$$\begin{array}{lll} 0^{pc_2} & 1^{pc_2} = 1 & a^{pc_2} = 1 + a \\ (e + f)^{pc_2} = e^{pc_2} + f^{pc_2} & (e \cdot f)^{pc_2} = e^{pc_2} + e \cdot f^{pc_2} & (e^*)^{pc_2} = e^* \cdot e^{pc_2} \end{array}$$

We may then define $[\![-]\!]_{\mathsf{NZ}} : \mathcal{T}_{\mathsf{NZ}} \to \mathcal{P}(\Sigma^*)$ exactly by $[\![e]\!]_{\mathsf{NZ}} = [\![e]\!]_{\mathsf{KA}}$ for any $e \in \mathcal{T}_{\mathsf{NZ}}$. Using this definition, we achieve the analogous results to Lemma 3.2.4 and Theorem 3.2.5. That is,

Lemma 3.2.7. *Let $e \in \mathcal{T}_{\mathsf{NZ}}$ be an NZ-regular expression. Then for any $xy \in [\![e^{pc_2}]\!]_{\mathsf{NZ}}$, we have that $x \in [\![e^{pc_2}]\!]_{\mathsf{NZ}}$.*

Theorem 3.2.8. *Let $e \in \mathcal{T}_{\mathsf{NZ}}$. Then $[\![e^{pc_2}]\!]_{\mathsf{NZ}} = [\![e]\!]^{pc}_{\mathsf{KA}}$.*

The proof of these results are largely analogous to the proof of Lemma 3.2.4. However, since we have changed the syntax, we must first prove the statement for the nonzero expressions g, and then for general expressions e.

We have now shown that the interpretation of the either definition of the prefix-closure operator is a prefix-closed language. However, to complete the axiomatization, we wish to show also that all prefix-closed regular languages are of the form $[\![e^{pc_1}]\!]_{\mathsf{KA}}$, or that all prefix-closed regular languages are of the form $[\![e^{pc_2}]\!]_{\mathsf{KA}}$ for some $e \in \mathcal{T}_{\mathsf{KA}}$ or $e \in \mathcal{T}_{\mathsf{NZ}}$, respectively.

We begin by looking at the case of NZ-regular expressions:

Theorem 3.2.9. *Let L be a prefix-closed regular language over an alphabet Σ. Then there exists $e \in \mathcal{T}_{\mathsf{NZ}}$ such that $[\![e^{pc_2}]\!] = L$.*

Proof. We know that there exists $e \in \mathcal{T}_{\mathsf{KA}}$ such that $[\![e]\!]_{\mathsf{KA}} = L$, as all regular languages come from a regular expression of Kleene algebras. Furthermore, we know that $\mathcal{T}_{\mathsf{NZ}} \subseteq \mathcal{T}_{\mathsf{KA}}$, so if it happens that $e \in \mathcal{T}_{\mathsf{NZ}}$, then we are done. It thus suffices to show that there exists $e' \in \mathcal{T}_{\mathsf{NZ}}$ such that $[\![e']\!]_{\mathsf{KA}} = [\![e]\!]_{\mathsf{KA}}$.

Going by induction, we look at the base cases $e = 0$, $e = 1$, and $e = a$. In each of these cases, e is already in $\mathcal{T}_{\mathsf{NZ}}$, so we may take $e' = e$.

Now let us assume $e = f + g$ where there exist $f', g' \in \mathcal{T}_{\mathsf{NZ}}$ such that $[\![f']\!]_{\mathsf{KA}} = [\![f]\!]_{\mathsf{KA}}$ and $[\![g']\!]_{\mathsf{KA}} = [\![g]\!]_{\mathsf{KA}}$. Then we may simply choose $e' = f' + g'$.

The case $e = f \cdot g$ is slightly more complicated, as we may have that $[\![g']\!]_{\mathsf{KA}} = \emptyset$, in which case $g' \equiv_{\mathsf{KA}} 0$, and therefore we cannot take $e' = f' \cdot g'$. However, we can take $e' = 0$. If, on the other hand, $[\![g']\!]_{\mathsf{KA}} \neq \emptyset$, then we can pick $e' = f' \cdot g'$.

Finally, if $e = f^*$, then we again have two possibilities. If $[\![f']\!]_{\mathsf{KA}} = \emptyset$, then $[\![f^*]\!]_{\mathsf{KA}} = \{\varepsilon\}$, so we must choose $e = 1$. On the other hand, if $[\![f']\!]_{\mathsf{KA}} \neq \emptyset$, then we pick $e' = (f')^*$. □

We have an analogous result for the operator pc_1.

Theorem 3.2.10. *Let L be a prefix-closed regular language over an alphabet Σ. Then there exists $e \in \mathcal{T}_{\mathsf{KA}}$ such that $[\![e^{pc_1}]\!]_{\mathsf{KA}} = L$.*

Proof. All regular languages are interpretations of regular expressions of Kleene algebras, so there exists $e \in \mathcal{T}_{\mathsf{KA}}$ such that $L = [\![e]\!]_{\mathsf{KA}}$. Then, because L is prefix-closed, we know that $[\![e]\!]^{pc}_{\mathsf{KA}} = L^{pc} = L$. By Theorem 3.2.5, we have that $[\![e]\!]^{pc}_{\mathsf{KA}} = [\![e^{pc_1}]\!]_{\mathsf{KA}}$, and we are finished. □

We also observe that the following results on prefix-closed regular languages follow from Theorem 2.3.3.

Theorem 3.2.11. *Let $e, f \in \mathcal{T}_{\mathsf{KA}}$. Then $e^{pc_1} \equiv_{\mathsf{KA}} f^{pc_1}$ if and only if $[\![e]\!]^{pc}_{\mathsf{KA}} = [\![f]\!]^{pc}_{\mathsf{KA}}$.*

Theorem 3.2.12. *Let $e, f \in \mathcal{T}_{\mathsf{NZ}}$. Then $e^{pc_2} \equiv_{\mathsf{NZ}} f^{pc_2}$ if and only if $[\![e]\!]^{pc}_{\mathsf{NZ}} = [\![f]\!]^{pc}_{\mathsf{NZ}}$.*

We are now ready to write a formal axiomatization.

3.3 Axiomatization of Prefix-Closed Regular Languages

Definition 3.3.1. A *prefix-closed Kleene algebra*, or *pc*-KA, is a tuple $(A, +, \cdot, *, 0, 1, N)$ that satisfies the following axioms for all $e, f, g \in A$:

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

As usual, we write $e \leq f$ for $e + f = f$ and require the *least fixpoint axioms*: That is, for all $e, f, g \in A$ we have

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

Additionaly, we also require the following axioms on the unary operator N:

$$\begin{array}{ll} N(0) = 0 & N(e + f) = N(e) + N(f) \\ N(1) = 1 & N(e \cdot f) = N(e) \cdot N(f) \qquad N(e^*) = 1 \end{array}$$

Finally, we require the *prefix-closure axiom:*

$$N(e) = 1 \implies 1 + e = e.$$

The new axioms are discovered from formalization of the NZ-predicate and the application of the prefix-closure operator to the axiom $e = 1 \cdot e$. Surprisingly, the prefix-closure axiom is the only new axiom that comes from the second source. Any other new identities arising from application of the prefix-closure operator to the rest of the Kleene algebra axioms follow directly from this axiom and the existing Kleene algebra axioms. For example,

Proposition 3.3.2. *Let $(A, +, \cdot, *, 0, 1, N)$ be a pcKA. Then for any $e, f \in A$,*

$$N(f) = 1 \implies e + e \cdot f = e \cdot f.$$

Proof. Let $e, f \in A$ with $N(f) = 1$. Then $e + e \cdot f = e \cdot (1 + f) = e \cdot f$. □

Before we can define the grammar for prefix-closed regular expressions, we define an intermediate set of expressions, the N-regular expressions, $\mathcal{T}_N$. These are given by the grammar

$$\mathcal{T}_N \ni e, f := 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid N(e)$$

The regular expressions of a prefix-closed Kleene algebra are denoted $\mathcal{T}_{pc}$ and described by the grammar

$$\mathcal{T}_{pc} \ni p := 0 \mid 1 + e$$

where $e \in \mathcal{T}_N$. We see that every prefix-closed regular expression is also an N-regular expression, meaning $\mathcal{T}_{pc} \subseteq \mathcal{T}_N$.

As before we define $\equiv_N$ by the smallest equivalence on $\mathcal{T}_N$ by the prefix-closed Kleene Algebra axioms. However, we also require that $N(a) \equiv_N 0$ for all $a \in \Sigma$. Then, since $\mathcal{T}_{pc} \subseteq \mathcal{T}_N$, we define $\equiv_{pc}$ by $e \equiv_{pc} f$ if and only if $e \equiv_N f$ for all $e, f \in \mathcal{T}_{pc}$. We will then exploit a connection between prefix-closed regular expressions and regular expressions to achieve the following:

Theorem 3.3.3. *For any $p \in \mathcal{T}_{pc}$ there exists $p_{\mathsf{KA}} \in \mathcal{T}_{\mathsf{KA}}$ such that $p \equiv_{pc} p_{\mathsf{KA}}$.*

Proof. We shall go by induction on p. If $p = 0$, then we may simply take $p_{\mathsf{KA}} = p$.

Now suppose $p = 1 + e$ for $e \in \mathcal{T}_N$. We will show by another induction that there exists a regular expression e_{KA} such that $e \equiv_N e_{\mathsf{KA}}$.

First, we may look at the new base cases. Suppose $e = 0$ or $e = 1$, or $e = a$. Then e is already a regular expression, so we may take $e_{\mathsf{KA}} = e$.

Furthermore, in the case $e = N(f)$, we know that $N(f) \equiv_N 0$ or $N(f) \equiv_N 1$ for any $f \in \mathcal{T}_N$. Thus we may choose $e_{\mathsf{KA}} = 0$ or $e_{\mathsf{KA}} = 1$ accordingly.

Now, assume $e = f + g$ where there exist two regular expressions f_{KA} and g_{KA} such that $f \equiv_N f_{\mathsf{KA}}$ and $g \equiv_N g_{\mathsf{KA}}$. Then certainly $e \equiv_N f_{\mathsf{KA}} + g_{\mathsf{KA}}$, so we may choose $e_{\mathsf{KA}} = f_{\mathsf{KA}} + g_{\mathsf{KA}}$.

The case $e = f \cdot g$ is exactly analogous, so we choose $e_{\mathsf{KA}} = f_{\mathsf{KA}} \cdot g_{\mathsf{KA}}$.

Finally, we look at the case $e = f^*$. Then $f \equiv_N f_{\mathsf{KA}}$ implies $e \equiv_N f^*_{\mathsf{KA}}$, so we pick $e_{\mathsf{KA}} = f^*_{\mathsf{KA}}$.

Hence, for any $e \in \mathcal{T}_N$, there exists e_{KA} such that $e \equiv_N e_{\mathsf{KA}}$. This means that $1 + e_{\mathsf{KA}}$ is also a regular expression, and $1 + e \equiv_N 1 + e_{\mathsf{KA}}$. It therefore follows that $1 + e \equiv_{pc} 1 + e_{\mathsf{KA}}$, and our outer induction is complete. □

As in the previous cases, we define our interpretation in terms of languages as a map $[\![-]\!]_N : \mathcal{T}_N \to \mathcal{P}(\Sigma^*)$ defined inductively by

$$
\begin{array}{lll}
[\![0]\!]_N = \emptyset & [\![a]\!]_N = \{a\} & [\![e \cdot f]\!]_N = [\![e]\!]_N \cup [\![e_{\mathsf{KA}}]\!]_{\mathsf{KA}} \cdot [\![f]\!]_N \text{ if } N((f_{\mathsf{KA}})^{pc}) = 1 \\
[\![1]\!]_N = \{\varepsilon\} & [\![e + f]\!]_N = [\![e]\!]_N \cup [\![f]\!]_N & [\![e^*]\!]_N = [\![e_{\mathsf{KA}}]\!]^*_{\mathsf{KA}} [\![1 + e]\!]_N \text{ if } N((e_{\mathsf{KA}})^{pc}) = 1 \\
[\![N(e)]\!]_N = [\![e]\!]_N \cap \{\varepsilon\} & &
\end{array}
$$

where we also have $[\![e \cdot f]\!]_N = \emptyset$ if $N((f_{\mathsf{KA}})^{pc}) = 0$ and $[\![e^*]\!]_N = \{\varepsilon\}$ if $N((e_{\mathsf{KA}})^{pc}) = 0$.

Now, this is not quite good enough for our purposes. It is easy to see that not every language given by this function is prefix closed. For example, $[\![a]\!]_N = \{a\}$ is not prefix-closed. Furthermore, the requirement $N(a) \equiv_N 0$ means that the N-operator loses its ability to capture equivalence to zero on N-regular expressions.

To remedy this, we instead work on $\mathcal{T}_{pc}$, and we define $[\![-]\!]_{pc} : \mathcal{T}_{pc} \to \mathcal{P}(\Sigma^*)$ by

$$[\![0]\!]_{pc} = \emptyset \qquad [\![1 + e]\!]_{pc} = [\![1 + e]\!]_N$$

En route to soundness and completeness results, we wish to show that we are justified in using the term "prefix-closed" to describe prefix-closed Kleene algebras and their regular expressions. To this end, we have the following property:

Proposition 3.3.4. *For all $p \in \mathcal{T}_{pc}$, $(p_{\mathsf{KA}})^{pc} \equiv_{pc} p$.*

Proof. This is certainly true in the case $p = 0$, as then $(p_{\mathsf{KA}})^{pc} = 0$. From here we need only show that for any $e \in \mathcal{T}_N$, $(1 + e_{\mathsf{KA}})^{pc} \equiv_N 1 + e$. Then the result follows by induction and the definition of $\equiv_{pc}$.

We proceed again with an intermediate induction. The cases $e = 0$ and $e = 1$ are trivial, as in these cases $(e_{\mathsf{KA}})^{pc} = e_{\mathsf{KA}} = e$. In the case $e = a$, we have also $e_{\mathsf{KA}} = e = a$. But now we have also $(1 + a)^{pc} = 1 + (1 + a)$. Through associativity and idempotence of addition, we have $(1 + a)^{pc} \equiv_N 1 + a$.

In the case $e = N(f)$, we know $(N(f))_{\mathsf{KA}} \equiv_N 0$ or $(N(f))_{\mathsf{KA}} \equiv_N 1$, so $(1+(N(f))_{\mathsf{KA}})^{pc} \equiv_N 1 \equiv_N 1 + N(f)$.

Now suppose $e = f + g$ where $(1 + f\mathsf{KA})^{pc} \equiv_N 1 + f$ and $(1 + g_{\mathsf{KA}})^{pc} \equiv_N 1 + g$. Then $1 + f + g \equiv_N (1 + f) + (1 + g)$ by idempotence, commutativity, and associativity, and hence $1+f+g \equiv_N (1+f_{\mathsf{KA}})^{pc}+(1+g_{\mathsf{KA}})^{pc} \equiv_N (1+1+f_{\mathsf{KA}}+g_{\mathsf{KA}})^{pc} \equiv_N (1+f_{\mathsf{KA}}+g_{\mathsf{KA}})^{pc}$.

In the case $e = f \cdot g$, we have two possibilities. The case $N((g_{\mathsf{KA}})^{pc}) = 0$ reduces to a trivial case, so we need only look at the case $N((g_{\mathsf{KA}})^{pc}) = 1$. Then $(1 + f_{\mathsf{KA}} \cdot g_{\mathsf{KA}})^{pc} = 1 + (f_{\mathsf{KA}})^{pc} + f_{\mathsf{KA}} \cdot (g_{\mathsf{KA}})^{pc}$. By the inductive hypothesis, this is N-equivalent to $1+f+f_{\mathsf{KA}} \cdot (g_{\mathsf{KA}})^{pc} \equiv_N 1+f_{\mathsf{KA}}+f_{\mathsf{KA}} \cdot (g_{\mathsf{KA}})^{pc}$ by the definition of f_{KA}. We can then apply distributivity to get the N-equivalent expression $1+f_{\mathsf{KA}} \cdot (1+(g_{\mathsf{KA}})^{pc}) \equiv_N 1+f_{\mathsf{KA}} \cdot (1+g_{\mathsf{KA}})$. But then we know that $N(g) = 1$, so it follows that $g \equiv_N 1 + g \equiv_N 1 + g_{\mathsf{KA}} \equiv_N$. We can also substitute f for f_{KA} to obtain $(1 + f_{\mathsf{KA}} \cdot g_{\mathsf{KA}})^{pc} \equiv_N 1 + f \cdot g$.

Finally, we look at the case $e = f^*$. Once again, the case $N((f_{\mathsf{KA}})^{pc}) \equiv_N 0$ reduces to a trivial case, so we can assume $N((f_k a)^{pc}) \equiv_N 1$. By the inductive hypothesis, we also have that $(1 + f_{\mathsf{KA}})^{pc} \equiv_N 1 + f$. But then we apply the prefix-closure axiom to see that $(f_{\mathsf{KA}})^{pc} \equiv_N f$. Beginning with $(1 + (f_{\mathsf{KA}})^*)^{pc} = 1 + (f_{\mathsf{KA}})^* \cdot (f_{\mathsf{KA}})^{pc}$, we then obtain the N-equivalent expression $1 + (f)^* \cdot f \equiv_N fa^* \equiv_N 1 + f^*$ through the star axiom and the prefix-closure axiom, since $N(f^*) = 1$. □

The above property will aid us in our attempt to prove soundness and completeness.

Theorem 3.3.5. *(Soundness and Completeness of pc). For all $p, q \in \mathcal{T}_{pc}$, $p \equiv_{pc} q$ if and only if $[\![p]\!]_{pc} = [\![q]\!]_{pc}$.*

Before we proceed, however, we will present the following lemma:

Lemma 3.3.6. *Let $p, q \in \mathcal{T}_{pc}$. Then,*

1. *$p \equiv_{pc} q$ if and only if $(p_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (q_{\mathsf{KA}})^{pc}$*
2. *$[\![p]\!]_{pc} = [\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$.*

Proof. To prove the first implication of 1., let us first suppose that $p, q \in \mathcal{T}_{pc}$ with $p \equiv_{pc} q$. If $p = 0$, then the only option is $p = q = 0$, which in turn implies that $(p_{\mathsf{KA}})^{pc} = (q_{\mathsf{KA}})^{pc}$,

so of course $(p_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (q_{\mathsf{KA}})^{pc}$. The case of $p = 1$ is also easy. Now we may assume $p = 1 + e$ for some N-regular expression e. Thus $q = 1 + f$ for some N-regular expression f (or $q = 1$, and we return to the previous case). We have $1 + e \equiv_{pc} 1 + f$, and hence also $1 + e \equiv_N 1 + f$, since $\equiv_{pc}$ is just the restriction of $\equiv_N$ to the prefix-closed regular expressions.

Now, by Theorem 3.3.3, we know that $1 + e_{\mathsf{KA}} \equiv_N 1 + f_{\mathsf{KA}}$ as well. Let us first look at the cases where $N((e_{\mathsf{KA}})^{pc}) \equiv_N 0$.

The only possibility is $e_{\mathsf{KA}} \equiv_N 0$. In this case $1 + f_{\mathsf{KA}} \equiv_N 1$, so either $f_{\mathsf{KA}} \equiv_N 0$ or $f_{\mathsf{KA}} \equiv_N 1$. There are no "new" ways for a regular expression to be N-equivalent to 0 or 1, so $1 + e_{\mathsf{KA}} \equiv_{\mathsf{KA}} 1 + 0 \equiv_{\mathsf{KA}} 1 \equiv_{\mathsf{KA}} 1 + f_{\mathsf{KA}}$. Hence, $(1 + e_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} 1 \equiv_{\mathsf{KA}} (1 + f_{\mathsf{KA}})^{pc}$.

Now we look at the cases where $N((e_{\mathsf{KA}})^{pc}) \equiv_N 1$.

The only possible case where we have $N(f_{\mathsf{KA}}) = 0$ and $1+e_{\mathsf{KA}} \equiv_N 1+f_{\mathsf{KA}}$ is $e_{\mathsf{KA}} \equiv_N 1$ and $f_{\mathsf{KA}} \equiv_N 0$. Then $1 + e_{\mathsf{KA}} \equiv_N 1 \equiv_N 1 + f_{\mathsf{KA}}$. The prefix-closure axiom does not introduce any ways to be N-equivalent to 1 that are not also KA-equivalent, so we may conclude $1 + e_{\mathsf{KA}} \equiv_{\mathsf{KA}} 1 \equiv_{\mathsf{KA}} 1 + f_{\mathsf{KA}}$. Hence, the result follows.

Now we may assume that $N(f_{\mathsf{KA}}) \equiv_N 1$. This implies through the prefix-closure axiom that $e_{\mathsf{KA}} \equiv_N f_{\mathsf{KA}}$. We inductively look at all the ways this equivalence can occur.

Firstly, $e_{\mathsf{KA}} = f_{\mathsf{KA}}$. Then certainly $(1 + e_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (1 + f_{\mathsf{KA}})^{pc}$, since the two expressions are equal.

Then, we look at the $pc\mathsf{KA}$ axioms to determine how else the expressions may be N-equivalent. For example, we may have $e_{\mathsf{KA}} = e_1 + e_2$ and $f_{\mathsf{KA}} = f_1 + f_2$, where $e_1 \equiv_{\mathsf{KA}} f_2$, and vice-versa. This case aligns with the Kleene algebra axioms, so certainly $e_{\mathsf{KA}} \equiv_{\mathsf{KA}} f_{\mathsf{KA}}$, and consequently the result holds. In fact, the result will hold in this fashion if $e_{\mathsf{KA}} \equiv_N f_{\mathsf{KA}}$ due to any of the first ten axioms or the least fixpoint axioms, as these are also KA axioms.

We have assumed that e_{KA} and f_{KA} are regular expressions given by Theorem 3.3.3, so we cannot have N-equivalence due to any of the N-axioms. Thus, the only other possible case is $e_{\mathsf{KA}} = g + f_1$ where $g \equiv_{\mathsf{KA}} 1$ and $f_1 \equiv_{\mathsf{KA}} f_{\mathsf{KA}}$. Hence $(1+e_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (1+1+f_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (1 + f_{\mathsf{KA}})^{pc}$, and the first implication is done.

In the other direction, suppose that $(p_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (q_{\mathsf{KA}})^{pc}$. Then certainly $(p_{\mathsf{KA}})^{pc} \equiv_{pc} (q_{\mathsf{KA}})^{pc}$, since every Kleene algebra axiom is also a $pc\mathsf{KA}$ axiom, and by Proposition 3.3.4 and transitivity of equivalence relations, $p \equiv_{pc} q$.

We now prove the second claim. First, if $e = 0$, then $(p_{\mathsf{KA}})^{pc} = 0$, and $[\![0]\!]_{pc} = \emptyset = [\![0]\!]_{\mathsf{KA}}$. Going forward we assume $p = 1 + e$ where $e \in \mathcal{T}_N$ and prove the claim by induction.

Let $e = 0$. Then $p = 1 + 0$, so $p_{\mathsf{KA}} \equiv_{\mathsf{KA}} 1$. Then $[\![1 + 0]\!]_{pc} = [\![1]\!]_N = \{\varepsilon\} = [\![1]\!]_{\mathsf{KA}}$.

The case $e = 1$ follows by identical reasoning.

Let $e = a \in \Sigma$. Then $p = 1 + a$ and $p_{\mathsf{KA}} = 1 + a$. Next, $[\![1 + a]\!]_{pc} = \{\varepsilon, a\} = [\![(1 + a)^{pc}]\!]_{\mathsf{KA}}$.

Let $e = N(f)$ for some N-regular expression f. Then $e = 1 + N(f)$ and $p_{\mathsf{KA}} \equiv_{\mathsf{KA}} 1$. Now, $[\![1 + N(f)]\!]_{pc} = \{\varepsilon\} \cup ([\![f]\!]_N \cap \{\varepsilon\}) = \{\varepsilon\} = [\![1]\!]_{\mathsf{KA}}$.

We now prove by induction that whenever $N(e) \equiv_N 1$, we have $[\![e]\!]_N = [\![(e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$. From this and the definition of the pc-operator, it will certainly follow that $[\![1 + e]\!]_{pc} = [\![(1 + e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$.

We have already shown this holds in the case $e = 1$, and in the case $e = N(f) \equiv_N 1$. Thus we may begin by assuming $e = f + g$. We know that either $N(f) = 1$ or $N(g) = 1$. Let us assume $N(f) = 1$. Then if $N(g) = 0$, we may simply take instead $g_1 = 1 + g$, as the prefix-closure axiom ensures $e \equiv_N f + g_1$ and we can also see that $[\![f + g]\!]_N = [\![f + g_1]\!]_N$.

Thus we may assume $N(g) = 1$ as well, and by the inductive hypothesis, $[\![f]\!]_N = [\![(f_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$ and $[\![g]\!]_N = [\![(g_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$. Hence $[\![e]\!]_N = [\![f]\!]_N + [\![g]\!]_N = [\![(f_{\mathsf{KA}} + g_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$.

Now take the case $e = f \cdot g$. In this case we must have $N(f) = 1 = N(g)$, we already know we can apply the inductive hypothesis. Thus $[\![e]\!]_N = [\![(f_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} \cup [\![f_{\mathsf{KA}}]\!]_{\mathsf{KA}} \cdot [\![(g_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![(f_{\mathsf{KA}} \cdot g_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$.

Finally, we suppose $e = f^*$. If $N(f) \equiv_N 0$, then either $f \equiv_N 0$ or $f \equiv_N a$. In the former case, we are reduced to the case $e \equiv_N 1$, which we have seen already. In the latter case, we obtain $[\![a^*]\!]_N = [\![a^*]\!]_{\mathsf{KA}}[\![1 + a]\!]_N = [\![a^*]\!]_{\mathsf{KA}} = [\![a^*]\!]_{\mathsf{KA}}[\![a]\!]_{\mathsf{KA}} = [\![(a^*)^{pc}]\!]_{\mathsf{KA}}$.

Next we assume that $N(f) \equiv_N 1$. Then by the inductive hypothesis $[\![f]\!]_N = [\![(f_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$. Hence we have $[\![e]\!]_N = [\![(f_{\mathsf{KA}})^*]\!]_{\mathsf{KA}} \cdot [\![1 + f]\!]_N = [\![(f_{\mathsf{KA}})^*]\!]_{\mathsf{KA}} \cup [\![(f_{\mathsf{KA}})^*]\!]_{\mathsf{KA}} \cdot [\![(f_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![((f_{\mathsf{KA}})^*)^{pc}]\!]_{\mathsf{KA}}$. □

Then Theorem 3.3.5 follows from a direct application of the lemma and Theorem 3.2.11 as follows:

Proof. (of Theorem 3.3.5.)
Let $p, q \in \mathcal{T}_{pc}$. Then by Lemma 3.3.6 (1), $p \equiv_{pc} q$ if and only if $(p_{\mathsf{KA}})^{pc} \equiv_{\mathsf{KA}} (q_{\mathsf{KA}})^{pc}$. By Theorem 3.2.11, this happens if and only if $[\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![(q_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$. But by Lemma 3.3.6 (2), $[\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![p]\!]_{pc}$ and $[\![(q_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![q]\!]_{pc}$, so we have that $[\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![(q_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$ if and only if $[\![p]\!]_{pc} = [\![q]\!]_{pc}$. Hence $pc\mathsf{KA}$ is sound and complete with respect to the prefix-closed regular expressions. □

Chapter 4

Synchronous Languages

4.1 The Synchronous Product and F_1-Algebras

Synchronous languages are used to denote executions of synchronous process, where one or more actions take place concurrently in a stepwise manner.

Definition 4.1.1. Given an alphabet Σ we write $\mathcal{P}_n(\Sigma) := \mathcal{P}(\Sigma) \setminus \{\emptyset\}$ and say that words over $\mathcal{P}_n(\Sigma)$ are called *synchronous strings*. Sets of synchronous strings form *synchronous languages*.

Just as for general languages as we have seen and used above, the operations of union, concatenation, and Kleene star are also defined on synchronous language. We additionally define a new operation on synchronous languages by the following:

Definition 4.1.2. Let L_1, L_2 be two synchronous languages. We define the *synchronous product* of L_1 and L_2 by

$$L_1 \times L_2 = \{u \times v \mid u \in L_1 \wedge v \in L_2\}$$

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

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

The synchronous product satisfies the following properties:

Proposition 4.1.3. *Let L_1, L_2, and L_3 be synchronous languages over the same alphabet.*

1. $L_1 \times \emptyset = \emptyset$,
2. $L_1 \times \{\varepsilon\} = L_1$,
3. $L_1 \times L_2 = L_2 \times L_1$, *and*
4. $(L_1 \times L_2) \times L_3 = L_1 \times (L_2 \times L_3)$.

Proof. Suppose L_1, L_2, and L_3 are synchronous languages over the same alphabet.

We first want to show that $\emptyset$ annihilates the synchronous product. $L_1 \times \emptyset = \{u \times v \mid u \in L_1 \wedge v \in \emptyset\}$. Since $v \in \emptyset$ never holds, $L_1 \times \emptyset = \emptyset$.

Second, $L_1 \times \{\varepsilon\} = \{u \times \varepsilon \mid u \in L_1\} = L_1$.

Third, it suffices to show that $u \times v = v \times u$ at the level of synchronous words. We can do this by a quick induction. For the base case, $u \times \varepsilon = u = \varepsilon \times u$. Now let us assume $u \times v = v \times u$. Then we must show $(x \cdot u) \times (y \cdot v) = (y \cdot v) \times (x \cdot u)$ for $x, y \in \mathcal{P}_n(\Sigma)$. But we have $(x \cdot u) \times (y \cdot v) = (x \cup y) \cdot (u \times v) = (y \cup x) \cdot (v \times u) = (y \cdot v) \times (x \cdot u)$.

Finally, take $(L_1 \times L_2) \times L_3 = \{(u \times v) \times w \mid u \in L_1 \wedge v \in L_2 \wedge w \in L_3\}$. Once again, it therefore suffices to show associativity at the level of words. And, once again, the technique is induction. Certainly the base case of the empty word holds. Assume $(u \times v) \times w = u \times (v \times w)$. Now take $((x \cdot u) \times (y \cdot v)) \times (z \cdot w)$, where $x, y, z \in \mathcal{P}_n(\Sigma)$. This is equal to $((x \cup y) \cup z) \cdot ((u \times v) \times w) = (x \cup (y \cup z) \cdot (u \times (v \times w))$ using associativity of union and the inductive hypothesis. This yields $((x \cdot u) \times (y \cdot v)) \times (z \cdot w) = (x \cdot u) \times ((y \cdot v) \times (z \cdot w))$. □

Before we embark on our endeavour of finding a an axiomatization for synchronous regular languages using "synchronous Kleene algebras," we must first define a new – but related – algebra, which provides some stronger axioms that will serve as the foundation for our synchronous algebra.

Definition 4.1.4. An F_1-algebra [12] is a tuple $(A, +, \cdot, *, 0, 1, H)$ where A is a set, $+, \cdot$ are binary operators, $*$ is a unary operator, and 0 and 1 are constants such that for all $e, f, g \in A$ the following axioms are satisfied:

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

Additionally, H is a unary operator that satisfies the axioms:

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

And finally, the *loop tightening* and *unique fixpoint axiom* hold:

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

For $e \in A$, whenever $H(e) = 1$, e is said to have the *empty word property*. The set of F_1-expressions, the grammar for F_1-algebras, denoted $\mathcal{T}_{\mathsf{F}_1}$, is described by:

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

Note that the F_1-expressions are exactly the prefix-closed regular expressions, with the exception that we write H instead of N. Clearly they do not need to satisfy the prefix

closure axiom in general. One can interpret F_1 expressions in terms of languages via $[\![-]\!]_{\mathsf{F}_1} : \mathcal{T}_{\mathsf{F}_1} \to \mathcal{P}(\Sigma)$, defined analogously to $[\![-]\!]_{\mathsf{KA}}$, where for any $e \in \mathcal{T}_{\mathsf{F}_1}$, we have

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

Again, analogous to the Kleene algebra case, we write $\equiv_{\mathsf{F}_1}$ for the smallest congruence on $\mathcal{T}_{\mathsf{F}_1}$ induced by the F_1-axioms. We also require that $H(a) \equiv_{\mathsf{F}_1} 0$ for any $a \in \Sigma$. Then we can achieve analogous soundness and completeness results to those of Theorem 2.3.3.

Theorem 4.1.5. *(Soundness and Completeness of F_1 [12]). For all $e, f \in \mathcal{T}_{\mathsf{F}_1}$, we have that $e \equiv_{\mathsf{F}_1} f$ if and only if $[\![e]\!]_{\mathsf{F}_1} = [\![f]\!]_{\mathsf{F}_1}$.*

F_1-algebras were first proposed by Salomaa and their axioms are strictly stronger than the Kleene algebra axioms due to the unique fixpoint axiom. However, we see that all prefix-closed Kleene algebras as defined in the previous chapter correspond exactly to an F_1-algebra.

Theorem 4.1.6. *Let $\mathbf{A} = (A, +, \cdot, *, 0, 1, N)$ be a prefix-closed Kleene algebra. Then $\mathbf{A}$ is an F_1-algebra.*

Proof. The first ten F_1 algebra axioms are exactly identical to the first ten prefix-closed Kleene algebra axioms. We need only show that $\mathbf{A}$ satisfies the loop tightening, unique fixpoint, and H axioms. We begin with the axioms for H.

In fact, the only H axiom that is not identical to the N axioms of prefix-closed Kleene algebras is the axiom $H(e^*) = (H(e))^*$. We must therefore show $N(e^*) = (N(e))^*$. But $N(e^*) = 1$, and either $N(e) = 0$ or $N(e) = 1$, so $(N(e))^* = 1$.

By the prefix-closure axioms, $e + 1 = e$ whenever $N(e) = 1$, so certainly $(e + 1)^* = e^*$ in this case. Furthermore, we have already stated that $0^* = 1^* = 1$, so the loop tightening axiom holds for any $e \in A$.

Finally, assume that we have $e, f, g \in A$ such that $N(f) = 0$ and $e + f \cdot g = g$. We know already that $N(f) = 0$ implies $f = 0$. Thus we have also $f^* = 1$ and $f \cdot g = 0$. This implies $e = g$, which of course implies $f^* \cdot e = 1 \cdot e = g$, and we are finished. □

Now, since Salomaa's axiomatization is strictly stronger, we know that every F_1-algebra is also a Kleene algebra. This provides us with the intuition that our axiomatization of prefix-closed Kleene algebras may, in fact, be an axiomatization of prefix-closed F_1-algebras.

4.2 An Axiomatization of Prefix-Closed F_1-Algebras

In analogy to the result for prefix-closed regular expressions, we obtain the result

Theorem 4.2.1. *For any $e \in \mathcal{T}_{\mathsf{F}_1}$ there exists $e_{\mathsf{KA}} \in \mathcal{T}_{\mathsf{KA}}$ such that $e \equiv_{\mathsf{F}_1} e_{\mathsf{KA}}$.*

The proof follows exactly the proof of Theorem 3.3.3. We also note the following link between regular languages and F_1-languages.

Proposition 4.2.2. *Let $e_{\mathsf{KA}} \in \mathcal{T}_{\mathsf{KA}}$. Then $[\![e_{\mathsf{KA}}]\!]_{\mathsf{F}_1} = [\![e_{\mathsf{KA}}]\!]_{\mathsf{KA}}$.*

Proof. The proof is trivial, as the interpretation $[\![-]\!]_{\mathsf{F}_1}$ of F_1-expressions is defined exactly as the interpretation $[\![-]\!]_{\mathsf{KA}}$ of regular expressions, except in the case of terms of the form $H(e)$, which do not exist as regular expressions. □

This provides us with the tools needed to easily prove the next result.

Theorem 4.2.3. *1. Let $e \in \mathcal{T}_{\mathsf{F}_1}$. Then $[\![e]\!]_{\mathsf{F}_1}$ is prefix-closed if and only if there exists $e' \in \mathcal{T}_{pc}$ such that $[\![p]\!]_{pc} = [\![e]\!]_{\mathsf{F}_1}$.*

2. Let $p \in \mathcal{T}_{pc}$. Then there exists $e \in \mathcal{T}_{\mathsf{F}_1}$ such that $[\![p]\!]_{pc} = [\![e]\!]_{\mathsf{F}_1}$

Proof. First, let us assume that $e \in \mathcal{T}_{\mathsf{F}_1}$ such that $[\![e]\!]_{\mathsf{F}_1}$ is prefix-closed. Then, by Theorem 4.2.1 there exists e_{KA} such that $e \equiv_{\mathsf{F}_1} e_{\mathsf{KA}}$. By Theorem 4.1.5, this implies that $[\![e]\!]_{\mathsf{F}_1} = [\![e_{\mathsf{KA}}]\!]_{\mathsf{F}_1}$, which by Proposition 4.2.3, is equal to $[\![e_{\mathsf{KA}}]\!]_{\mathsf{KA}}$. But then $[\![e_{\mathsf{KA}}]\!]_{\mathsf{KA}}$ is prefix-closed, so $[\![e_{\mathsf{KA}}]\!]_{\mathsf{KA}} = [\![(e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$.

Now, we have two possibilities: First, $[\![(e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = \emptyset$. Then we may pick $p = 0$. Second, $[\![(e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} \neq \emptyset$. In this case, we have $\varepsilon \in [\![(e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$, which implies $[\![(e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![(1 + e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$. We also have $1 + e_{\mathsf{KA}} \in \mathcal{T}_{pc}$ with $(1 + e_{\mathsf{KA}})_{\mathsf{KA}} = 1 + e_{\mathsf{KA}}$. Hence, by Lemma 3.3.6 (2), $[\![(1 + e_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![1 + e_{\mathsf{KA}}]\!]_{pc}$. Hence, if we take $p = 1e_{\mathsf{KA}}$, we achieve $[\![e]\!]_{\mathsf{F}_1} = [\![p]\!]_{pc}$. The converse is certainly true because $[\![p]\!]_{pc}$ must be prefix-closed.

To prove the second item, now suppose $p \in \mathcal{T}_{pc}$. Then by the results of section 3.3, there exists $p_{\mathsf{KA}} \in \mathcal{T}_{\mathsf{KA}}$ such that $[\![p]\!]_{pc} = [\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}}$. But by Proposition 4.2.2, $[\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{KA}} = [\![(p_{\mathsf{KA}})^{pc}]\!]_{\mathsf{F}_1}$. Hence, if we take $e = p_{\mathsf{KA}}$, we are done. □

Taken together, the results of the above theorem serve to say that every prefix-closed F_1-language is also a prefix-closed regular language, and vice-versa. That is, the axiomatization of prefix-closed Kleene algebras from the previous section is precisely the correct axiomatization for the prefix-closed F_1-algebras, and absolutely nothing new is needed.

4.3 The Synchronous F_1-Algebra

One may ask why we bother with these new definitions, and why the name "synchronous" befits them. The name follows from the work of [10] whose motivation "spawns from the need to represent and reason about actions which can be performed 'at the same time."' However, the axiomatization of Synchronous Kleene Algebra in this work, while sound, is incomplete. Indeed, a counterexample and a new axiomatization, which is complete, are given in [14], and it is this axiomatization and terminology that we follow.

Definition 4.3.1. A *synchronous F_1-algebra*, or SF_1-algebra, is a tuple $(A, S, +, \cdot, *, \times, 0, 1, H)$ such that $(A, +, \cdot, *, 0, 1, H)$ is an F_1-algebra and $\times$ is a binary operator on A, with $S \subseteq A$ closed under $\times$ and $(S, \times)$ a semilattice. Furthermore, the following hold for all $e, f, g \in A$ and $e, f \in S$:

$$\begin{array}{lll} e \times (f + g) = e \times f + e \times g & e \times (f \times g) = (e \times f) \times g & e \times 0 = 0 \\ (e \cdot e) \times (f \cdot f) = (e \times f) \cdot (e \times f) & e \times f = f \times e & e \times 1 = e. \end{array}$$

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

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

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

where the *semilattice terms*, denoted $\mathcal{T}_{\mathsf{SL}}$, are given by the grammar

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

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

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

Again, denote the smallest congruence on $\mathcal{T}_{\mathsf{SL}}$ with respect to idempotence, associativity, and commutativity of $\times$ with $\equiv_{\mathsf{SL}}$. This once again allows the following result:

Lemma 4.3.2. *(Soundness and Completeness of SL). For all $e, f \in \mathcal{T}_{\mathsf{SL}}$, we have $[\![e]\!]_{\mathsf{SL}} = [\![f]\!]_{\mathsf{SL}}$ if and only if $e \equiv_{\mathsf{SL}} f$.*

We interpret $\mathcal{T}_{\mathsf{SF}_1}$ in terms of languages via $[\![-]\!]_{\mathsf{SF}_1} : \mathcal{T}_{\mathsf{SF}_1} \to \mathcal{P}((\mathcal{P}_n(\Sigma))^*)$ defined by

$$\begin{array}{lll} [\![0]\!]_{\mathsf{SF}_1} = \emptyset & [\![1]\!]_{\mathsf{SF}_1} = \{\varepsilon\} & [\![a]\!]_{\mathsf{SF}_1} = \{[\![a]\!]_{\mathsf{SL}}\} \\ [\![e^*]\!]_{\mathsf{SF}_1} = [\![e]\!]^*_{\mathsf{SF}_1} & [\![e \cdot f]\!]_{\mathsf{SF}_1} = [\![e]\!]_{\mathsf{SF}_1} \cdot [\![f]\!]_{\mathsf{SF}_1} & [\![e + f]\!]_{\mathsf{SF}_1} = [\![e]\!]_{\mathsf{SF}_1} \cup [\![f]\!]_{\mathsf{SF}_1} \\ [\![e \times f]\!]_{\mathsf{SF}_1} = [\![e]\!]_{\mathsf{SF}_1} \times [\![f]\!]_{\mathsf{SF}_1} & [\![H(e)]\!]_{\mathsf{SF}_1} = [\![e]\!]_{\mathsf{SF}_1} \cap \{e\} & \end{array}$$

As one might expect, we define $\equiv_{\mathsf{SF}_1}$ to be the smallest congruence on $\mathcal{T}_{\mathsf{SF}_1}$ induced by the axioms of SF_1, where $\mathcal{T}_{\mathsf{SL}}$ fulfills the role of the semilattice. Then a theorem from [14] says the following:

Theorem 4.3.3. *(Soundness and Completeness of SF_1.) For all $e, f \in \mathcal{T}_{\mathsf{SF}_1}$, we have $[\![e]\!]_{\mathsf{SF}_1} = [\![f]\!]_{\mathsf{SF}_1}$ if and only if $e \equiv_{\mathsf{SF}_1} f$.*

This proof is not at all trivial, particularly in the aspect of completeness, and the use of F_1-algebras as the foundation remedies the incompleteness of the axiomatization [10] by using the unique fixpoint axiom in lieu of the least fixpoint axiom to connect the synchronous product and the Kleene star.

Chapter 5

Prefix-Closure on Synchronous Languages

The goal of this chapter is to reach an axiomatization for the prefix-closed synchronous regular languages in an analogous way to our axiomatization of the prefix-closed regular languages. The main focus will be on prefix-closing the synchronous product, as everything else can be done analogously to the previous case.

5.1 The Prefix-Closure Operator on Synchronous Terms

We can define the prefix-closure operator in a similar way on $\mathcal{T}_{\mathsf{SF}_1}$ to our definition on $\mathcal{T}_{\mathsf{KA}}$. Namely, we first define the NZ-predicate analogously to before with addition

$$NZ(e \times f) \iff NZ(e) \wedge NZ(f).$$

Then we can define for $e \in \mathcal{T}_{\mathsf{SF}_1}$ the prefix-closure operator by:

$0^{pc} = 0$	$(e+f)^{pc} = e^{pc} + f^{pc}$	$(H(e))^{pc} = H(e)$
$1^{pc} = 1$	$NZ(f) \implies (e \cdot f)^{pc} = e^{pc} + e \cdot f^{pc}$	$Z(f) \implies (e \cdot f)^{pc} = 0$
$a^{pc} = 1 + a$ for $a \in \mathcal{T}_{\mathsf{SL}}$	$NZ(e) \implies (e^*)^{pc} = e^* \cdot e^{pc}$	$Z(e) \implies (e^*)^{pc} = 1$

where we additionally define the operator recursively on the synchronous product by

$$(e \times 0)^{pc} = 0 = (0 \times e)^{pc} \qquad NZ(e \times f) \implies ((a \cdot e) \times (b \cdot f))^{pc} = (a \times b)^{pc} + (a \times b) \cdot (e \times f)^{pc}$$
$$(e \times 1)^{pc} = e^{pc} = (1 \times e)^{pc} \qquad Z(e \times f) \implies ((a \cdot e) \times (b \cdot f))^{pc} = 0$$

where $a, b \in \mathcal{T}_{\mathsf{SL}}$.

Our first goal, as in the regular case, is to show that the SF_1-interpretation of this operator correctly expresses the prefix-closed SF_1-languages.

Lemma 5.1.1. *Let $e \in \mathcal{T}_{\mathsf{SF}_1}$. Then for any $xy \in [\![e^{pc}]\!]_{\mathsf{SF}_1}$, we have $x \in [\![e^{pc}]\!]_{\mathsf{SF}_1}$.*

Proof. As before, we go by induction on the expression. When $e = 0$ or $e = 1$, the result is clearly true. Furthermore, if $e = a \in \mathcal{T}_{\mathsf{SL}}$, then $e^{pc} = 1 + a$. We then have $[\![1 + a]\!]_{\mathsf{SF}_1} = \{\varepsilon\} \cup \{[\![a]\!]_{\mathsf{SL}}\} = [\![a]\!]^{pc}_{\mathsf{SF}_1}$. Additionally, if $e = H(f)$, then $[\![(H(f))^{pc}]\!]_{\mathsf{SF}_1} = [\![H(f)]\!]_{\mathsf{SF}_1}$, which

is equal to either $\emptyset$ or $\{\varepsilon\}$, which are prefix-closed.

The steps for $e = f + g$, $e = f \cdot g$, and $e = f^*$ are all exactly as in the proof of Lemma 3.2.4. The new step is that we must prove the result for the synchronous product We do this itself inductively, using the recursive definition of the prefix-closure operator.

First, if $e = f \times 0$, then $e^{pc} = 0$, and $[\![0]\!]_{\mathsf{SF}_1}$ is certainly prefix-closed. Second, if $e = f \times 1$, then $e^{pc} = f^{pc}$. But by induction $[\![f^{pc}]\!]_{\mathsf{SF}_1}$ is prefix-closed.

Third, suppose $e = (a \cdot f) \times (b \cdot g)$, where a and b are synchronous letters. If $Z(f \times g)$, then we are reduced to the case of 0, which is prefix-closed. Thus, we may assume $NZ(f \times g)$. Then $e^{pc} = (a \times b)^{pc} + (a \times b) \cdot (e \times f)^{pc}$. We note that $a \times b \in \mathcal{T}_{\mathsf{SL}}$, so $(a \times b)^{pc} = 1 + a \times b$.

Then $[\![e^{pc}]\!]_{\mathsf{SF}_1} = \{\varepsilon, \{a\}, \{b\}\} \cup \{\{a\}, \{b\}\} \cdot [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$. To see that this is prefix-closed, let $xy \in [\![e^{pc}]\!]_{\mathsf{SF}_1}$. Now, we have two cases:

First, $xy \in \{\varepsilon, \{a\}, \{b\}\}$. Then either $x = \varepsilon$ and $y \in \{\{a\}, \{b\}\}$, or vice-versa. Regardless, it follows that $x \in \{\varepsilon, \{a\}, \{b\}\} \subseteq [\![e^{pc}]\!]_{\mathsf{SF}_1}$.

Second, $xy \in \{\{a\}, \{b\}\} \cdot [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$. We split into multiple cases again.

Case 1: $x \in \{\{a\}, \{b\}\}$ and $y \in [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$. Then $x \in \{\varepsilon, \{a\}, \{b\}\} \subseteq [\![e^{pc}]\!]_{\mathsf{SF}_1}$.

Case 2: $x = x_1 x_2$ where $x_1 \in \{\{a\}, \{b\}\}$ and $x_2 y \in [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$. Because $[\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$ is prefix-closed due to the inductive hypothesis, we have $x_2 \in [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$, and hence $x = x_1 x_2 \in \{\{a\}, \{b\}\} \cdot [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1} \subseteq [\![e^{pc}]\!]_{\mathsf{SF}_1}$.

Case 3: $y = y_1 y_2$ with $xy_1 \in \{\{a\}, \{b\}\}$ and $y_2 \in [\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$. Then either $x = \varepsilon$ or $x \in \{\{a\}, \{b\}\}$. Hence, $x \in \{\varepsilon, \{a\}, \{b\}\} \subseteq [\![e^{pc}]\!]_{\mathsf{SF}_1}$.

It therefore follows that $[\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1}$ is prefix-closed for $f, g \in \mathcal{T}_{\mathsf{SF}_1}$, and hence $[\![e^{pc}]\!]_{\mathsf{SF}_1}$ is prefix-closed for any $e \in \mathcal{T}_{\mathsf{SF}_1}$. □

This leads us toward the next result, that the interpretation of the prefix-closure operator gives not just any prefix-closed language, but in fact the prefix-closure.

Theorem 5.1.2. *Let $e \in \mathcal{T}_{\mathsf{SF}_1}$. Then $[\![e^{pc}]\!]_{\mathsf{SF}_1} = [\![e]\!]^{pc}_{\mathsf{SF}_1}$.*

Proof. The previous lemma proves that $[\![e]\!]^{pc}_{\mathsf{SF}_1} \subseteq [\![e^{pc}]\!]_{\mathsf{SF}_1}$. We prove the reverse inclusion. The base cases $0, 1$, and $a \in \mathcal{T}_{\mathsf{SF}_1}$ follow analogously to the proof of Theorem 3.2.5. We can also see that $[\![(H(f))^{pc}]\!]_{\mathsf{SF}_1} = [\![H(f)]\!]_{\mathsf{SF}_1} = [\![H(f)]\!]^{pc}_{\mathsf{SF}_1}$.

Once again, the cases $e = f + g$, $e = f \cdot g$, and $e = f^*$ follow as in Theorem 3.2.5. We will therefore only show the case of the synchronous product, namely that $[\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1} = [\![f \times g]\!]^{pc}_{\mathsf{SF}_1}$ for any SF_1-expressions f and g. If either of f or g is 0 or 1, we reduce to a simpler case and are finished. If $f = a$ and $g = b$ are synchronous letters, then we also

reduce to the semilattice case, and the result follows.

Now, assume $[\![(f \times g)^{pc}]\!]_{\mathsf{SF}_1} \subseteq [\![f \times g]\!]^{pc}_{\mathsf{SF}_1}$. We will prove $[\![((a \cdot f) \times (b \cdot g))^{pc}]\!]_{\mathsf{SF}_1} \subseteq [\![(a \cdot f) \times (b \cdot g)]\!]^{pc}_{\mathsf{SF}_1}$. If $Z(f \times g)$, we return to the case of 0, so we may assume $NZ(f \times g)$.

Then $[\![((a \cdot f) \times (b \cdot g))^{pc}]\!]_{\mathsf{SF}_1} = \{\varepsilon, \{a\}, \{b\}\} \cup \{\{a\}, \{b\}\} \cdot [\![f \times g]\!]^{pc}_{\mathsf{SF}_1}$, using the inductive hypothesis to move ^{pc} to the outside of $[\![f \times g]\!]_{\mathsf{SF}_1}$. Meanwhile, we have $[\![(a \cdot f) \times (b \cdot g)]\!]_{\mathsf{SF}_1} = \{\{a\}, \{b\}\} \cdot [\![f \times g]\!]_{\mathsf{SF}_1}$.

Let us suppose $x \in [\![((a \cdot f) \times (b \cdot g)^{pc}]\!]_{\mathsf{SF}_1}$. We of course go by cases.

Case 1: $x \in \{\varepsilon, \{a\}, \{b\}\}$. If $x = \varepsilon$, then certainly $x \in [\![(a \cdot f) \times (b \cdot g)]\!]^{pc}_{\mathsf{SF}_1}$, as it is nonempty. If $x \in \{\{a\}, \{b\}\}$, then for any $y \in [\![f \times g]\!]_{\mathsf{SF}_1}$, $xy \in [\![(a \cdot f) \times (b \cdot g)]\!]_{\mathsf{SF}_1}$, so x is a prefix.

Case 2: $x \in \{\{a\}, \{b\}\}[\![f \times g]\!]^{pc}_{\mathsf{SF}_1}$. Then $x = x_1x_2$ where $x_1 \in \{\{a\}, \{b\}\}$ and $x_2 \in [\![f \times g]\!]^{pc}_{\mathsf{SF}_1}$. Hence there exists y such that $x_2y \in [\![timesg]\!]_{\mathsf{SF}_1}$, which implies $xy \in [\![(a \cdot f) \times (b \cdot g)]\!]_{\mathsf{SF}_1}$, so x is again a prefix. □

We have therefore correctly defined the prefix-closure operator at the level of SF_1-expressions, and can use what we have learned from this definition to axiomatize the prefix-closed synchronous regular languages.

5.2 The Prefix-Closed Synchronous F_1-Algebra

Definition 5.2.1. A *prefix-closed* SF_1 *algebra*, or $pc\mathsf{SF}_1$-algebra, is a tuple $(A, S, +, \cdot, *, \times, 0, 1, H)$ that satisfies all the axioms of an SF_1-algebra, alongside the prefix-closure axiom

$$H(e) = 1 \implies 1 + e = e.$$

In the definition of the prefix-closed Kleene algebra, we added an operator N meant to capture the NZ-predicate in a formal way. Here, we do not add such an operator, but this will not be a provlem, as we have seen moving from $pc\mathsf{KA}$'s to F_1-algebras that H exactly fulfills the role of N. The same will be true here.

We first form a new congruence on the $\mathcal{T}_{\mathsf{SF}_1}$, in analogy to $\equiv_N$ on the N-regular expressions. We let $\equiv_{pc'}$ be the smallest congruences on $\mathcal{T}_{\mathsf{SF}_1}$ by the $pc\mathsf{SF}_1$ axioms, where $\mathcal{T}_{\mathsf{SL}}$ again fulfills the role of a semilattice. Corresponding to this congruence, we also define a new function $[\![-]\!]_{pc'} : \mathcal{T}_{\mathsf{SF}_1} \to \mathcal{P}((\mathcal{P}_n(\Sigma))^*)$ inductively by:

$$[\![0]\!]_{pc'} = \emptyset \qquad [\![H(e)]\!]_{pc'} = [\![e]\!]_{pc} \cap \{\varepsilon\}$$
$$[\![1]\!]_{pc'} = \{\varepsilon\} \qquad [\![e + f]\!]_{pc'} = [\![e]\!]_{pc'} \cup [\![f]\!]_{pc'}$$
$$[\![a]\!]_{pc'} = \{[\![a]\!]_{\mathsf{SL}}\}$$
$$H(f^{pc}) \equiv_{pc'} 1 \implies [\![e \cdot f]\!]_{pc'} = [\![e]\!]_{pc'} \cup [\![e]\!]_{\mathsf{SF}_1} \cdot [\![f]\!]_{pc'} \qquad H(f^{pc}) \equiv_{pc'} 0 \implies [\![e \cdot f]\!]_{pc} = \emptyset$$
$$H(e^{pc}) \equiv_{pc'} 1 \implies [\![e^*]\!]_{pc'} = [\![e^*]\!]_{\mathsf{SF}_1} \cdot [\![1 + e]\!]_{pc'} \qquad H(e^{pc}) \equiv_{pc'} 0 \implies [\![e^*]\!]_{pc'} = \{\varepsilon\}$$

where we additionally define $[\![-]\!]_{pc'}$ on the synchronous product by

$H((e \times f)^{pc}) \equiv_{pc'} 0 \implies [\![e \times f]\!]_{pc'} = \emptyset$
$[\![e \times 1]\!]_{pc'} = [\![1 \times e]\!]_{pc'} = [\![e]\!]_{pc'}$
$H((e \times f)^{pc}) \equiv_{pc'} 1 \implies [\![(a \cdot e) \times (b \cdot f)]\!]_{pc'} = [\![a \times b]\!]_{pc'} \cup [\![a \times b]\!]_{\mathsf{SF}_1} \cdot [\![e \times f]\!]_{pc'}$

As before, this is not quite good enough, so we restrict $\mathcal{T}_{\mathsf{SF}_1}$ to $\mathcal{T}_{pcS}$, which is given by the grammar

$$\mathcal{T}_{pcS} \ni p := 0 \mid 1 \mid 1 + e \in \mathcal{T}_{\mathsf{SF}_1}$$

and likewise define $\equiv_{pcS}$ as the restriction of $\equiv_{pc'}$ to $\mathcal{T}_{pcS}$. For $p \in \mathcal{T}_{pcS}$ we define $[\![p]\!]_{pcS} = [\![p]\!]_{pc'}$.

The next lemma is a direct analogy to Lemma 3.3.6.

Lemma 5.2.2. *Let $p, q \in \mathcal{T}_{pcS}$.*

1. *$p \equiv_{pcS} q$ if and only if $p^{pc} \equiv_{\mathsf{SF}_1} q^{pc}$, and*
2. *$[\![p]\!]_{pcS} = [\![p^{pc}]\!]_{\mathsf{SF}_1}$.*

Proof. Let $p, q \in \mathcal{T}_{pcS}$. First, suppose $p \equiv_{pcS} q$. The base cases $q = 0$ and $q = 1$ certainly hold. We may then look at the cases $q = 1 + e$ where $e \in \mathcal{T}_{\mathsf{SF}_1}$. Then $p \equiv_{pcS} 1 + e$. Hence $p \equiv_{pc'} 1 + e$. Since $p \in \mathcal{T}_{pcS}$, we know that either $p = 1$, in which case we return to one of the base cases, or $p = 1 + f$ for some $f \in \mathcal{T}_{\mathsf{SF}_1}$. Now, the only possible way that $p \equiv_{pc'} q$ and not $p \equiv_{\mathsf{SF}_1} q$ is if p contains a term $g \in \mathcal{T}_{\mathsf{SF}_1}$ and q contains a term in the "same" position that is SF_1-equivalent to $1 + g$. Inductively, we can show that $(g + h)^{pc} \equiv_{\mathsf{SF}_1} (1 + g + h)^{pc}$, $(g \cdot h)^{pc} \equiv_{\mathsf{SF}_1} ((1 + g) \cdot h)^{pc}$, $(h \cdot g)^{pc} \equiv_{\mathsf{SF}_1} (h \cdot (1 + g))^{pc}$, $(g \times h)^{pc} \equiv_{\mathsf{SF}_1} ((1 + g) \times h)^{pc}$, and $(g^*)^{pc} \equiv_{\mathsf{SF}_1} ((1 + g)^*)^{pc}$. This largely follows from the fact that nonempty, prefix-closed languages contain the empty word, meaning $(1+e)^{pc} \equiv_{\mathsf{SF}_1} e^{pc}$ as long as $[\![e]\!]_{\mathsf{SF}_1} \neq \emptyset$. Essentially, the prefix-closure operator eliminates any anomalies that come from the addition of an extra 1 in an term of a nonzero SF_1-expression. Thus the first implication of (1) follows.

Conversely, we suppose $p^{pc} \equiv_{\mathsf{SF}_1} q^{pc}$. Then certainly $p^{pc} \equiv_{pcS} q^{pc}$, since all SF_1 axioms are also $pc\mathsf{SF}_1$ axioms. Thus, we need only show that $p \equiv_{pcS} p^{pc}$. Most of this result is analogous to the proof of Proposition 3.3.4, so we focus only on the synchronous product. First, in the case $p = 1 + 0 \times f$, we have $p^{pc} = 1 + 0$, which is certainly equivalent to p. Then we have the case $p = 1+1\times f$. Inductively, we can assume $(1+f)^{pc} \equiv_{pcS} 1+f$, so it follows that $p^{pc} \equiv p$. If we assume $(1 + f \times g)^{pc} \equiv_{pcS} 1 + f \times g$, then we can prove the result for the case $p = 1+(a\cdot f)\times(b\cdot g)$. If $Z(f\times g)$ (which is equivalent to $H((f\times g)^{pc}) = 0$), then we reduce to the case $p \equiv_{pcS} 1 \equiv_{pcS} p^{pc}$. Assume instead that $NZ(f\times g)$, or equivalently $H((f\times g)^{pc}) = 1$. Then $p^{pc} = 1+1+a\times b+(a\times b)\cdot(f\times g)^{pc} \equiv_{pcS} 1+a\times b+(a\times b)\cdot(f\times g)^{pc}$. But then the prefix-closure axiom implies $1+a\times b+(a\times b)\cdot(f\times g)^{pc} \equiv_{pcS} 1+(a\times b)(1+(f\times g)^{pc})$. Applying the inductive hypothesis followed by the prefix-closure axiom, this yields $1+(a\times b)(f\times g)$, so we are finished.

Now we may move on to the second point. This proof is largely analogous the the proof of Lemma 3.3.6 (2), again differing only by the addition of the synchronous product. As before, we focus on that case. Certainly $[\![1 + 0 \times f]\!]_{pcS} = \{1\} = [\![1^{pc}]\!]_{\mathsf{SF}_1}$. By induction, $[\![1 + 1 \times f]\!]_{pcS} = [\![1 + f]\!]_{pcS} = [\![(1 + f)^{pc}]\!]_{\mathsf{SF}_1}$. If $Z(e \times f)$, then we end up with

$[\![p]\!]_{pcS} = \{\varepsilon\} = [\![p^{pc}]\!]_{\mathsf{SF}_1}$ again, so we may assume $NZ(e \times f)$. Then assume $[\![1+e\times f]\!]_{pcS} = [\![(1+e\times f)^{pc}]\!]_{\mathsf{SF}_1}$. Then $[\![1+(a\cdot e)\times(b\cdot f)]\!]_{pcS} = \{\varepsilon, \{a\}, \{b\}\} \cup \{\{a\}, \{b\}\} \cdot [\![e\times f]\!]_{pcS}$. Applying our assumption and using distributivity of concatenation over union, we discover that this is precisely $[\![(1+(a\cdot e)\times(b\cdot f))^{pc}]\!]_{\mathsf{SF}_1}$. □

Theorem 5.2.3. *For every prefix-closed synchronous regular language L there exists $p \in \mathcal{T}_{pcS}$ such that $[\![p]\!]_{pcS} = L$.*

Proof. Let L be a prefix-closed synchronous regular language. First, if $L = \emptyset$, then we may certainly pick $p = 0$. Now we may assume $L \neq \emptyset$, in which case $\varepsilon \in L$. Because L is a synchronous regular language, we know there exists $e \in \mathcal{T}_{\mathsf{SF}_1}$ such that $[\![e]\!]_{\mathsf{SF}_1} = L$. But since $\varepsilon \in L$, it follows that $[\![1+e]\!]_{\mathsf{SF}_1} = [\![e]\!]_{\mathsf{SF}_1}$. It additionally follows that $[\![(1+e)^{pc}]\!]_{\mathsf{SF}_1} = [\![1+e]\!]_{\mathsf{SF}_1}$ since L is prefix-closed. Therefore we can take $p = 1+e$, and we have $L = [\![1+e]\!]_{\mathsf{SF}_1} = [\![(1+e)^{pc}]\!]_{\mathsf{SF}_1} = [\![1+e]\!]_{pcS}$. □

The above theorem lets us know that not only are all $pc\mathsf{SF}_1$-languages prefix-closed, but also that every prefix-closed synchronous regular language is captured as a $pc\mathsf{SF}_1$-language.

Theorem 5.2.4. *(Soundness and Completeness of $pc\mathsf{SF}_1$.) Let $p, q \in \mathcal{T}_{pcS}$. Then $p \equiv_{pcS} q$ if and only if $[\![p]\!]_{pcS} = [\![q]\!]_{pcS}$.*

Proof. Let $p, q \in \mathcal{T}_{pcS}$. Then by Lemma 5.2.2 (1) $p \equiv_{pcS} q$ if and only if $p^{pc} \equiv_{\mathsf{SF}_1} q^{pc}$. By soundness and completeness of SF_1, this occurs if and only if $[\![p^{pc}]\!]_{\mathsf{SF}_1} = [\![q^{pc}]\!]_{\mathsf{SF}_1}$, which by Lemma 5.2.2 (2) happens if and only if $[\![p]\!]_{pcS} = [\![q]\!]_{pcS}$. □

Chapter 6

Applications

6.1 A Brief Introduction to Finite Automata

As mentioned briefly in Chapter 1, automata theory is inextricably linked with formal language theory, with automata acting as acceptors for various kinds of languages. In particular, finite-state automata accept regular languages and inputs. But just what is a finite-state automaton?

Informally speaking, a finite automaton has a finite set of states that it moves between according to the inputs it receives. To begin our formal definition, we start with deterministic finite automata.

Definition 6.1.1. A *deterministic finite automaton*, or *DFA* [6], is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where

1. Q is a set of states,
2. Σ is an alphabet representing the input symbols,
3. $\delta : Q \times \Sigma \to Q$ is called the *transition function*,
4. $q_0 \in Q$ is the *starting state*, and
5. $F \subseteq Q$ is the set of *final states*.

Let A be a DFA and $x \in \Sigma^*$ be a word. Suppose $x = a_1 a_2 \dots a_n$ for some $n \in \mathbb{Z}_{\geq 0}$, where each $a_i \in \Sigma$. We can determine whether A *accepts* x in the following way:

We can apply the transition function at the starting state with the input a_1 to get $q_1 := \delta(q_0, a_1)$. We proceed in this manner defining $q_i := \delta(q_{i-1}, a_i)$ until we reach q_n. If $q_n \in F$, then q_n is a final state, and A accepts x as an input.

The set of all words that a DFA accepts as input is called the *language* of the DFA, and is given by

$$L(A) := \{x \in \Sigma^* \mid \delta(q_0, x) \in F\},$$

where $\delta(q_0, x)$ is shorthand for $\delta(q_{n-1}, a_n)$ in the terminology we have been using.

Example 6.1.2. Let $\Sigma = \{a, b\}$. We will define a DFA that accepts as input words that contain the substring ab. We must have a single starting state q_0, and a final state that can only be reached if the automaton has previously received the input a immediately followed by the input b. Let us call the final state q_2. In between, we will have an intermediate state, q_1, which represents the state in which ab has not yet appeared as input, but we have just received a as input. Therefore, our set of states will be $Q = \{q_0, q_1, q_2\}$, and the set of final states will be $F = \{q_2\}$. It is left to determine how the transition function will work.

Suppose that we are in state q_0 and we receive the input a. Then we are now in the state q_1, since our most recent input is a, but we don't have ab as a substring yet. Therefore,

$$\delta(q_0, a) = q_1.$$

On the other hand, if we receive b as input, then we must remain in state q_0, since $b \neq a$, and we do not have ab as a substring. Hence

$$\delta(q_0, b) = q_0.$$

Now, let us assume that we are in state q_1, i.e., our most recent input is a and we have not already received the substring ab. If we receive input a again, nothing changes, so

$$\delta(q_1, a) = q_1.$$

However, if we receive input b, then we have now received the substring ab, so we are allowed to stop if we wish. Thus,

$$\delta(q_1, b) = q_2.$$

Finally, if we are in state q_2, then we already have the substring ab, so any input we receive is fine, leading us to conclude

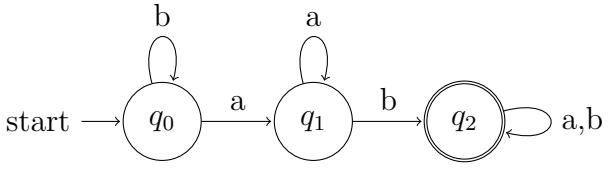
$$\delta(q_2, a) = \delta(q_2, b) = q_2.$$

Taken together, our DFA is given by

$$A = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, \{q_2\}),$$

where $\delta : Q \times \Sigma \to Q$ is defined as above.

Now, writing an automaton as a tuple is not the most intuitive way to display it, so sometimes we use a *transition diagram*, or a graph where the nodes represent states, and the edges represent the action of the transition function. We can represent the DFA from the previous example by the following:



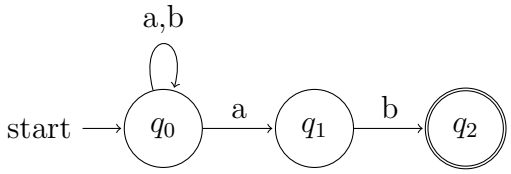
Up till now, we have been discussing deterministic finite automata, but we may also define *non-deterministic* finite automata:

Definition 6.1.3. A *non-deterministic finite automaton*, or *NFA* [6], is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where

1. Q is a set of states,
2. Σ is an alphabet representing the input symbols,
3. $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is called the *transition function*,
4. $q_0 \in Q$ is the *starting state*, and
5. $F \subseteq Q$ is the set of *final states*.

The difference between a DFA and an NFA is that an NFA has the ability to be in several states at once, as shown by the codomain of the transition function. However, it is known that every NFA accepts a language that is also accepted by a DFA. Furthermore, just as with DFAs, NFAs can also be expressed by means of a transition diagram.

Example 6.1.4. We present the diagram of an NFA, $A = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, \{q_2\})$ that takes as input strings ending in ab.



Note that $\delta(q_0, a) = \{q_0, q_1\}$, a subset of Q, in contrast to the transition function of a DFA.

Let $q \in Q$. We then expand δ to a function $\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$ inductively by

$$\hat{\delta}(q, \varepsilon) = q$$

and, if $x = x'a$ is a word with $x' \in \Sigma^*$ and $a \in \Sigma$, and $\hat{\delta}(q, x') = \{p_1, p_2, \ldots, p_k\}$ where $p_i \in Q$ for $i = 1, \ldots, k$, then we define

$$\hat{\delta}(q, x) := \bigcup \hat{\delta}(p_i, a).$$

Let A be an NFA. Then the language of A is given by

$$L(A) := \{x \in \Sigma^* \mid \hat{\delta}(q_0, x) \cap F \neq \emptyset\}.$$

Although DFAs and NFAs appear distinct, we have already claimed that the language of every NFA is also the language of a DFA. In fact, we can do better.

Theorem 6.1.5. *A language L is accepted by a DFA if and only if it is accepted by some NFA.*

The proof of this theorem involves a rather long construction which is outside the scope of this thesis, but which can be found in detail in Section 2.3.5 of [6]. Although DFAs and NFAs are in this sense equivalent, it can still be advantageous to work with NFAs. Not only are NFAs often easier to formulate, but also, in some cases, an NFA may have exponentially fewer states than the DFA that accepts the same language – n states compared to 2^n[9][7].

Now, before we connect these finite automata to regular languages, we require one more variation.

Definition 6.1.6. A *non-deterministic finite automaton with ε-transitions*, or ε-NFA, is an NFA in which transitions are allowed on the empty string. That is to say, an ε-NFA is a tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q, Σ, q_0, and F are defined as in an NFA, and $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ acts as the transition function.

The main difference between an NFA and an ε-NFA is that an ε-NFA may "spontaneously" make transitions, that is, make transitions upon input of the empty word, which is essentially no input at all. Just as before, we can expand δ to define word acceptance, and therefore define the language of an ε-NFA. Luckily for us, any language accepted by an ε-NFA is also accepted by the DFA given by using the construction given in [9]. Since DFAs accept the same languages as NFAs, this is tantamount to saying that we can eliminate the ε-transitions in an ε-NFA.

6.2 Regular Languages

Now that we know something about finite automata, we can begin with some applications of regular languages. First of all, we claim that regular languages are precisely the languages accepted by finite automata.

Theorem 6.2.1. *(Kleene's Theorem.) A language L over an alphabet Σ is a regular language if and only if it is accepted by some NFA (equivalently, DFA).*

A full treatment of the proof of this theorem is given in Sections 3.4 and 3.5 of [9].

Example 6.2.2. We will compute the regular expressions that give the languages of the two automata given above. Let $\Sigma = \{a, b\}$. Let A_1 be the DFA given in Example 6.1.2 and A_2 the NFA given in Example 6.1.4. Recall

$$L(A_1) = \{xaby \mid x, y \in \Sigma^*\}.$$

Let $L_1 = \Sigma$ and $L_2 = \{a, b\}$. Then we can see

$$L(A_1) = L_1^* \cdot L_2 \cdot L_1^*.$$

Similarly, we remember

$$L(A_2) = \{xab \mid x \in \Sigma^*\},$$

so, it follows that

$$L(A_2) = L_1^* \cdot L_2.$$

6.3 Prefix-Closed Regular Languages

Now, we might ask what type of automata accept prefix-closed regular languages as input. Since we know that regular languages are the languages of NFAs, we can assume that by applying the right restriction to an NFA, we can find the NFAs that correspond to prefix-closure. In fact,

Theorem 6.3.1. *A regular language L over an alphabet Σ is prefix-closed if and only if it is accepted by some NFA with all states final [7].*

The theorem is given without proof in [7], but we prove it here, using an analogous technique to the suffix-closed result given in [4]. In the proof of this theorem, we assume Theorem 6.2.1.

Proof. The first implication is easy: Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA with all states final, that is $F = Q$. By Kleene's Theorem, we know that $L(A)$ is regular. We wish to show that $L(A)$ is prefix-closed.

Suppose $xy \in L(A)$ for some $x, y \in \Sigma^*$. We have that $x \in L(A)$ if and only if $\hat{\delta}(q_0, x) \cap F \neq 0$. But $F = Q$ and $\hat{\delta}(q_0, x) \subseteq Q$, so this is obviously true. Thus $x \in L(A)$, and $L(A)$ is prefix-closed.

Conversely, let P be a prefix-closed regular language. Again, by Kleene's Theorem, there exists $A = (Q, \Sigma, \delta, q_0, F)$ an NFA such that $L(A) = P$, since P is regular. We will now construct an NFA, $\overline{A}$, with all states final such that $L(\overline{A}) = P$.

Define $\overline{A} = (Q, \Sigma, \delta, q_0, \overline{F})$, where $\overline{F} = Q$. This is an NFA that has all the same states and transitions as A, except that every state is final. We will prove that $L(\overline{A}) = L(A)$.

First, let $x \in L(A)$. Then $\hat{\delta}(q_0, x) \cap F \neq \emptyset$, which implies that $\hat{\delta}(q_0, x) \cap Q \neq \emptyset$. Hence $x \in L(\overline{A})$ and $L(A) \subseteq L(\overline{A})$.

Now, assume $x \in L(\overline{A})$. Then we know that $\hat{delta}(q_0, x)$ is a nonempty subset of Q. Hence there exists $q \in \hat{\delta}(q_0, x)$ and $y \in \Sigma^*$, such that $\hat{\delta}(q, y) \cap F \neq \emptyset$. Consequently $\hat{\delta}(q_0, xy) \cap F \neq \emptyset$, so $xy \in L(A) = P$. But P is prefix-closed, so $x \in P$. Consequently $L(\overline{A}) \subseteq L(A)$.

Therefore $P = L(A) = L(\overline{A})$. □

NFAs with all states final themselves have applications in modeling systems such as production lines, with composition of these systems represented by operations such as intersection and parallel composition of these automata [3].

Before discussing the next application of prefix-closed languages, it behooves us to define infinite strings, and what it means for a finite automaton to accept such objects. Whereas a word was defined as a finite sequence of symbols in an alphabet, an *infinite string* is simply any infinite sequence of such symbols. One can consider the finite prefixes of infinite strings as a sequence of increasingly accurate approximations converging to the

infinite string, something akin to Cauchy sequences in a complete metric space. sets of finite and infinite strings can be denoted byt their finite prefixes in the case the set is generated b a finite automaton

An infinite string is accepted by a finite automaton if it passes through an accepting state infinitely often. Then a theorem from [2] says the following:

Theorem 6.3.2. *Let A be a finite automaton. Let $T(q)$ denote the set of all finite and infinite strings accepted beginning from state q of A, and let $t_\perp(q)$ be the language given by all finite prefixes of $T(q)$. Then, for any two states p, q*

$$T(q) \subseteq T(p) \iff t_\perp(q) \subseteq t_\perp(p).$$

It is well known that the above results does not hold in general, meaning that there are languages of finite and infinite strings for which equivalence cannot be determined by their finite prefix closure only [2].

Chapter 7

Conclusion

In this thesis we have defined prefix-closed Kleene algebras and prefix-closed synchronous F_1-algebras in order to provide axiomatizations that are sound and complete with respect to the prefix-closed regular languages and prefix-closed synchronous regular languages respectively. These structures are not only interesting from a theoretical standpoint but also have widespread applications throughout programming languages and other areas of computer science.

One direction for further work might be to repeat this process of axiomatization whilst removing 0 from the grammar entirely. The goal would then be to capture all *nonempty* prefix-closed regular and/or synchronous regular languages. Often, removing $\emptyset$ from the conversation, so to speak, simplifies many problems and is desirable in computer science. For example, the need to define the NZ-predicate or make additional edits to the grammar to exclude multiplication by 0 as in Chapter 3 would be completely eliminated. Ideally such an axiomatization would be complete, but we leave this for future research

Zeroing in on the synchronous languages, we note that the definition of synchronous languages and the synchronous operator were very concrete, relying specifically on $\mathcal{P}_n(\Sigma)$ and the operation of union. Powerset with union is known to be a commutative monoid, so this brings to mind an abstraction, substituting instead another monoid, $(\Sigma, \otimes)$, and attempting to redefine and rediscover the properties of synchronous languages and the synchronous operator using this structure. Of course, why stop at a monoid? One might then ask if one could do the same thing with a group, which would introduces inverse actions and would therefore allow synchronization with so called "silent operations" in a computer science setting.

Finally, we note that the synchronous alphabet $\mathcal{P}_n(\Sigma)$ deliberately excludes the empty set from the alphabet. The apparent question then, is how one can rebuild synchronous languages and our various axiomatizations whilst allowing $\emptyset$ in the alphabet. This, too, has meaning in the realm of computer science. While synchronous languages allow one to express two operations occurring simultaneously, in lock-step with one another, the presence of the empty set would allow one string to be delayed whilst the other proceeds.

In short, whilst this thesis answers an important question regarding prefix-closed languages, one can certainly set out from it onto many open problems with direct applications in areas such as programming languages and computer network systems.

Chapter 8

Bibliography

[1] Arden, D. N.: "Delayed logic and finite state machines," Theory of Computing Machine Design, University of Michigan Press, 1–35, 1960.

[2] Bonsangue, M., Rot, J., Ancona, D., de Boer, F., Rutten, J.: *A Coalgebraic Foundation for Coinductive Union Types*, Esparza, J., Fraigniaud, p., Husfeldt, T., Koutsoupias, E. (eds), Automata, Languages, and Programming. ICALP 2014. Lecture Notes in Computer Science, Vol. 8573, 2014.

[3] Čevorová, K., Jirásková, G., Mlynárčik, P., Palmovský, M., Šebej, J.: "Operations on Automata with All States Final," Electronic Proceedings in Theoretical Computer Science, Vol. 151: 201–215, 2014.

[4] Gill, A., Kou, L.T.: "Multiple-entry finite automata," J. Comput. System Sci., Vol. 9: 1–19, 1974.

[5] Greibach, S.A.: "Formal Languages: Origins and Directions," Annals of the History of Computing, Vol. 3, No. 1: 14–41, 1981.

[6] Hopcroft, J., Motwani, R., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation*, Pearson, 3rd ed., 2007.

[7] Kao, J., Rampersad, N., Shallit, J.: "On NFAs where all states are final, initial, or both," Theoretical Computer Science, Vol. 410, No. 47–49: 5010–5021, 2009.

[8] Kozen, D.: "A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events," Information and Computation, Vol. 110, No.2: 366–390, 1994.

[9] Martin, J.C.: *Introduction to Languages and the Theory of Computation*, McGraw-Hill, 4th ed., 2011.

[10] Prisacariu, C.: "Synchronous Kleene Algebra," The Journal of Logic and Algebraic Programming, Vol. 79, No. 17: 608–635, 2010.

[11] Rot, J., Bonsangue, M., Rutten, J.: "Proving language inclusion and equivalence by coinduction," Information and Computation, Vol. 246: 62–76, 2016.

[12] Salomaa, A.: "Two Complete Axiom Systems for the Algebra of Regular Events," J. ACM, Vol. 13, No. 1, 1966.

[13] Silva, A.: *Kleene Coalgebra*. PhD thesis, Radboud Universiteit Nijmegen, 2010.

[14] Wagemaker, J., Bonsangue, M., Kappé, T., Rot, J., Silva, A.: "Completeness and Incompleteness of Synchronous Kleene Algebra," Hutton, G. (eds), Mathematics of Program Construction. MPC 2019. Lecture Notes in Computer Science, Vol. 11825, 2019.